École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

PhD Thesis

submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

of the Ecole Nationale Supérieure des Télécommunications

Specialty : Computer Science

Olivier Thonnard

A multi-criteria clustering approach to support attack attribution in cyberspace

Defense date: 31 March 2010

Committee in charge:

Prof. Dr. Hervé Debar (Telecom SudParis)	Reviewers
Dr. Mohamed Kaaniche (LAAS-CNRS)	
Prof. Dr. Daniel A. Keim (Univ. of Konstanz)	Examiners
Prof. Dr. Engin Kirda (EURECOM)	
Prof. Dr. Michel Grabisch (Univ. Paris I Panthéon-Sorbonne)	
Prof. Dr. Marc Dacier (EURECOM)	Advisors
Prof. Dr. Wim Mees (Royal Military Academy)	

École Doctorale
d'Informatique,
Télécommunications
et Électronique de Paris

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique

Olivier Thonnard

Vers un regroupement multicritères comme outil d'aide à l'attribution d'attaque dans le cyber-espace

Soutenue le 31 Mars 2010 devant le jury composé de

Prof. Dr. Hervé Debar (Telecom SudParis)	Rapporteurs
Dr. Mohamed Kaaniche (LAAS-CNRS)	
Prof. Dr. Daniel A. Keim (Univ. of Konstanz)	Examinateurs
Prof. Dr. Engin Kirda (EURECOM)	
Prof. Dr. Michel Grabisch (Univ. Paris I Panthéon-Sorbonne)	
Prof. Dr. Marc Dacier (EURECOM)	Directeurs de thèse
Prof. Dr. Wim Mees (Ecole Royale Militaire)	

" A man should look for what is, and not for what he thinks should be."

Albert Einstein

" Chance is a word void of sense; nothing can exist without a cause."

Voltaire

Abstract

In the recent years, the security community has observed a paradigm shift in the nature of attack phenomena in the Internet. Many security experts have acknowledged the fact that the cyber-crime scene is becoming increasingly organized and more consolidated. As organized crime has always gone where money could be made, cyber-criminals have, not surprisingly, developed an underground economy by which they are able to monetize all sort of malicious activities, like identity theft, fraud, spam, phishing, extortion, etc. Most of those illegal activities are presumably enabled by creating large groups of compromised machines (also called *botnets*) that are remotely controlled and coordinated by criminal organizations.

Even though there are some plausible indicators about the origins, causes, and consequences of these new malicious activities, very few claims can be backed up by scientific evidence. In particular, many questions remain regarding the attribution of the attacks and the organization of cybercrime. The main reason is that no global threat analysis framework exists to rigorously investigate emerging attacks using *different data sources* and *different viewpoints*. Up to now, much emphasis has been put on the development of data collection infrastructures, such that detailed information could be gathered on various aspects of Internet threats. However, current analysis techniques remain rather immature, and do not allow us to discover or extract new relevant knowledge about those coordinated attack phenomena.

The main contribution of this thesis consists in developing an analytical method to systematically address this complex problem related to *attack attribution* in cyberspace. Our approach is based on a novel combination of a graph-based clustering technique with a data aggregation method inspired by multi-criteria decision analysis (MCDA). More specifically, we show that it is possible to analyze large-scale attack phenomena from different viewpoints, revealing meaningful patterns with respect to various attack features. Secondly, we show how to systematically combine all those viewpoints such that the behavioral properties of attack phenomena are appropriately modeled in the aggregation process.

Consequently, our global threat analysis method can attribute (apparently) different security events to a common root cause or phenomenon, based on the combination of all available evidence. Perhaps even more importantly, our attack attribution technique can also enable a precise analysis of the *modus operandi* of the attackers. This can help an analyst to get better insights into how cybercriminals operate in the real-world, but also which strategies they are using.

Finally, thanks to its generic aspect, we are able to apply the very same approach to a broad range of security data sets without any fundamental modification. An experimental validation on two completely different data sets (i.e., honeypot traces and rogue antivirus websites) demonstrates the applicability and the effectiveness of our attack attribution method.

Résumé

Ces dernières années, les experts en sécurité ont observé un changement radical dans la nature des phénomènes d'attaque sur Internet. La plupart des experts ont d'ailleurs reconnu le fait que le phénomène global de cybercriminalité est devenu de plus en plus organisé et consolidé. Il est bien connu que le crime organisé s'est toujours développé là où d'importants profits financiers peuvent être réalisés, c'est pourquoi il n'est pas surprenant qu'une nouvelle économie "underground" se soit développée par laquelle les cybercriminels peuvent monétiser toutes sortes d'activités malveillantes, telles que le vol d'identité, la fraude, le spam, l'hameçonnage, l'extorsion, etc. La plupart de ces activités illégales semblent être facilitées par la création de véritables armées de machines compromises (appelées "botnets") qui sont contrôlées à distance et coordonnées par des organisations criminelles.

Même s'il existe des indices probables indiquant les origines, les causes et les conséquences de ces nouvelles activités malveillantes, assez peu d'affirmations peuvent être réellement soutenues et démontrées par des preuves scientifiques. En particulier, pas mal de questions subsistent concernant l'attribution des attaques et l'organisation de la cybercriminalité. Ceci est principalement dû au fait qu'il n'existe pas d'outil d'analyse de menace globale qui permet d'investiguer des phénomènes d'attaque en utilisant *différentes sources* de données et *différents points de vue*. Jusqu'à présent, beaucoup d'efforts ont été consacrés au développement d'infrastructures de collecte de données, de sorte que des informations précises et détaillées puissent être rassemblées à propos des menaces sur Internet. Par contre, les techniques d'analyse actuelles restent, quant à elles, plutôt immatures et ne permettent pas d'extraire des connaissances nouvelles et pertinentes sur ces phénomènes d'attaque coordonnés.

La contribution principale de ce travail est le développement d'une méthode analytique permettant d'aborder de manière systématique le problème de l'attribution d'attaque dans le cyber-espace. Le caractère innovant de l'approche choisie consiste à combiner une technique de regroupement basée sur les graphes, avec une méthode de fusion de données inspirée par le domaine de l'analyse décisionnelle multicritères (MCDA). Plus spécifiquement, nous démontrons d'abord qu'il est possible d'analyser des phénomènes d'attaque distribués à grande échelle à partir de différents points de vue révélant des motifs de corrélation intéressants par rapport à des caractéristiques d'attaque diverses. Ensuite, nous démontrons comment combiner systématiquement tous ces points de vue de sorte que les propriétés comportementales des phénomènes étudiés soient modélisées de manière adéquate dans le processus d'agrégation de données.

Grâce à la combinaison appropriée de tous les indices disponibles, cette méthode globale d'analyse de menace peut non seulement attribuer des événements semblant différents à première vue, à une même cause d'origine ou à un même phénomène, mais l'aspect sans doute le plus intéressant est qu'elle facilite aussi l'analyse des *modes opératoires* des attaquants. Par conséquent, ceci permet à un analyste d'avoir une meilleure vue globale sur les stratégies réellement utilisées par les cybercriminels.

Finalement, grâce à son aspect générique, la méthode peut aisément être appliquée à un large éventail de données, sans que cela ne nécessite de changement fondamental. Une validation expérimentale sur deux ensembles de données complètement différents (i.e., des traces d'attaque collectées par des honeypots et des sites web distribuant de faux logiciels antivirus) démontre l'applicabilité et l'efficacité de notre méthode d'attribution d'attaque.

Acknowledgements

The very first person I would like to thank is my advisor, Marc Dacier, without whom this thesis would not have been possible. Working with him is always challenging and it was a very enriching experience for me. I am very grateful for his guidance and his constant commitment to my research. In particular, I appreciated his impressive knowledge in many areas, his insight into security problems, and the sound advice he has provided during the numerous technical discussions we had together.

Also, I would like to thank sincerely my co-advisor from the Royal Military Academy, Wim Mees, who helped me to make progress in the right direction when it was necessary. He definitively inspired me in many ways thanks to his valuable expertise in the computer and security-related domains.

I would like to thank Hervé Debar, Head of the Telecommunications Networks and Services department at TELECOM SudParis, and Mohamed Kaaniche, Director of Research at LAAS-CNRS, for having accepted to review this work. The insightful comments they have provided helped me to further improve the quality of this dissertation.

I wish to thank Engin Kirda (EURECOM), Daniel Keim (University of Konstanz), and Michel Grabisch (University of Paris I Panthéon-Sorbonne), who have accepted to serve in my committee and have contributed to a very interesting and fruitful discussion during the PhD defense. It was an honor for me to have them as members of the jury.

During my research, I had the chance to work with several great researchers. This includes my "Leurré.com friends", in particular Corrado Leita and Van-Hau Pham, the Symantec research team with Angelos Keromytis and Marco Cova, not forgetting also the "ReSIST team", with Vladimir Stankovic, Ilir Gashi, Jouni Viinikka and Urko Zurutuza, and also many other brilliant guys of the WOMBAT Project. Thanks to all for the time spent together and for having shared your knowledge, this was a valuable and particularly interesting experience for me.

I am also grateful to many people from EURECOM who provided assistance in different ways during my recurrent stays in Sophia Antipolis. In particular, I enjoyed the technical discussions we had with Guillaume Urvoy-Keller, Taoufik En-Najjary, Eduardo Ramirez, and many others. Special thanks go to Gwenaëlle Le Stir, for her care and attention in processing all the administrative issues. Further I would like to extend my thanks to all Eurecom staff (business administration, library, IT department, and human resources) for their assistance in many organisational aspects surrounding this work.

I would like also to thank all my friends in Belgium and my colleagues of the Royal Military Academy, who gave me a continued moral support.

I wish to thank my entire extended family for the support they provided me throughout my life, and in particular during this challenging work. I am especially grateful to my parents, who came all the way to Sophia Antipolis to stand by me during the PhD defense.

Lastly, and most importantly, I wish to thank my dear little wife, Caroline, who has at all times supported me and given encouragement when it was most required. Without her endless patience, love and encouragement, the completion of this thesis would hardly been possible. I can't even find appropriate words to express my gratitude for all she has done for me.

To her, and to all people who supported me in any respect during my research, I dedicate this thesis.

Contents

1 Introduction **15**
1.1 Problem statement . . . 17
1.2 Research methodology . . . 18
1.3 Structure of the thesis . . . 19

2 Background and Related Work **21**
2.1 Foreword . . . 21
2.2 On the analysis of malicious activities . . . 23
2.2.1 Analysis of honeypot traffic . . . 23
2.2.2 Analysis of darknet traffic . . . 25
2.2.3 Analysis of IDS and firewall logs . . . 26
2.2.4 Analysis of large malware repositories . . . 27
2.2.5 Research on botnet tracking . . . 27
2.2.6 Cyber SA . . . 28
2.2.7 Preliminary conclusions . . . 29
2.3 Investigative data mining . . . 30
2.3.1 Security data mining . . . 30
2.3.2 Crime data mining . . . 31
2.4 Multicriteria decision analysis in security . . . 32
2.5 Summary . . . 33

3 Graph-based Knowledge Discovery in Security Datasets **35**
3.1 Introduction . . . 35
3.2 Attack Feature Selection . . . 37
3.3 Clustering analysis . . . 39
3.3.1 Introduction . . . 39
3.3.2 Similarity measures . . . 42
3.3.3 Graph-based clustering . . . 47
3.4 Evaluation of clustering results . . . 53
3.4.1 Estimating the optimal number of clusters . . . 54
3.4.2 Cluster validity indices . . . 57
3.4.3 Objective evaluation and comparison with other approaches . . . 59
3.5 Summary . . . 60

4 Attack Attribution using Multi-criteria Decision Analysis 63
4.1 Introduction . . . 63
4.1.1 Underlying motivation . . . 63
4.1.2 The MCDA approach . . . 65
4.1.3 Combining edge-weighted graphs using MCDA . . . 66
4.2 Formalizing the problem . . . 68
4.2.1 Aggregation function . . . 68
4.2.2 Building a combined graph G^*. . . . 69
4.2.3 Choice of an aggregation function . . . 71
4.3 Ordered Weighted Averaging (OWA) functions . . . 72
4.3.1 Classical OWA function . . . 72
4.3.2 Weighted OWA (WOWA) . . . 73
4.3.3 OWA functions: an illustrative example on attack events. . . . 75
4.3.4 Strategies to determine weights in OWA. . . . 80
4.3.5 A final note on OWA functions. . . . 83
4.4 On the use of Choquet integral . . . 84
4.4.1 Introduction . . . 84
4.4.2 Fuzzy measures . . . 85
4.4.3 Choquet integral . . . 87
4.4.4 Interaction indices among criteria . . . 89
4.4.5 Illustration on attack events . . . 90
4.4.6 Determination of fuzzy measures . . . 92
4.5 Summary . . . 97

5 Application to Network Attack Traces 99
5.1 Introduction . . . 99
5.1.1 Honeynet . . . 99
5.1.2 Terminology . . . 100
5.1.3 Experimental dataset . . . 101
5.2 Selection of attack features . . . 105
5.2.1 Spatial distributions of attackers . . . 105
5.2.2 Temporal correlation on the sensors . . . 106
5.2.3 Type of activity . . . 107
5.2.4 Common IP addresses . . . 107
5.2.5 Other possible features . . . 108
5.3 Graph-based clustering . . . 109
5.3.1 Distance metrics for measuring pattern proximities . . . 109
5.3.2 On mapping distances to similarities . . . 111
5.3.3 Cluster analysis of attack events . . . 114
5.4 Aggregation of all features . . . 119
5.4.1 Defining parameters . . . 119
5.4.2 Results overview . . . 121
5.5 Behavioral analysis of attack phenomena . . . 123
5.5.1 Introduction . . . 123
5.5.2 Coordinated Botnet . . . 124
5.5.3 Worm-behaving Cloud . . . 127

5.5.4 Botnet serving Messenger Spammers . . . 129
5.5.5 Cloud of Allaple Worms . . . 133
5.5.6 P2P aberrations . . . 134
5.6 Summary . . . 135

6 Application to Rogue Security Software **137**
6.1 Introduction . . . 137
6.1.1 Rogue AV ecosystem . . . 138
6.1.2 HARMUR data set . . . 140
6.2 Selection of site features . . . 142
6.2.1 Server IP addresses . . . 142
6.2.2 Whois information . . . 144
6.2.3 Domain names . . . 144
6.2.4 Other possible features . . . 146
6.3 Graph-based clustering . . . 147
6.3.1 Distance metrics for measuring pattern proximities . . . 147
6.3.2 Cluster analysis of Rogue domains . . . 150
6.4 Aggregation of all features . . . 155
6.4.1 Defining parameters . . . 155
6.4.2 Results overview . . . 157
6.5 Behavioral analysis of Rogue AV campaigns . . . 157
6.5.1 Introduction . . . 157
6.5.2 Two similar campaigns related to Anti-{virus, spyware}-2010 . . . 158
6.5.3 Two different campaigns within the .cn TLD . . . 160
6.5.4 An affiliate-based rogue network . . . 161
6.6 Summary . . . 162

7 Conclusion and Perspectives **167**
7.1 Research contributions . . . 167
7.1.1 Answers to the research questions . . . 167
7.1.2 Answer to the problem statement . . . 168
7.2 Future research . . . 169

Synthèse en français **171**

Appendix: list of figures best viewed in color print **205**

Bibliography **207**

Chapter 1

Introduction

" The Internet is the first thing that humanity has built that humanity doesn't understand, the largest experiment in anarchy that we have ever had."
– Eric Schmidt

Understanding the existing and emerging threats on the Internet should help us to effectively protect the Internet economy, our information systems and the net citizens. This assertion may look blindingly obvious to many people. However, things are less evident when looking more closely at the problem. Among security experts, there is at least one thing on which everybody agrees: combating cyber-crime becomes harder and harder [137, 33, 127]. Recent threat reports published by major security companies have also acknowledged the fact that the cyber-crime scene is becoming increasingly more organized, and more consolidated [157, 158, 72, 94].

There is obviously more at stake than just technical challenges, hacking fame, or digital vandalism. Money is at stake. In the last years, it has been often reported that cyber-criminals were building and maintaining an underground economy, which can offer the commoditization of activities, such as the sale of *0-day* exploits or new malware, the sale of compromised hosts, spamming and phishing resources, the sale of stolen credentials, etc [54, 158]. In most cases, these illegal and profitable activities are enabled by gaining control over *botnets* [11, 34, 126, 10] comprising thousands or even millions of machines, with many of those computers belonging to innocent home users. The worldwide spam problem is also largely due to those groups of compromised computers under the control of cyber criminal organizations. According to the 2009 Annual Security Report of MessageLabs [94], the annual average spam rate was 87.7% of all intercepted messages (an increase of 6.5% w.r.t. 2008), with 83.4% of this spam volume that originated only from botnets. As analyzed by SecureWorks [151], in 2008 the top botnets were collectively able of sending over 100 billion spams per day. Today, this figure has further increased to 107 billion spams per day [94].

Perhaps even more worrying, the analysis of recent "cyber conflicts", such as the pre-

sumed cases related to Estonia and Georgia [3, 37, 41], have led experts to the conclusion that botnets can be easily turned into digital weapons. Botnets can be used by cybercriminals (or dissidents) to attack the network resources of a country by performing Distributed Denial-of Service (DDoS) attacks against critical web services (e.g., DNS servers, network routers, government or financial websites, etc), which can lead to substantial economical or financial loss. A deep understanding of the long-term behavior of those new armies, and their evolution, is thus a vital requirement to combat effectively those latent threats [168].

Recently, we have also observed the monetization of another type of illegal activities through the propagation and distribution of rogue anti-virus software. Using social engineering, but also some highly-sophisticated techniques (such as the exploitation of client-side vulnerabilities or compromising legitimate web sites), cyber-crooks are able to distribute rogue AV programs, thanks to which they generate a substantial profit [159, 112, 36]. The business model of those miscreants is presumed to be an affiliate-based structure, with per-installation prices for affiliate distributors. The volume of profits generated for those cyber-criminals is impressive: earnings of as much as $332,000 a month were reported in affiliate commissions alone, as observed on the distribution website `TrafficConverter.biz` [77, 159].

Open questions in security

Since 2003, there seems to be a shift in the nature of attacks in the Internet, from server-side to client-side attacks and from fast spreading worms to profit-oriented activities like identity theft, fraud, spam, phishing, online gambling, extortion. Most of those illegal activities are supported by several large botnets controlled by criminal organizations. All the facts and figures presented in public threat reports are certainly valuable and help to shed some light on those cyber-criminal phenomena, but a lot of unknowns remain.

Current analysis techniques do not allow us to automatically discover new relevant knowledge about attack phenomena, certainly not from a strategic viewpoint. Today, there is still a gap of knowledge between what we believe to be happening, and what we can actually observe and prove *scientifically*. Even if there are some plausible indicators about the origins, causes, and consequences of these new malicious activities, very few claims can be backed up by scientific evidence. The main reason is that no global threat analysis framework exists to rigorously investigate emerging attacks using *different data sources*, and different *viewpoints*.

Consequently, many open issues remain. Who is behind the current attacks, i.e., how many organized communities are responsible for them? Where do they originate? How many organizations control the botnets? What are the emerging strategies used in cybercrime? Which "rogue networks" [152] are used as bullet-proof hosting (e.g., RBN[1], Atrivo a.k.a. Intercage, McColo, or 3FN, and maybe some others), but more importantly, how do they evolve over time ? Are botnets able to coordinate their actions?

As another example, we observe a growing number of malware of various types spreading all over the Internet, sometimes at a very high pace. For instance, companies like VirusTotal and Symantec receive hundreds of thousands of seemingly unique malware

[1]The Russian Business Network (RBN) is a multi-faceted cybercrime organization, which is notorious for its hosting of illegal and dubious businesses such as phishing, spam, child pornography and malware distribution `http://www.bizeul.org/files/RBN_study.pdf`.

samples per week. Figuring out which groups of malware samples are likely due to the same criminal organization, or could be linked to the same root phenomenon, is a daunting task for which no real solution has been found yet. To succeed, defenders need to have at their disposal efficient techniques that prioritize for them the malware they should first look at, depending on their likely impact. They must have tools and techniques to support them characterizing the threats and producing countermeasures in an automated way, as much as possible. The answers to such questions are extremely important, as they help decision makers to invest in the appropriate security measures. Without such knowledge, security decisions tend to be stabs in the dark.

Attack attribution

All previously described issues are related to a common security problem often referred to as "attack attribution". The main contribution of this thesis consists in developing an analytical method in order to systematically address the problem of *attack attribution*, i.e., how to attribute (apparently) different attacks to a common root cause, based on the combination of all available evidence.

By *root cause*, we do not refer to the identification of a given machine that has launched one specific, isolated attack. Instead, we are interested in having a better idea of the various individuals, groups or communities (of machines) that are responsible for large-scale attack phenomena. A method for attack attribution must also enable a precise analysis of the *modus operandi* of the attackers. As a result, it will also help an analyst to get better insights into how cybercriminals operate in the real-world, and the strategies they are using.

Note that the ultimate goal of this work is not to offer names of individuals to law enforcement agencies. The goal is, instead, to provide models of the acting entities that we are facing. Through generalization, these models can help in understanding the threats that every person or organization who connects to the Internet currently faces.

Finally, a last important requirement is the *generic aspect* of the method. We must be able to apply the very same approach to a broad range of security data sets, and thus the method should not be tailored to fit the needs of one specific data set only.

1.1 Problem statement

We have outlined here above some very important open issues in security, and the need for developing a systematic approach to facilitate the attack attribution process. This leads us to define the following problem statement.

> *Can we effectively address the problem of attack attribution in the cyber-space by means of a systematic, analytical approach?*

Conceptually speaking, this problem comes down to mining a very specific dataset, made of attack traces that are presumably representative of the various phenomena under scrutiny. To address the problem, we have thus focused on clustering and classification techniques applied to *attack events*, which are enriched with metada and contextual information. By grouping them based upon a series of common elements, we hope to be able to

derive semantically rich models but also to effectively associate new phenomena to previous ones. We will use all sort of metadata related to the attacks in order to group, as much as possible and in a meaningful way, the observed phenomena. Examples of information that could possibly help such an analysis are the origins of the attack, their timing, their spreading strategy, their coding style, their behavior, etc. These various characteristics will be also referred to as *attack features*.

In our approach, we hypothesize that coherent data sets (e.g., honeynet data, malware repositories, web crawlers) are available for the observation and analysis of these attack phenomena. Another reasonable assumption is that the high degree of coordination inherent to cybercriminal organizations should be reflected by various types of correlation patterns between different attack events. Those correlations are usually hidden in large security data sets, which justifies the use of data mining and knowledge discovery algorithms. Consequently, we formulate the following hypotheses:

- **Hypothesis 1 (H1)**: *attack phenomena can be observed through different sensors distributed in the cyber-space. The design and diversity of those sensors allow to collect observations whose features are representative and sufficiently discriminant for the phenomena under study.*
- **Hypothesis 2 (H2)**: *at any given point in time on which it can be observed, an attack phenomenon should exhibit at least a number of features k (with $k > 1$) that are correlated with precedent or subsequent observations of this very same phenomenon.*

From the problem statement and considering the hypotheses here above, we derive two specific research questions:

- **Research question 1 (RQ1)**: *how can we analyze attack phenomena from separate viewpoints, revealing different correlation patterns?*
- **Research question 2 (RQ2)**: *how can we systematically combine all viewpoints such that the behavioral properties of attack phenomena are appropriately modeled in the aggregation process?*

In particular, research question **RQ1** deals with the knowledge discovery process within security data sets, with the purpose of creating separate viewpoints with respect to meaningful *attack features*, whereas **RQ2** deals with the *fusion* of all viewpoints, by combining the extracted correlations in a systematic and effective way. Moreover, we will show in Chapter 4 that **RQ2** can be further divided into two sub-problems: (1) a problem related to the *dynamic aspect* of attack phenomena, i.e., the fact that certain attack features of a given phenomenon may evolve between two points in time on which we observe them; and (2) the problem of *uncertainty* which is inherently associated to any real-world phenomenon and to imperfect measurements.

1.2 Research methodology

A careful review of the relevant literature revealed the insufficiencies of existing analysis techniques for addressing the problem stated here above. This enabled us to establish an appropriate research methodology.

First, we have considered different data mining approaches, in particular unsupervised *clustering* techniques, in order to find meaningful patterns in a security data set [165]. Various types of statistical distances were evaluated such that the extracted patterns were truly reflecting meaningful correlations. Several experiments on network attack traces validated the soundness of our graph-based clustering approach. The results were quantitatively evaluated by means of cluster validity indices. We performed also a qualitative evaluation using different cluster visualization techniques (such as dimensionality reduction), which enabled us to assess the meaningfulness of the approach [39].

To address the aggregation problem, different *data fusion* and automated classification techniques were considered, e.g., clusters intersections (based on Formal Concept Analysis [166]), ensemble learning, artificial neural networks (ANN), Bayesian inference system, decision trees, Support Vector Machines (SVM), combinatorial optimization (Simulated Annealing and Hill Climbing), among others. Their capability and appropriateness for combining different security viewpoints was evaluated. However, the unsupervised aspect of the attack attribution problem led us to reject machine learning techniques whose effectiveness highly depends upon representative training data sets. Considering the intrinsic fuzzy aspect of real-world phenomena, *fuzzy inference* systems (Mamdani, Sugeno) were then evaluated, by which fuzzy rules could be elaborated to model arbitrary combinations of features among attack events [167, 168].

Consequently, we have further investigated different *aggregation functions* used in multi-criteria decision analysis (MCDA) as a way to combine multiple edge-weighted similarity graphs, with the purpose of finding connected components in the aggregated graph. The resulting clusters were evaluated quantitatively using the graph compactness index (on a per-feature basis), and qualitatively by visually assessing the correlations by means of multidimensional graphs.

Finally, the overall method was validated experimentally on two security data sets representing completely different attack phenomena [40, 36, 159].

1.3 Structure of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, we review the relevant literature and we compare our multi-criteria approach with other analysis techniques currently used in different domains related to threat monitoring, security data mining, and multi-criteria decision analysis.

Chapter 3 attempts to answer research question **RQ1**, by presenting a graph-based knowledge discovery technique that can be applied to security data sets in order to extract meaningful correlation patterns with respect to different attack features. In Chapter 4, we attempt to answer research question **RQ2**. In particular, we show how we can effectively combine or aggregate different attack features using a method based on *multi-criteria decision analysis* (MCDA).

Chapters 5 and 6 present in detail our experimental validation on two different security data sets. In Chapter 5, we have applied our method to a set of real-world attack traces collected by worldwide distributed honeypots during more than two years. An in-depth analysis of the experimental results is also provided to demonstrate the kind of insights we obtain into the behaviors of large-scale coordinated phenomena which have been called

Misbehaving Clouds. In Chapter 6, we present another experimental validation performed on a specific security data set made of rogue Anti-Virus web sites. By attributing rogue web sites to a common root cause, we show how our technique offers a unique perspective on how *rogue AV campaigns* and their server-side components are effectively organized, created and managed by cyber-criminal organizations responsible for them.

Finally, Chapter 7 concludes the thesis by presenting the answers to the research questions and to the problem statement. This final Chapter gives also some interesting perspectives on future research for improving our approach.

Chapter 2

Background and Related Work

" The art and science of data fusion is directly applicable in cyberspace for intrusion and attack detection."
– Tim Bass [12]

As introduced previously, the main contribution of this thesis consists in developing a generic analysis method applicable to a broad range of security data sets, in order to systematically address the problem of *attack attribution*, or how to attribute different attacks to a common root cause. This method must not only help an analyst to determine the root causes of attack phenomena in the cyber space, but also emphasize the *modus operandi* of attackers.

As a matter of fact, our research methodology lies at the crossroads of several research areas, which are briefly reviewed in this Chapter. We start by introducing the concept of *attack attribution*, as we refer to in this dissertation. Then, we review different research domains that are relevant to our discussion.: (i) the monitoring of malicious activities in the Internet; (ii) investigative and security data mining; and (iii) multi criteria decision analysis (MCDA), in particular in the security field.

2.1 Foreword

There is apparently no universally agreed definition for "attack attribution" in the cyber domain. If one looks at a general definition of the term *to attribute* in a dictionary, he will find something similar to: "explain by indicating a cause"[1], or "regard something as being caused by (someone or something)"[2]. However, most previous work related in cyber-security use the term "attribution" as a synonym for *traceback*, which consists in "determining the identity or location of an attacker or an attacker's intermediary" [186]. In the context of a cyber-attack, the obtained identity can refer to a person's name, an account, an alias, or similar information associated with a person, a computer or an organisation.

[1]Definition given by Merriam-Webster. http://www.merriam-webster.com/dictionary/attribute
[2]Definition given by the New Oxford American Dictionary.

The location may include physical (geographic) location, or any virtual address such as an IP address. In other words, *IP traceback* is a process that begins with the defending computer and tries to recursively step backwards in the attack path toward the attacker so as to identify her, and to subsequently enable appropriate protection measures. The rationales for developing such attribution techniques lie in the untrusting nature of the IP protocol, in which the source IP address is not authenticated and can thus be easily falsified. For this reason, most existing approaches dealing with IP traceback have been tailored toward (D)DoS attack detection, or eventually to some specific cases of targeted attacks performed by a human attacker who uses stepping stones or intermediaries in order to hide her true identity.

Some typical methods used for IP traceback include packet marking techniques ([135, 147, 15]), maintaining information on routers situated on the path between the attacker and the defender [146], packet watermarking techniques [179], and reconfiguring network paths during an attack (e.g., controlled link flooding [22]). An extensive survey of attack attribution techniques used in the context of IP traceback is available in [186].

In this work, we refer to "attack attribution" as something quite different from what is described here above, both in terms of techniques and objectives. Although tracing back to an ordinary, isolated hacker is an important issue, we are primarily concerned by larger scale attacks that could be mounted by criminal organizations, dissident groups, rogue corporations, and profit-oriented underground organizations.

We aim at developing an effective and systematic method that can help security analysts to determine the *root cause* of global attack phenomena (which usually involve a large amount of sources or events), and to easily derive their *modus operandi*. These phenomena can be observed through many different means (e.g., honeypots, IDS's, sandboxes, web crawlers, malware collecting systems, etc). Typical examples of attack phenomena that we want to identify and characterize can go from malware families that propagate in the Internet via code injection attacks, to zombie armies controlled by underground groups and targeting machines in the IP space, or even to rogue networks hosting large amounts of malicious websites deployed by cyber-crooks to propagate fake security software.

Attack phenomena are often largely distributed in the Internet, and their lifetime can vary from a few days to several months. They typically involve a considerable amount of features interacting sometimes in a non-obvious way, which makes them inherently complex to identify. Due to their changing nature, the attribution of distinct events to the same root cause can be a challenging task, as several attack features may evolve over time (e.g., geographical or network origins, targeted IP space, type of exploit or malware, etc).

As noted by Tim Bass in [12] , "Next-generation cyberspace intrusion detection (ID) systems will require the fusion of data from myriad heterogeneous distributed network sensors to effectively create cyberspace situational awareness [...] Multisensor data fusion is a multifaceted engineering approach requiring the integration of numerous diverse disciplines such as statistics, artificial intelligence, signal processing, pattern recognition, cognitive theory, detection theory, and decision theory. The art and science of data fusion is directly applicable in cyberspace for intrusion and attack detection".

Not surprisingly, our methods are at the crossroads of several active research domains. More specifically, we have categorized previous works that are relevant for this state-of-the-art as follows:

(i) general techniques used for analyzing *malicious activities* on the Internet, with an emphasis on methods that aim to improve the "cyber situational awareness" (Cyber-SA);

(ii) investigative and security data mining, i.e., knowledge discovery and data mining (KDD) techniques that are specifically tailored to problems related to computer security or intelligence analysis;

(iii) problems related to multi criteria decision analysis (MCDA), and multisensor data fusion.

In the next Sections, we give an overview of some key contributions in each research area. We also compare our approach with current analysis techniques used in each domain.

2.2 On the analysis of malicious activities

The experimental study of attacks on the Internet is a very active research domain. In the recent years, it has quite naturally gained a lot of attention, due to a substantial increase in cyber-criminal activities. To get a better understanding of the emerging threats that are targeting the Internet or people using it, we need to collect sound measurements about the ongoing attack phenomena observed worldwide on the net.

Broadly speaking, there are four classes of techniques used to monitor malicious network activities in the Internet: (i) honeypots; (ii) Internet telescopes and darknets; (iii) IDS and firewall logs sharing; and (iv) malware collection initiatives.

We give a brief overview of each of them here after. Note that we are primarily concerned by the *data analysis* perspective of each approach, rather than how we can build those various security data sets. Indeed, it is out of the scope of this dissertation to describe the technical details of each type of sensor. This information can be found in several previous works, such as in [187, 82, 117, 120], which provide extensive technical surveys of projects aiming at developing and deploying sensors to collect specific data on malicious activities.

2.2.1 Analysis of honeypot traffic

To observe malicious activities, a first approach consists in deploying so-called *honeypots* at several places in the Internet in order to collect unsolicited traffic. A classical definition of a honeypot was given by Lance Spitzner, which states that "a honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource" [149]. Informally, honeypots are vulnerable computers intentionally set up as traps to observe attackers on the Internet. Honeypots have normally no production value and hence should not see any legitimate traffic or activity. Whatever they capture can then be considered as malicious, or at least suspicious. A commonly-used approach to categorize honeypots is by considering their level of interactivity:

- honeypots performing simple service emulation are categorized as low-interaction honeypots, such as *Honeyd* [110], *Honeytrap* [183], *Nepenthes* [5] or *Billy Goat* [128];

- honeypots can offer more advanced emulation and are then called medium-interaction honeypots. A good example is *ScriptGen* [84], which implements a novel service emulation combined with an automated protocol learning approach (see also [82]);
- and finally, honeypots can also run real services (i.e., high-interaction honeypots), like *Argos* [119] or *Honeywall* developed by The Honeynet Project [162].

In most cases, honeypots are configured to listen to some unused (or "dark") IP addresses, and they wait passively for attackers to probe them. In this case, we talk about *server honeypots*. A contrario, *client-side honeypots* follow a different approach. Instead of waiting to be attacked, they are actively searching the Internet for other types of attacks that are usually linked to web threats hosted on malicious or compromised web servers, and targeting different vulnerabilities in client applications such as Internet browsers. Some examples of client-side honeypots (also called *honeyclients*) are *HoneyC* (a low interaction client honeypot [139]), *Capture* (a high interaction client honeypot solution, see [24]), *Strider HoneyMonkeys* [180], and *Shelia* [129], which can also emulate an e-mail client by examining e-mail attachments. More recently, Nazario has developed *PhoneyC* [107], a new virtual client honeypot solution (completely written in Python). PhoneyC is able to remove the obfuscation from many malicious pages, and can emulate specific vulnerabilities to pinpoint the attack vector.

In the recent years, honeypots have thus proved being valuable means for gathering and analyzing information about cyber-attacks. By extension, a network of interconnected honeypots has been termed "honeynet", and different projects take advantage of honeypot sensors to deploy large-scale, sometimes even complex infrastructures for collecting vast amounts of data regarding global malicious activities in the Internet. An example of such distributed architectures is the *Leurré.com* Project [86], a worldwide distributed honeypot deployment based on a modified version of Honeyd. In 2008, this project was improved and extended by the SGNET deployment [83, 85], which is based on a distributed set of *ScriptGen* sensors. Other well-known examples of distributed architectures are (among others) NoAH [108], Collapsar [73], Potemkin [178], ARAKIS [26] and Honeytank [176], which were extensively reviewed in other previous works such as those mentioned previously.

In the first experimental validation of this dissertation, we have used a particular data set made of attack traces collected for almost two years by a distributed set of honeypots (deployed by the Leurré.com Project). However, quite differently from all previous analyses of honeypot traffic, our approach leverages *multiple attack features* by relying on a multi-criteria aggregation process that can model the behavior of attack phenomena with little prior knowledge. Moreover, some key features used in our analysis include external or contextual information such as the spatial distributions (in terms of originating countries and IP subnets) of malicious sources involved in attack events observed by the honeypots. As we show in the experimental results, some unique insights can be obtained into the behavioral characteristics of large-scale phenomena (which we have called *Misbehaving Clouds*), and these could not have been found previously using more classical analysis techniques.

Finally, the previous work that is the closest to ours is due to Pouget in [120], where the author proposed a clique-based clustering approach to analyze correlations found in

honeypot traffic. Our work has extended but also largely improved this approach. In particular, we note the following improvements:

- a first notable difference concerns the data input to the correlation method. Pouget has applied a clique-based clustering to so-called *attack fingerprints* which represent long-lasting activities (usually, more than a year), whereas we propose to use *attack events* or relevant activity, which are defined on shorter periods of time (typically, a few days only). Such attack events provide a much finer granularity in the modeling of attack behaviors, and by consequent, a correlative analysis of those events gives far better results in terms of global phenomena.

- to perform the graph-based clustering, Pouget suggested the use of the *dominant set* framework developed in [116], and he acknowledged the importance of further investigating this approach. In Chapter 3, we justify the choice of this graph-based clustering approach by an objective comparison with other classical clustering methods (such as K-Means or Hierarchical Clustering).

- Pouget proposed the use of fairly simple distances to compute clusters (e.g., Euclidean distance, peak-picking). However, those distances proved experimentally to perform quite poorly w.r.t certain attack features for which statistical distributions must be compared. Instead, we propose to use statistical distances (such as Kullback-Leibler, Jensen-Shannon or Bhattacharyya), which are more appropriate for comparing distributions, as validated by our experiments.

- finally, Pouget proposed to combine multiple attack features by computing the intersections between clusters (or *cliques* as used in his work). While this approach has given interesting preliminary results, we show in Chapter 4 that this method does not hold in general for more complex phenomena. The reasons for this failure are twofold: *(i)* the problem of *uncertainty*, which is due to the fuzziness of attack phenomena; and *(ii)* the *dynamicity* problem, which is due to the evolving nature of real-world phenomena. We propose an elegant solution to both problems in Chapter 4.

Another significant difference with [120] (but also with all previous approaches) is the *generic aspect* of our method, i.e., it is not specifically tailored to honeypot traffic. We propose a formal and general solution to the *attack attribution* problem, such that it can be applied to any security data set without requiring any fundamental modification. This generic aspect is demonstrated in the last Chapter, in which we present the results of an experimental validation performed on a security data set made of rogue AV web sites provided by Symantec through the WOMBAT Project[3].

2.2.2 Analysis of darknet traffic

Darknets are large portions of unused IP subnets (or "dark subnets") that are used to monitor all unsolicited traffic directed to them. Several well-known projects operate darknets, such as CAIDA [23], the Internet Motion Sensor [6], or the Team Cymru darknet [164].

[3]WOMBAT: Worldwide Observatory of Malicious Behaviors and Attack Threats - http://www.wombat-project.eu.

However, we observe that darknets have been primarily used to analyze specific global phenomena that are essentially related to worm propagation ([141, 150, 100, 102]), or distributed denial-of-service (DDoS) attacks (using backscatter traffic [101]). Our approach is quite different, both in terms of techniques and objectives. Indeed, we do not only focus on a specific class of phenomenon (such as Internet worms), and by consequent our approach is not designed for the analysis of these phenomena only.

Still, other previous works have tried to analyze at a more global level the traffic collected by means of darknets. For example, Pang et al. [114] have tried to characterize the incessant, nonproductive network traffic (which they termed Internet *background radiation*) that can be monitored on unused IP subnets when deploying network telescopes or more active responders such as honeypots. They analyzed temporal patterns and correlated activity within this unsolicited traffic, and they found that probes from worms and autorooters heavily dominate. More recently, similar research has been performed by Chen et al. [30]. While all these previous works provide meaningful results and have much contributed in making advances in malicious traffic analysis, the traffic correlation performed by the authors relies on common statistical techniques. Indeed, they basically break down the components of background radiation by protocol, by application and sometimes also by specific exploit, and then apply some statistics across each component. Whereas we apply more elaborated techniques on honeynet traffic, such as graph clustering based on statistical distances, combined with multicriteria analysis in order to elevate the abstraction level, and to improve the insights into the behavior of global phenomena.

2.2.3 Analysis of IDS and firewall logs

Another class of techniques for gathering information about computer attacks aims at collecting and sharing firewall or IDS logs collected from a very large number of heterogeneous sources, sometimes even from home users. The most well-known project of this kind is probably D-Shield [45]. In [195], the authors studied the global characteristics and prevalence of Internet intrusions by systematically analyzing a set of firewall logs collected from a wide perspective. However, their study is a very general analysis that focused on the issues of volume, distribution (e.g., spatial and temporal), categorization and prevalence of intrusions. Their analysis is thus limited to basic statistics built around the IDS logs (e.g., distributions of ports, sources, targets, top sources, most prevalent threats, etc). Moreover, their analysis is limited to a 4-month observation period, whereas our experimental validation uses a 2-year honeynet trace. Still, it is worth noting that our results seem to be consistent with this previous analysis. In particular, they found that a very small collection of sources are responsible for a significant fraction of intrusion attempts in any given month, and their on/off patterns exhibit *cliques* of correlated behavior, which is quite similar to one aspect of the experimental results given in Chapter 5.

More recently, the open source community project *EmergingThreats* [47] (formally called *Bleeding Snort*) was created for sharing IDS logs. The purpose of this project is to provide some support to the Intrusion Detection community by producing a fast moving and diverse Snort Signature set and firewall rules (available for free). The objectives of this project are thus completely different from ours. However, we argue that analyzing such data set with our multi-criteria approach could yield very interesting results in terms of global phenomena (e.g., identifying which groups of intrusions seem to be linked to the

same root cause).

2.2.4 Analysis of large malware repositories

Another interesting and useful way of observing malicious activities in the Internet consists in collecting malware into large repositories. There are essentially two approaches for doing this:

- certain honeypot deployments take advantage of different techniques to collect new malware samples, such as SGNET [83], Mwcollect Alliance [104], Offensive Computing [111] or the Shadowserver Foundation [163]. Those deployments are usually maintained on a voluntary basis. However, there are also some commercially-supported initiatives with similar objectives, like Symantec DeepSight Services [35] (which goes far beyond malware analysis only), Support-Intelligence [155], Norman Sandbox [109], or Sunbelt CWSandbox [154], among others.

- other projects work instead on a community basis, i.e., unknown binaries may be uploaded by everyone who managed to catch a new suspicious binary file, and those samples are shared within the community. The samples are then further analyzed using anti-virus engines (such as VirusTotal [177]) or a sandbox (like Anubis [2]).

Besides the complexity in coordinating data collection efforts, the raw data collected by this kind of projects is overwhelming. For instance, companies like VirusTotal and Symantec receive hundreds of thousands of seemingly unique malware samples per week. A manual analysis of all these malware would be outrageously expensive. As a result, prioritization, grouping and automated pre-analysis of such data is needed. This is even more important today for community-based projects (such as VirusTotal) where a huge amount of new samples submitted on a daily basis are either false positives, or replicates of the same polymorphic malware family. For those reasons, finding which groups of malware samples are likely due to the same criminal organization, or could be linked to the same root phenomenon, is a daunting task for which no real solution has been found yet. Malware analysts need thus efficient warning and classification systems that can analyze and prioritize for them the malware they should first look at, depending on their likely impact. They must also have automated techniques to support them characterizing the threats and producing countermeasures in an automated way, as much as possible.

Recently, different techniques have been developed in order to enrich the collected code with *metadata* that might reveal insights into the origin of the code and the intentions of those that created, released or used it. By taking advantage of this metadata (e.g., specific code patterns, code behavior, code structure, etc) as well as some contextual information (e.g., the context in which the code sample was collected), we believe that a multi-criteria decision analysis method such as the one we propose in this dissertation could be of great help to malware analysts in order to achieve the goals mentioned here above.

2.2.5 Research on botnet tracking

Another more specific but quite active research area is related to botnet detection and botnet analysis [126, 10], or botnet mitigation [34, 79]. For example, *BotMiner* [64] is a

general botnet detection framework that is independent of botnet C&C protocol and structure, and it requires no a priori knowledge of botnets. The authors developed a prototype system that is based on: *i)* a two-steps clustering (based on X-Means) of C&C communication and activity flows of bots, so as to detect similarity patterns; and *ii)* the combination of both types of patterns by means of cross-correlation. While complementary in certain aspects, our research is also very different as we do not focus exclusively on the problem of detecting phenomena such as botnets, neither on directly disrupting them. Instead, we aim at understanding the high-level modus operandi of global attack phenomena (be they botnets or something else), e.g.: which "communities of machines" are involved in certain types of activities, on which (rogue) networks they are hosted, can we observe different botnets coordinating their efforts, etc. However, we acknowledge the utmost importance of those botnet-related works, since they can provide us additional *domain knowledge* that is necessary to model this type of phenomenon in our global analysis method.

2.2.6 Cyber SA

This review would be incomplete without mentioning the active research carried out in Cyber Situational Awareness (or Cyber-SA). *Situational awareness* is a military term referring to "the degree of consistency between one's perception of their situation and the reality". Extending this principle to the network security field, Yegneswaran et al. [196] envision a "network situational awareness" as analysts with accurate, terse summaries of attack traffic, organized to highlight the most prominent facets thanks to novel visualization techniques. In particular, a network SA environment should enable an analyst to quickly assess high-level information such as:

- the cause of an attack (e.g., a new worm, a botnet, or a misconfiguration), even when this attack has no known signature,
- whether the attacker specifically targeted the victim network, and
- if this attack event matches one seen in the past.

An important requirement of a network SA environment is thus its capability to provide high-level, meaningful representations that can help analysts to identify abnormal patterns, and get a better understanding of the possible root causes of attack phenomena occurring on their networks.

We acknowledge the seminal work of Yegneswaran and colleagues in this field, such as in [196] where they explore ways to integrate honeypot data into daily network security monitoring, with the purpose of effectively classifying and summarizing the data to provide ongoing situational awareness on Internet threats. However, their approach aims at providing tactical information, usable for the day-to-day operations, whereas we are interested in strategic information that reveal long term trends and the modus operandi of the attackers. Closer to our research, Li et al. have described in [198] a framework for automating the analysis of large-scale botnet probing events and worm outbreaks using different statistical techniques applied to aggregated traffic flows. They also design schemes to extrapolate the global properties of the observed scanning events (e.g., total population and target scope) as inferred from the limited local view of a honeynet.

Then, a first compilation of scientific approaches for Cyber-SA has recently been published in [145], in which a multidisciplinary group of leading researchers (from cyber security, cognitive science, and decision science areas) try to establish the state of the art in Cyber-SA and to set the course for future research. The goal of this pioneering book is to explore ways to elevate the situation awareness in the Cyber domain. We have contributed to [145] with a chapter on Macroscopic Cyber Situational Awareness, in which we present an extensive data collection infrastructure and illustrate the usefulness of applying a multidimensional analysis to the attack events detected by honeypots.

Finally, another interesting project is Cyber-Threat Analytics (Cyber-TA), founded by SRI International [38]. Cyber-TA is an initiative that gathers several reputed security researchers. This project maintains a malware data set that is built upon the logging capabilities of *BotHunter* [65]. According to the project description, it aims at accelerating the ability of organizations to defend against Internet-scale threats by delivering technology that will enable the next-generation of privacy-preserving digital threat analysis centers. Cyber-TA researchers argue that these analysis centers must be fully automatic, scalable to alert volumes and data sources that characterize attack phenomena across millions of IP addresses, and give higher fidelity in their ability to recognize attack commonalities, prioritize, and isolate the most critical threats. However, very few information is available on the public website of the project about which scientific techniques could enable organisations to achieve such goals or to elevate their cyber situation awareness.

2.2.7 Preliminary conclusions

Data collection infrastructures for monitoring malicious activities are now widely deployed and technically mature. Unfortunately, we note that current analysis techniques remain *immature* or aim at solving particular problems. Furthermore, most analyses are performed on a *reactive* basis, motivated by a specific attack phenomenon that has already occurred.

We are pursuing a fundamentally different goal than the one classical response teams are after. Their responsibility is to address in near real time newly observed events by, e.g, creating new antivirus signatures, investigating the spread of a new botnet, etc. This is what can be referred to as a *tactical* approach to the fight against Internet crimes. This work deals with the *strategic* battle instead, i.e., the observation of the modus operandi of the attackers and their strategies on a long-term basis. Both are, of course, complementary but involve different time scales. Tactical battles are won, or lost, in terms of minutes or hours whereas strategic approaches involve days, weeks or months.

Another pitfall of current analysis techniques is the limited use of *contextual information* and *collective intelligence*, probably by lack of effective analysis techniques to combine different viewpoints. In particular, we observe that every project is able to collect different key aspects related attack phenomena. However, these are usually only a few pieces of the overall puzzle, but in the end, we have thus an incomplete picture of cyber-criminal activities.

In summary, we note that most approaches for analyzing malicious activities are tailored to solving *specific issues*, by means of a particular data set, in a reactive manner. Current techniques do not allow us to automatically discover new relevant knowledge about attack phenomena from a strategic viewpoint. Furthermore, the shifting paradigm of Internet attacks has apparently created a gap of knowledge between what we believe to be

happening, and what we can actually observe and prove scientifically. Even if rumors and stories circulate within the security community about the origins, causes, and consequences of these new malicious activities, very few claims can be backed up by scientific evidence. More specifically, several issues remain open, e.g.:

(i) the *attack attribution* problem, i.e.: which group of attacks, malware samples, rogue domains,... are attributed to the same large-scale phenomenon? Who is behind the current attacks, and where do they originate? Perhaps more importantly, how can we link attack events that may look different to a common root cause?

(ii) the *organization of cybercrime*: how many organizations control the botnets? What are the new strategies used in cyber-crime? Which "rogue networks" are used as bullet-proof hosting, and how do they evolve over time?

(iii) other specific questions, such as: what is the spreading pattern over time? Are botnets able to coordinate their actions? How do attack phenomena evolve over time?

Developing a systematic analysis method that could provide some answers to such questions is thus extremely important, as it will not only help analysts to get better insights into how cybercriminals operate in the real-world, but it will also help decision-makers to invest in the appropriate countermeasures. Unfortunately, current analysis techniques can only provide *partial* answers to those questions. The multicriteria analytical approach proposed in this dissertation is meant to make advances toward filling this gap.

2.3 Investigative data mining

In this Section, we review some research efforts in a field referred to as *investigative data mining*, in which specific data mining techniques are tailored to fit the needs of investigative tasks.

2.3.1 Security data mining

In the last decenny, a considerable effort has been devoted to applying data mining techniques to problems related to computer security. However, a great deal of those efforts has been exclusively focused on the improvement of intrusion detection systems (IDS) via data mining techniques, rather than on the discovery of new fundamental insights into the nature of attacks or their underlying root causes [9]. Furthermore, only a subset of common data mining techniques (e.g., association rules, frequent episode rules or classification algorithms) have been applied to intrusion detection, either on raw network data (such as ADAM [7], MADAM ID [80, 81] and MINDS [48]), or on intrusion alerts streams [44, 75]. A comprehensive survey of Data Mining (DM) techniques applied to Intrusion Detection (ID) can be found in [8, 21].

We note that all previous approaches aim at improving alert classification or intrusion detection capabilities, or at constructing better detection models thanks to the automatic generation of new rules (e.g., using some inductive rule generation mechanism). Our work is very different in many aspects. First, we take advantage of data sets that contain only malicious activities, so the objectives of mining this kind of data set are quite different.

Secondly, we use a graph-based, unsupervised data mining technique to discover unknown attack patterns performed by groups or communities of attackers. Finally, our objective does not consist in generating new detection signatures to protect a single network, but instead to understand the root causes of large-scale attack phenomena, and get insights into their long-term behavior, i.e.: how long do they stay active, what is their average size, their spatial distribution, and how do they evolve over time with respect to their origins, or the type of activities performed.

2.3.2 Crime data mining

There are many similarities between the tasks performed by analysts in computer security and in crime investigations or in law-enforcement domains. As a result, several researchers have explored the possibilities of DM techniques to assist law-enforcement professionals. In [92], McCue provides real-world examples showing how data mining has identified crime trends and helped crime investigators in refining their analysis and decisions. Previous to that work, Jesus Mena has described and illustrated the usefulness of data mining as an investigative tool by showing how link analysis, text mining, neural networks and other machine learning techniques can be applied to security and crime detection [93]. More recently, Westphal provides additional examples of real-world applications in the field of crime data mining, such as border protection, money laundering, financial crimes or fraud analytics, and elaborates also on the advantages of using information-sharing protocols and systems in combination with those analytical methods [184].

We observe, however, that most previous work in the crime data mining field has primarily focused on "off-the-shelf" software implementing traditional data mining techniques (such as clustering, classification based on neural networks and Kohonen maps, or link analysis). Although very useful, those techniques are generally not appropriate for modeling complex behaviors of attack phenomena that we aim to identify. Still, Chen et al. have conducted some active research in crime data mining in the context of the COPLINK project [28], using text mining, neural networks and Social Network Analysis (SNA) on different case studies. In our graph-based approach, we can see some similarity with those link analysis methods used in crime data mining. However, there are also many differences: for instance, how relationships are created in classical link analysis tools is quite straightforward (usually, using the output of simple comparisons between basic features), whereas we use an aggregation function to combine multiple correlation patterns found in different graphs in order to identify more complex relationships. Furthermore, our approach can be applied to many different types of feature vectors, even to statistical distributions.

Finally, there are also some obvious relationships between our graph-based clustering technique and *Social Network Analysis* (SNA), which has been recognized as an appropriate methodology to uncover previously unknown structural patterns from social or criminal networks. SNA heavily relies on the usage of network graphs and link analysis as key techniques to analyze social communities and networks. Different metrics are used to emphasize the characteristics of a social group and its members, such as centrality, betweenness, closeness, structural cohesion of actors, clustering coefficient, etc ([138, 182]). In this context, analysis of *n-cliques*, *n-clans*, *k-plexes*, or more generally "connected components", can reveal interesting subgroups within a network or a community that are strongly connected in the graph representation, i.e., network members sharing many common traits. Probably

for those reasons, SNA has been ranked in the top 5 intelligence analysis methods by K. Wheaton[4] [185].

Some of our techniques are admittedly inspired by SNA, namely the clique-based clustering of attackers. However, we use a novel, efficient clique algorithm based on dominant sets, and our attribution method can combine multiple graphs (reflecting multidimensional features) into a combined graph by using a multicriteria aggregation function, which enables us to model more complex relationships among coalitions of features (e.g., an interdependency between two or more features). As far as we know, this kind of processing is not yet available in traditional SNA techniques.

2.4 Multicriteria decision analysis in security

In this work, we have formalized the attack attribution problem as an application of *multi criteria decision analysis* (MCDA), in which the criteria of concern are given by the links (or similarity) values computed during the graph-based clustering (which is performed iteratively for each attack feature). That is, we use the distance values between two events as *degrees of evidence* (or fuzzy measures) to decide whether they are likely due to the same root phenomenon. As such, it can be considered as a classical multi-attribute decision making problem where a decision has to be chosen based on several criteria. A combined output is evaluated based on different attributes (or features), which are expressed numerically and can even be obtained as the output of a fuzzy system (e.g., to model vagueness or uncertainty of a given attribute). It is worth noting that MCDA has also been ranked in the top 5 intelligence analysis methods by K. Wheaton [185].

In our formalization, we need, thus, to define an appropriate function that can model a decision scheme matching as closely as possible the phenomena under study. In many MCDA methods, the aggregation process is a sort of averaging function, like a simple weighted means. Some well-known examples of such methods include Simple Additive Weighting, Weighted Product Method, and the Analytical Hierarchy Process [197], or an Ordered Weighted Average (OWA) [192], and Choquet or Sugeno integrals [63, 62, 17, 173].

ELECTRE, TOPSIS and PROMETHEE are three other well-known outranking methods that are based on a similar averaging process [51]. These techniques aim at selecting or ranking different alternatives by averaging multiple criteria weighted by coefficients. Similarly to ELECTRE, we also combine multiple criteria using a *relational* approach. However, we further extend this approach by showing how to include more complex aggregation functions, such that interactions among coalitions of criteria (or attack features) can be modeled effectively. Despite their great flexibility in combining features or evidences, we note that rather few previous works have used MCDA approaches to address security-related problems.

Still, in [27] the authors consider the problem of discovering anomalies in a large-scale network based on the *data fusion* of heterogeneous monitors. The authors evaluate the usability of two different approaches for multisensor data fusion: one based on the Dempster-Shafer Theory of Evidence and one based on Principal Component Analysis.

[4] Kristan J. Wheaton is assistant professor of intelligence studies at Mercyhurst College. http://www.sourcesandmethods.blogspot.com/

The *Dempster-Shafer* theory is a mathematical theory of evidence based on belief functions and plausible reasoning [140]. It allows one to combine evidence from different sources and to obtain a certain degree of belief (represented by a belief function) that takes into account all the available evidence. It can be seen as a generalization of Bayesian inference where probability distributions are replaced by *belief functions*. When used as method for sensor fusion, different degrees of belief are combined using Dempster's rule which can be viewed as a generalization of the special case of Bayes theorem where events are independent. In our attribution method, we prefer using aggregation functions as described previously, for the greater flexibility they offer in defining how we want to model interactions among criteria (e.g., a positive or negative synergy between a pair of criteria). Moreover, in Dempster-Shafer all criteria are considered as independent of each other, which is usually not the case with features used in attack attribution. Interestingly, it has been showed that there is a direct connection between *fuzzy measures* used in MCDA, and *belief* or *plausability* functions used in Dempster-Shafer theory ([63, 181]).

2.5 Summary

As showed in this Chapter, collecting data related to different types of Internet threats has become a relatively common task for security researchers. However, *analytical methods* for effectively analyzing those vast amounts of data are still very immature. Deciding which samples or which attack patterns should be investigated first is still a challenging task today for security practitioners.

Another fundamental issue with the analysis of those security data sets is that very few methods enable analysts to identify and characterize global attack phenomena in a *systematic* and *reliable* way (e.g., based on multiple features used as decision criteria). Furthermore, we observed that no analytical means have been developed yet to emphasize the *modus operandi* of attackers, or to help finding the *root causes* of attack phenomena. In other words, there is a lack of rigorous and scientific methodologies that could allow analysts to easily discover new relevant knowledge about attack processes on the Internet.

For those reasons, we need new techniques for classifying, clustering and correlating the gathered information in order to guide and help the analysis process of the most important threats. To address those important problems, this dissertation will deal with the development of a new generic analysis method which is able to *combine different data sources and viewpoints*. The ultimate goal of this method is: (i) to help an analyst to identify and to determine the root causes of cyber attack phenomena, and (ii) to produce a precise analysis of the modus operandi of the attackers, by revealing patterns that may result from the grouping of apparently different attacks.

Chapter 3

Graph-based Knowledge Discovery in Security Datasets

"The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data."
– John W. Tukey

This Chapter introduces a generic, multi-criteria analysis method that is designed to address in a systematic way the *attack attribution* problem. We start by providing an overview of the proposed method, which is made of three principal components: (i) selection of *attack features* from a security data set; (ii) *graph based clustering*, which aims at discovering meaningful relations among patterns extracted from the data set; and (iii) *multi-criteria decision analysis* (MCDA), which takes advantage of previous steps to combine different graphs using an appropriate aggregation function.

The two first components, as well as the underlying rationales, are then described and illustrated in this Chapter. The third component (multi-criteria analysis) will be discussed in Chapter 4.

3.1 Introduction

A method for attack attribution should enable us to systematically discover, extract and combine patterns from a security dataset, according to a set of relevant features, and with limited knowledge on the phenomena being studied. By applying this method to security datasets (e.g., attack events, threats data set, network traces, IDS alerts, etc), our hope is to identify attack phenomena occuring at a larger scale, but also to help the analyst in their quest for discovering their root cause.

Our approach relies on a novel combination of graph-based clustering with a multi-criteria aggregation process. As illustrated in Fig. 3.1, the design of the method is based on three main components:

1. **Attack feature selection**: we determine which relevant *attack features* we want to include in the overall analysis, and we characterize each object of the data set according to this set of extracted features $\mathcal{F} = \{F_k\}$, $k = 1, \ldots, n$ (e.g., by creating feature vectors for each object);

2. **Graph-based clustering**: an undirected edge-weighted graph is created regarding every feature F_k, based on an appropriate distance for measuring pairwise similarities. Then, strongly connected components can easily be identified within each graph, so as to reveal relationships among objects that share common patterns w.r.t. a given feature;

3. **Multi-criteria aggregation**: in this information fusion process, we combine different graphs of attack features using an *aggregation function* that models the expected behavior of the phenomena under study.

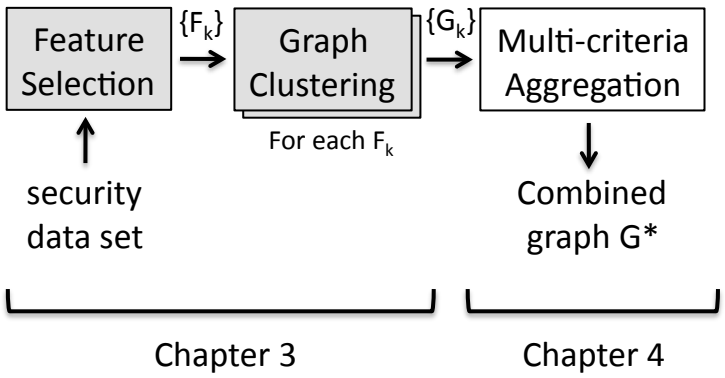


Figure 3.1: Overview of the attack attribution method proposed. $\{F_k\}$ refers to the set of features $\mathcal{F}$ selected from a security dataset, and $\{G_k\}$ refers to the set of corresponding edge-weighted graphs resulting from the clustering component. Finally, the multi-criteria component (described in Chapter 4) combines all G_k, $k = 1, \ldots, n$, so as to produce a *combined* node-link graph, which can support the root cause analysis thanks to a multi-dimensional graph visualization.

The approach is mostly unsupervised, i.e., it does not rely on a preliminary training phase to classify objects or events to larger scale phenomena. Instead, we have only a data set of unlabeled observations, and we need to learn what patterns w.r.t. each feature F_k are present in the data set. Then, these patterns have to be combined in a meaningful way, so as to emphasize the underlying phenomenon that may have caused the observations.

In this Chapter, we extensively describe the two first components (i.e., *feature selection* and *graph-based clustering*), whereas the multi-criteria aggregation will be discussed in the next Chapter. More precisely, we will focus on a novel graph-based clustering approach and how we can take advantage of it in the context of attack attribution. Each important step of the two first components will be illustrated by means of a real-world data set made of attack events collected in the Internet. We further validate the choice of our graph-based clustering approach by applying different objective evaluation methods. Finally, we also demonstrate the meaningfulness of the approach by comparing our clustering results

against two other approaches that are commonly used in clustering, namely Hierarchical clustering and K-Means.

3.2 Attack Feature Selection

In many data mining procedures, one of the very first steps consists in selecting some key characteristics from the data set, i.e., salient features that may reveal interesting *patterns*. As described by [70], typical clustering activities involve the following steps:

(i) feature selection and/or extraction;

(ii) definition of an appropriate distance for measuring similarities between pairs of feature vectors;

(iii) applying a grouping algorithm;

(iv) data abstraction (if needed), to provide a compact representation of each cluster;

(iv) assessment of the clusters quality and coherence (also optional), e.g., by means of validity indices.

In this Section, we turn our attention to step (i), which is implemented in the first component of our attribution method. Subsequent steps will be further detailed in Section 3.3.

Feature selection is the process of identifying, within the raw data set, the most effective subset of characteristics to use in clustering. The selection of these features may optionally be completed by a *feature extraction* process, i.e., one or more transformations of the input to produce features that are more suited to subsequent processing. Pattern representation refers to the number of categories, classes, or variables available for each feature to be used by the clustering algorithm.

More formally, we have thus a data set $\mathcal{D}$ composed of m objects, which are usually defined as security *events*. We define a feature set $\mathcal{F}$ made of n different features F_k, $k = 1, \ldots, n$, that can be extracted for each event e_i from $\mathcal{D}$ ($i = 1, \ldots, m$).

Let us denote by $\mathbf{x}_i^{(k)}$ the feature vector extracted for the event e_i w.r.t. feature F_k. In fact, $\mathbf{x}_i^{(k)} \in \mathbb{R}^d$ is a d-dimensional vector of real values, i.e.:

$$\mathbf{x}_i^{(k)} = \{x_{i,1}^{(k)}, \ldots, x_{i,d}^{(k)}\}$$

where d is the dimension of the vector and is a function of the feature F_k.

Finally, we can group all feature vectors defined w.r.t. a given feature into a data set $\mathbf{X}_k = \{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_m^{(k)}\}$. In many data mining books, it is customary to use a matrix notation to represent a set of feature vectors $\mathbf{X}_k$, i.e.:

$$\mathbf{X}_k = \begin{bmatrix} x_{1,1}^{(k)} & x_{1,2}^{(k)} & \cdots & x_{1,d}^{(k)} \\ x_{2,1}^{(k)} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{m,1}^{(k)} & \cdots & \cdots & x_{m,d}^{(k)} \end{bmatrix}$$

where the i^{th} row represents the feature vector $\mathbf{x}_i^{(k)}$ extracted for the event e_i of $\mathcal{D}$, obtained for the k^{th} feature F_k.

Summarizing, our problem space is made of three main dimensions: m is the number of security events, n is the number of attack features, and d is the dimension of the feature vector (the latter is variable and is a function of each considered feature F_k).

Illustrative example on an attack data set.

Let us consider an example of feature vector $\mathbf{x}_i^{(k)}$ for a given attack feature F_k. For this purpose, let's assume we are able to collect data about computer attacks in the Internet. Moreover, we consider that these attacks manifest themselves under the form of *attack events*[1], which involve a group of malicious sources (or computers) that seem to coordinate their actions. If we are able to observe those events, we can also record their origins, and as a result, we can define an attack feature F_k that represents the spatial distribution of those attacking machines, for example in terms of originating countries. By convenience, we will denote this feature as F_{geo}.

So, this will lead to the creation of a feature vector $\mathbf{x}_i^{(geo)}$ for each event e_i, and every element of the vector will hold the number of observations that correspond to a given country (i.e., the number of attackers coming from that specific country). Hence, this set of possible (or observable) countries will define the total number of categories, which sets also the dimensionality d of the feature vectors.

In the Internet, we can observe attacks coming from almost every possible country. As a result, for each event e_i we create a feature vector $\mathbf{x}_i^{(geo)}$ made of $d = 229$ positions corresponding to all countries (ordered alphabetically) where potential attackers may apparently reside. In other words,

$$\mathbf{x}_i^{(geo)} = \{x_{i,1}^{(geo)}, \ldots, x_{i,229}^{(geo)}\}$$

Fig. 3.2 gives an example of such a feature vector for a given attack event (with i=128), which has been observed by a sensor located in France in October 2006. The top attacking countries of this event are: US ($x_{128,215}^{(geo)} = 51$), CN ($x_{128,47}^{(geo)} = 10$), CA and DE (8 observations each). Another standard representation for this type of feature vector would be under the form of relative frequencies, e.g.: US(35%),CN(7%),DE(5%),CA(5%), others(47%).

On selecting attack features

The geographical origin of attackers is obviously only one possible feature that may be useful in a global root cause analysis. A strong advantage of our approach consists in combining in an effective manner multiple attack features according to a multi-criteria analysis process. In attack attribution, there is unfortunately no theoretical guideline supporting the selection of features to include in a cluster analysis. This choice highly depends on each specific situation, so it is up to the analyst to define carefully those feature vectors, based on facts and conjectures about the phenomena that have created the observations of the data set. In many cases, the features are quantitative (i.e., made of numerical values, defined either on a continuous or discrete domain), like in the example given here above. However, nothing forbids us from using also qualitative features in the

[1] A more complete and formal definition of such an attack event will be given in Chapter 5 when we will apply this method to a set of network attack traces.

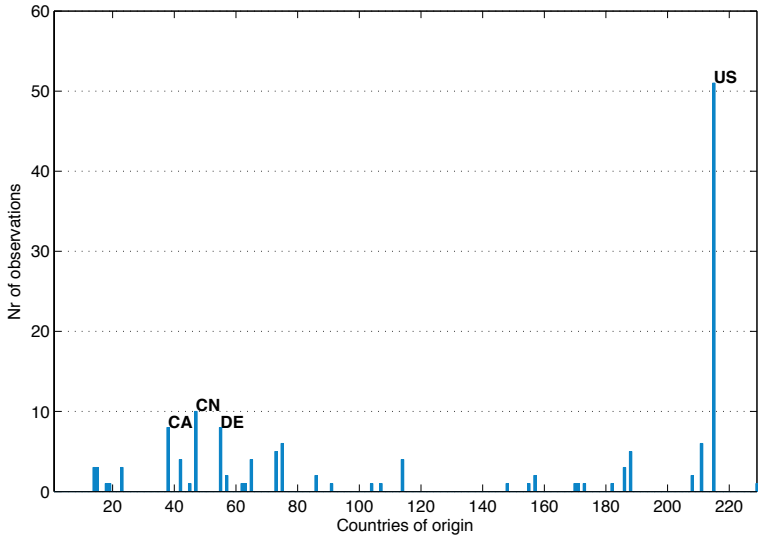


Figure 3.2: Illustrative example of a feature vector for an attack feature defined as F_{geo}. The vector represents the spatial distribution of attacking machines for a given attack event (e_{128}) observed by a sensor located in France in October 2006, i.e., it is a representation of $\mathbf{x}_{128}^{(geo)}$.

clustering process, like ordinal values (e.g., a ranking, or subjective characteristics like "suspicious", "benign", "malicious", etc) or unordered values (e.g., a set of email addresses).

3.3 Clustering analysis

3.3.1 Introduction

Cluster analysis and EDA.

Cluster analysis aims at finding natural groupings from a data set. It is essentially an approach relying on *exploratory data analysis* (EDA), in contrast with confirmatory data analysis where the analyst is mostly concerned with statistical hypothesis testing, model estimation or classification problems (John Tukey [174]). Regarding this exploratory aspect, *clustering* refers to a process of *unsupervised classification* by which unlabeled patterns are grouped into clusters based on a measure of similarity. As a result, all patterns found in a "valid" cluster should be more similar to each other than they are to patterns of another cluster. The goal of clustering consists in discovering interesting and meaningful patterns from a data set, without any prior knowledge on the phenomena being studied. In the realm of attack attribution, this can be helpful to understand the underlying phenomena that may have created the observations or the measurements. It is also useful to provide a data abstraction level, since every cluster can then be described by a cluster prototype that is representative of the attack patterns being grouped in that cluster.

There exists a plethora of clustering algorithms, which can be roughly categorized as either *partitional* or *hierarchical*. Partitional techniques aim at finding the most effective partition of the data set by optimizing a clustering criterion (e.g., minimizing the sum of

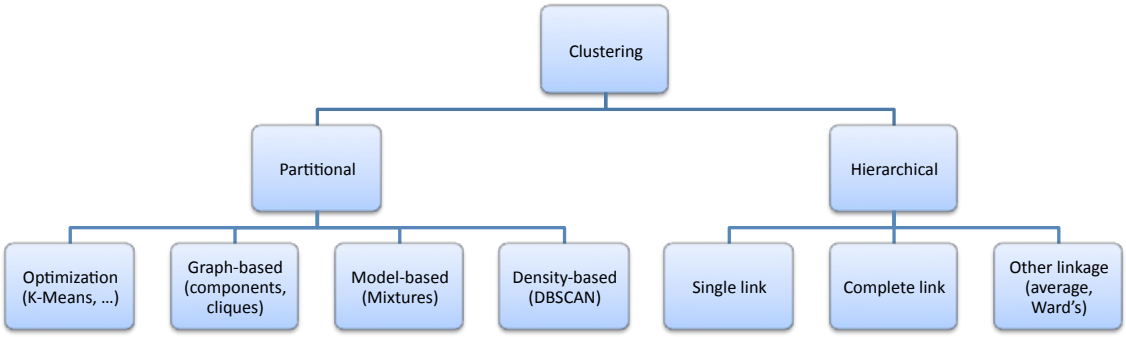


Figure 3.3: A general taxonomy of classical clustering approaches.

squared distances within each cluster). Hierarchical clustering methods produce a nested series of partitions (i.e., a hierarchical tree-like structure) in which the decision to merge objects or clusters at each level is performed based on a similarity criterion and a linkage method (e.g., the smallest or largest distance between objects).

Partitional techniques usually require the user to specify a predetermined number of clusters ahead of time, which can be seen as a drawback of these techniques (even though some additional techniques may help to evaluate the approximate number of clusters within a data set). On the other hand, optimization techniques normally require only the data as input, and not all interpoint distances as in hierarchical methods, which is an advantage when dealing with very large data sets. Note that graph-based clustering methods usually require all interpoint distances, even though they are usually considered as partitional techniques.

In both categories, we can further distinguish different strategies, based on how each clustering algorithm works to group data objects. In Fig. 3.3, we give a rough classification of classical approaches used in clustering, based on the taxonomy given by Jain et al. [71].

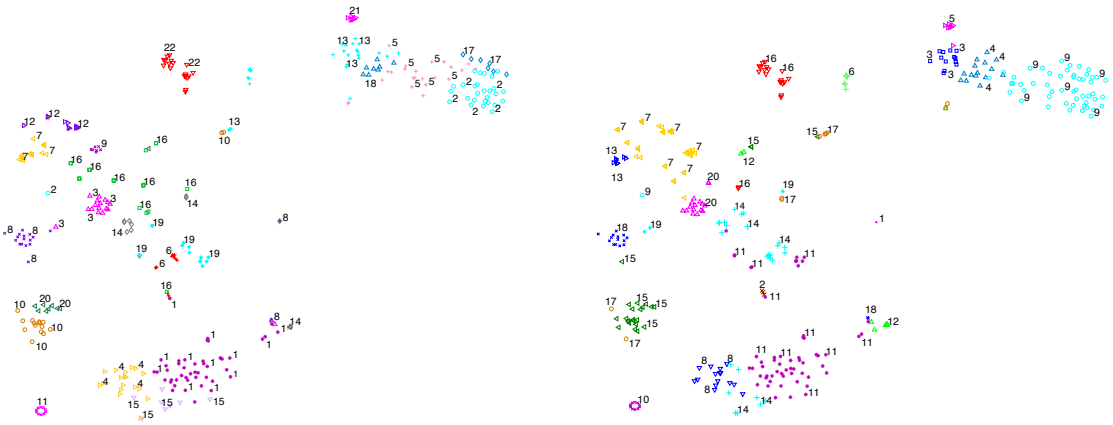
Note that there are also other cross-cutting aspects that can further characterize clustering algorithms, such as: *i)* agglomerative vs divisive clustering; *ii)* hard vs fuzzy clustering (where in fuzzy clustering, you may have overlapping clusters, and thus each object has a degree of membership to each cluster); *iii)* deterministic vs stochastic clustering; and *iv)* incremental vs non-incremental clustering [49, 76, 160, 70].

The intrinsic problem of clustering.

Since clustering is mostly a *data-driven* process, it can be hard sometimes to define what really constitutes a cluster, as underlined by several authors (e.g., [71, 91, 160]). Indeed, most clustering techniques rely on several input parameters which can largely influence the results. Furthermore, some algorithms assume some sort of structure for the clusters (e.g., spherical, elliptical, etc). Thus, if they are given a certain data set, most clustering algorithms will find clusters, regardless of whether they are really present in the data or not.

To illustrate this problem, let's consider a real data set in Fig. 3.4 where two clustering results were obtained by two different algorithms (using a graph-based algorithm in (a),

and using K-Means clustering in (b)), with the same number of clusters set as input to both algorithms. The different clusters are represented by a combination of a marker type and a color. As we can observe, there are significant differences between the groupings. Some clusters are overlapping with others, and certain points are (apparently) misclassified; but more importantly, how can we know which result makes more sense than the other when there is no clear underlying structure? Actually, by considering only the visual evaluation of both plots, it can be quite difficult to decide. Finally, another important issue relates to the estimation of the "correct" number of clusters hidden in the data set.



(a) Graph-based clustering (b) K-Means clustering

Figure 3.4: Some clusters obtained on a real data set of attack events, with two different clustering algorithms: (a) a graph-based clustering, and (b) K-Means. The different clusters are represented by a combination of a marker type and a color. In both cases, the number of desired clusters was set to 20 (not all clusters are depicted here).

On the choice of a clustering approach.

As suggested here above, we observe that clustering real data sets can be a difficult task, and thus different clustering methods will probably yield different results. For this reason, our attribution method is not necessarily limited to only one clustering algorithm. However, we have a preference for a novel graph-theoretical approach called *dominant sets*, and we already motivate this choice with the following reasons:

- the simplicity to formulate the problem, i.e., by representing the graph by its edge-weighted adjacency matrix (or proximity matrix);
- the graph-based approach does not require a number of clusters as input;
- to identify clusters in a graph, we can rely on a novel *clique-based* algorithm that can naturally extract the most significant groups in the first stages of the algorithm (as detailed hereafter);
- finding *cliques* in a graph can be formulated as a straightforward continuous optimization technique by relying on dominant sets. This is interesting since it can be

coded in a few lines of any high-level programming language, and it could be easily implemented in a parallel network, if scalability becomes an issue;

- as it will become clear in the multi-criteria analysis (in Chapter 4), different graphs (obtained for different attack features) can be easily combined using different types of aggregation functions (e.g., averaging functions, fuzzy integrals, etc) that enable us to model the phenomena under scrutiny, even when those phenomena have dynamic behaviors.

Thus, regarding this clustering component, we argue that the only requirement in this attribution method is to use a *pairwise clustering* technique where all interpoint distances are calculated ahead of time. We are aware of the fact that this approach can be computationally intensive for very large data sets, especially regarding the memory requirements. However, we argue that the computation of those pairwise similarities can be easily parallelized since all computations are independent of each other. Many database systems can even provide support for storing and indexing structures like proximity matrices.

On the other hand, as it will become clear in Chapter 4, *combining multiple features* using special aggregation functions can help to circumvent the intrinsic drawbacks of any clustering procedure (i.e., the difficulty to obtain groups that truly reflect the real underlying phenomena). That is, with this approach, we hope to reinforce the linkage of data objects (e.g., security events) that are related to the same root phenomenon, and at the same time, to break unfortunate linkage between unrelated objects which could be grouped by accident, as an artefact of the clustering algorithm used to explore the data set.

3.3.2 Similarity measures

Before turning to the clustering process by itself, we need to briefly discuss the issue of defining similarity measures. As stated previously, most clustering algorithms rely on certain metrics or distances to group objects into clusters. A *similarity measure* is a function that indicates how alike objects are to each other. However, it is quite common to calculate instead the *dissimilarity* between two patterns (which is just the opposite) using a distance metric defined on the feature space. Clearly, the choice of a distance metric is fundamental, since it has an impact on the properties of the final clusters, such as their size, quality, and consistency. Therefore, we review hereafter some important distance functions with their advantages and limitations. This enables us to have sound rationales later when we have to choose an appropriate distance function for a given attack feature F_k.

Some classical distances.

Probably one of the most commonly used distance measures is the Euclidean distance. For a pair of feature vectors[2] $\mathbf{x}_i$, $\mathbf{x}_j$ defined w.r.t. the same feature F_k, the Euclidean distance

[2]For the sake of clarity, we will omit the index (k), used to refer to a given feature F_k, since this dimension is not needed at this point of the discussion. In other words, we write $\mathbf{x}_i$, $\mathbf{x}_j$, instead of $\mathbf{x}_i^{(k)}$, $\mathbf{x}_j^{(k)}$, respectively.

can be calculated with:

$$\begin{aligned} d_2(\mathbf{x}_i, \mathbf{x}_j) &= (\textstyle\sum_{k=1}^{d}(x_{i,k} - x_{k,j})^2)^{\frac{1}{2}} \\ &= \|\mathbf{x}_i - \mathbf{x}_j\|_2 \end{aligned} \tag{3.1}$$

which is in fact a special case of the Minkowski metric (with $p = 2$):

$$\begin{aligned} d_p(\mathbf{x}_i, \mathbf{x}_j) &= (\textstyle\sum_{k=1}^{d}|x_{i,k} - x_{k,j}|^p)^{\frac{1}{p}} \\ &= \|\mathbf{x}_i - \mathbf{x}_j\|_p \end{aligned} \tag{3.2}$$

As observed in [88], Minkowski metrics work well when the data set contains compact or isolated clusters, but the drawback of these metrics is their sensitivity to the scale of the features. This problem can be alleviated with the normalization of the vectors. However, Euclidean distances suffer from other drawbacks, e.g., they can be completely inappropriate with high-dimensional data. This problem is known as the *curse of dimensionality* (the term was first coined by Richard Bellman in 1957 [18]), which is caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space. In fact, several previous works have showed that in high-dimensional space, the concept of proximity, distance or nearest neighbor may not even be qualitatively meaningful when relying on commonly used metrics such as L_k norms, especially in data mining applications [1].

Another common similarity measure that can be used with real-valued vectors is the *sample correlation* between observations treated as sequences of values:

$$d_{corr}(\mathbf{x}_i, \mathbf{x}_j) = \frac{(\mathbf{x}_i - \overline{\mathbf{x}})^T(\mathbf{x}_j - \overline{\mathbf{x}})}{\sqrt{(\mathbf{x}_i - \overline{\mathbf{x}})^T(\mathbf{x}_i - \overline{\mathbf{x}})}\sqrt{(\mathbf{x}_j - \overline{\mathbf{x}})^T(\mathbf{x}_j - \overline{\mathbf{x}})}} \tag{3.3}$$

where $\overline{\mathbf{x}}$ represents $(\mathbf{x}_i + \mathbf{x}_j)/2$.

The sample correlation (also called the *Pearson coefficient*) reflects the strength of the linear dependence between two real-valued vectors, which can also be viewed as a similarity degree between the "shapes" of the vectors. It is thus directly linked to the covariance of the vectors. A correlation value of 1 implies a perfect linear relationship between the two vectors (as x_i increases, x_j increases proportionally). A closely related similarity measure is the *cosine similarity* obtained by computing the cosine of the angle formed by the two vectors, which is commonly used for clustering document data in text mining [160].

The interpretation of a correlation coefficient depends on the context and purposes; however, a value between 0.5 and 1 is usually considered as an indication of a strong dependence between observations.

Dissimilarity measures for probability distributions.

When we have to deal with observations that are in the form of probability distributions (e.g., histograms), like in the illustration given in Fig. 3.2, then statistical distances seem more appropriate to measure pairwise distances. One such technique (which is commonly used in information theory) is the Kullback-Leibler divergence ([78]). Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be

for instance two feature vectors that represent two probability distributions over a discrete space X, then the K-L divergence of $\mathbf{x}_j$ from $\mathbf{x}_i$ is defined as:

$$D_{KL}(\mathbf{x}_i||\mathbf{x}_j) = \sum_{k=1}^{d} \mathbf{x}_i(k) \log \frac{\mathbf{x}_i(k)}{\mathbf{x}_j(k)}$$

which is also called the information divergence (or *relative entropy*). Because D_{KL} is not considered as a true metric, it is usually better to use instead the Jensen-Shannon divergence ([87]), defined as:

$$D_{JS}(\mathbf{x}_i, \mathbf{x}_j) = \frac{D_{KL}(\mathbf{x}_i||\bar{\mathbf{x}}) + D_{KL}(\mathbf{x}_j||\bar{\mathbf{x}})}{2} \tag{3.4}$$

where $\bar{\mathbf{x}} = (\mathbf{x}_i + \mathbf{x}_j)/2$. In other words, the Jensen-Shannon divergence is the *average* of the KL-divergences to the *average distribution*. To be a true metric, the JS divergence must also satisfy the triangular inequality, which is not true for all cases of $(\mathbf{x}_i, \mathbf{x}_j)$. Nevertheless, it can be demonstrated that the *square root* of the Jensen-Shannon divergence is a true metric ([55]).

An alternative metric for measuring the similarity of two discrete probability distributions is the *Bhattacharyya* distance ([19]), which is mostly used by the signal processing community. It gives an approximate measurement of the amount of overlap between two frequency vectors. For two probability distributions $\mathbf{x}_i$ and $\mathbf{x}_j$ over the same domain X, it is defined as:

$$D_{BC}(\mathbf{x}_i, \mathbf{x}_j) = -ln(BC(\mathbf{x}_i, \mathbf{x}_j)) \tag{3.5}$$

where

$$BC(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} \sqrt{\mathbf{x}_i(k)\mathbf{x}_j(k)}$$

While D_{BC} does not satisfy the triangle inequality, the Hellinger distance $\sqrt{1 - BC}$ does obey the triangle inequality, and can thus be used as metric in a clustering algorithm.

Yet other statistical metrics have been recently developed for measuring dissimilarities between probability distributions in a consistent manner, for instance the *Earth Mover's Distance* (EMD) [132], which derives from the Wasserstein metric [133]. The EMD provides also a measurement of the distance between two distributions over some region D. Informally, if the distributions are interpreted as two different ways of piling up a certain amount of dirt over the region D, the EMD is the minimum cost of turning one pile into the other (where the cost is assumed to be the amount of dirt moved times the distance by which it is moved).

Transforming distances to similarities.

To transform pairwise distances d_{ij} to similarity weights s_{ij}, we can use different mapping functions. Some of the most commonly used functions are:

$$s_{ij} = \begin{cases} 1 - d_{ij} \\ c - d_{ij}, \text{ for some constant } c \\ (1 + d_{ij})^{-1} \end{cases}$$

However, previous studies found that the similarity, or the confusion frequency, between stimuli decay exponentially with some power of the perceptual measure distance ([143]). So, it is also customary to use the following functional form to do this transformation:

$$s_{ij} = exp(\frac{-{d_{ij}}^2}{\sigma^2}) \tag{3.6}$$

where σ is a positive real number which affects the decreasing rate of s. In Chapter 5, we propose an empirical method for determining appropriate ranges of values for σ according to the statistical properties of the data set being analyzed, and the expected output similarities.

Several authors have extensively studied problems related to proximity measures, and how to choose them in a consistent manner based on the clustering problem and the features at hand (see for instance [49, 13, 74]). To underline the importance of such a choice, we illustrate the application of the presented distances on some feature vectors obtained from an attack data set.

Illustrative example.

Let us consider again the same example introduced in Section 3.2 (feature selection), i.e., a data set made of computer attacks observed in the Internet, which manifest themselves under the form of *attack events* that comprise groups of malicious sources targeting in a coordinated fashion other machines. Again, we can record for each source its geographical origin, which leads to the creation of *spatial distributions* for each observed event of the data set. What we are interested in, is to find a way to measure how similar those distributions are with each other, in order to infer which attack events may originate from the very same geographical areas (which could *eventually* mean that those events are due to the same attack phenomenon).

So, let us consider four different events, for which we provide in Table 3.1 the geographical distribution (i.e., the feature vectors for F_{geo}). These distributions are illustrated in Fig 3.5, where we represent the relative frequencies only for the countries mostly involved in the attacks, i.e., those lying in the upper quantile.

Table 3.1: Geographical distributions of four different events from an attack data set.

Feat. vector	Geo. distribution
$\mathbf{x}_1$	CA(25%), CN(22%), US(15%), IT(7%), FR(6%), others(25%)
$\mathbf{x}_2$	CN(28%), CA(23%), US(13%), IT(3%), FR(6%), others(27%)
$\mathbf{x}_3$	CN(48%), CA(12%), US(9%), FR(5%), IT(2%), others(25%)
$\mathbf{x}_4$	US(20%), FR(15%), CA(13%), TW(11%), IT(9%), JP(6%), others(26%)

Intuitively, from the shapes of the distributions and the relative frequencies given in Table 3.1, we observe that $\mathbf{x}_1$ and $\mathbf{x}_2$ are highly similar, $\mathbf{x}_1$ is somehow correlated to $\mathbf{x}_3$ (but there are several differences between the two), and $\mathbf{x}_1$ and $\mathbf{x}_4$ should be dissociated from each other because there are too many significant differences. In Table 3.2 we have computed the similarities using the Euclidean distance, Jensen-Shannon (JS), Bhattacharyya (BC), and the sample correlation. The distances were mapped to similarities using equa-

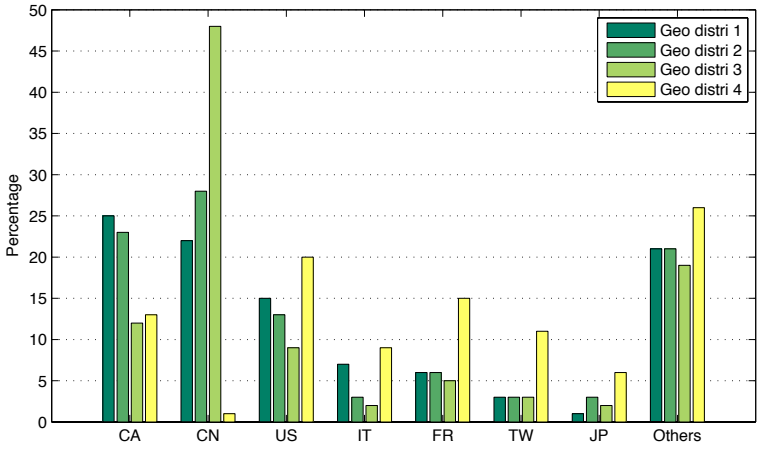


Figure 3.5: The geographical distributions of four different events. The Y-axis represents the relative proportion of attackers coming from each respective country given on X.

Table 3.2: Pairwise similarities for the distributions given in Table 3.1.

	Eucl.	Corr.	JS	BC	χ^2
$(\mathbf{x}_1, \mathbf{x}_2)$	0.85	0.97	**0.88**	0.72	1
$(\mathbf{x}_1, \mathbf{x}_3)$	0.34	0.23	**0.62**	0.56	0.62
$(\mathbf{x}_1, \mathbf{x}_4)$	0.16	0.01	**0.14**	0.28	0.04

tion 3.3.2, and with appropriate values for σ that were derived from the properties of the data set.

To validate our intuition about the relationships among those distributions, we have also calculated the *p-value* obtained with a χ^2 statistical test, which is one of the most commonly used methods for determining whether two underlying one-dimensional probability distributions differ in a significant way. The test of χ^2 confirms the strong dependence between $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, and reveals a (statistically) significant difference with $\mathbf{x}_4$ (since a p-value lower than 0.05 means that we can safely reject the hypothesis of a dependency between two distributions). As a result, we observe that certain distance metrics perform better than others. More precisely, the Euclidean distance and the sample correlation tend to underestimate the degree of similarity for $(\mathbf{x}_1, \mathbf{x}_3)$, whereas Bhattacharyya overestimates the relation for $(\mathbf{x}_1, \mathbf{x}_4)$. In this example, Jensen-Shannon seems to provide the best metric for measuring similarities between statistical distributions present in our data sets. This was confirmed by extensive experiments with large amounts of data as well as with other types of distributions from our data sets.

3.3.3 Graph-based clustering

3.3.3.1 Problem formalization

We can now turn to the description of the clustering component of our method, which implements a pairwise clustering technique. We formulate the problem using a graph-based approach that is inspired by the approach developed by Pouget in [120]. However, we have further extended and largely improved this graph-based approach (cf. the explanation on the positioning of this work in Chapter 2, page 25).

For each attack feature F_k, we build an edge-weighted graph G_k in which the vertices (or nodes) are mapped to the feature vectors $\mathbf{x}_i^{(k)}$, and the edges (or links) reflect the similarity between data objects regarding the considered feature. As customary, we can represent the undirected edge-weighted graph (with no self-loops) obtained for a given feature F_k by

$$G_k = (V_k, E_k, \omega_k)$$

$$\text{where} \begin{cases} V_k = \{\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}, \ldots, \mathbf{x}_m^{(k)}\} & \text{is the vertex set} \\ E_k \subseteq V_k \times V_k & \text{is the edge set (i.e., relations among vertices)} \\ \omega_k : E_k \rightarrow \Re^+ & \text{is a positive weight function} \end{cases}$$

In practice, we can represent each graph G_k with its corresponding weighted *adjacency matrix* (or dissimilarity matrix), which is the $m \times m$ symmetric matrix $A_k(i,j)$ defined as:

$$A_k(i,j) = \begin{cases} \omega_k(i,j), & \forall (i,j) \in E_k \\ 0, & \text{otherwise.} \end{cases}$$

Note that the weight function $\omega_k(i,j)$ must be defined with a similarity metric that is appropriate to the nature of the feature vector under consideration, as we have explained in Section 3.3.2.

In general, graph-theoretic clustering algorithms consist of searching for certain combinatorial structures in the similarity graph, such as a minimum spanning tree ([199]) or a minimum cut ([144, 190]). Among these methods, a classic approach to clustering reduces to a search for complete subgraphs, which is also known as the "complete-link" algorithm. Indeed, the maximal complete subgraph, also called a *(maximal) clique*, was considered the strictest definition of a cluster in [4] and [125].

The concept of a maximal clique was originally defined on unweighted graphs; however, it has been recently generalized to the edge-weighted case by Pavan et al. [116] who proposed a new framework for pairwise clustering based on *dominant sets*. The formal properties of dominant sets make them reasonable candidates for a new formal definition of a cluster in the context of edge-weighted graphs. Furthermore, Pavan et al. [116] established a correspondence between dominant sets and the extrema of a continuous quadratic form over the standard simplex. This means that we can find dominant sets (or clusters) using straightforward continuous optimization techniques such as *replicator equations*, a class of dynamical systems arising in evolutionary game theory. Such systems are interesting since they can be coded in a few lines of any high-level programming language.

In the next paragraphs, we formally define the notion of *dominant set*, as used in graph-theoretical clustering. Then, we present the dominant set algorithm introduced by Pavan et al. [116]. Finally, we apply the dominant set framework on a real data set of

attack events to illustrate the use of this technique in the context of our attack attribution method.

3.3.3.2 Graph-theoretical definition of a cluster

Informally, a cluster should satisfy two fundamental conditions: (a) it should have high internal homogeneity; (b) there should be high inhomogeneity between the objects of the cluster and those outside. Going back to our edge-weighted graph representation, these conditions are equivalent to saying that the weights on the edges within a cluster should be large, whereas those on the edges connecting the cluster nodes to the external ones should be small. To quantify these notions, let us consider a graph $G = (V, E, \omega)$ for a given feature F_k, and its corresponding similarity matrix $A = (a_{ij})$. For convenience, and without loss of generality, we can safely ignore for now the index k referring to the feature, since following notions are valid for any characteristic of the data set used as clustering feature.

Let $S \subseteq V$ be a non-empty subset of vertices and $i \in V$ be a certain vertex. The *average weighted degree* of i w.r.t. S is defined as:

$$\text{awdeg}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij}$$

Observe that $\text{awdeg}_{\{i\}}(i) = 0$ for any $i \in V$. Furthermore, if $j \notin S$ we define:

$$\phi_S(i, j) = a_{ij} - \text{awdeg}_S(i)$$

Note that $\phi_{\{i\}}(i, j) = a_{ij}$, for all $i, j \in V$. Intuitively, $\phi_S(i, j)$ measures the similarity between nodes j and i, with respect to the average similarity between node i and its neighbors in S. Note that $\phi_S(i, j)$ can be either positive, negative or null.

We are now able to formalize the notion of *induction* of node-weights, which is expressed by the following recursive definition.

Definition 3.1. (Node-weights induction [116]) *Let $S \subseteq V$ be a non-empty subset of vertices and $i \in S$. The weight of i w.r.t. S is*

$$\text{w}_S(i) = \begin{cases} 1, & if\ |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j, i) \text{w}_{S \setminus \{i\}}(j), & otherwise. \end{cases}$$

Additionally, the total weight of S is defined as:

$$\text{W}(S) = \sum_{i \in S} \text{w}_S(i)$$

Note that $\text{w}_{\{i,j\}}(i) = \text{w}_{\{i,j\}}(j) = a_{ij}$, for all $i, j \in V$ $(i \neq j)$. It is also worth noting that $\text{w}_S(i)$ is calculated simply as a function of the weights on the edges of the subgraph induced by S. That is, $\text{w}_S(i)$ gives us a measure of the overall similarity between vertex i and the vertices of $S \setminus \{i\}$ w.r.t. the overall similarity among the vertices in $S \setminus \{i\}$.

We are now in the position of formally defining the concept of cluster (or dominant set) in an edge-weighted graph.

Definition 3.2. (Dominant set) [116] *A non-empty subset of vertices $S \subseteq V$ such that* $\mathrm{W}(T) > 0$ *for any non-empty $T \subseteq S$, is said to be* dominant *if:*

1. $\mathrm{w}_S(i) > 0$, for all $i \in S$, and
2. $\mathrm{w}_{S \cup \{i\}}(i) < 0$, for all $i \notin S$.

The two conditions here above correspond to the two main properties of a cluster: condition (1) implies that a dominant set should have high internal homogeneity, whereas (2) imposes the external inhomogeneity.

3.3.3.3 Clustering using dominant sets

Considering definition 3.2, the clustering algorithm introduced by Pavan et al. consists basically of iteratively finding a dominant set in an edge-weighted graph, and then removing it from the graph until all vertices have been clustered (complete partitioning), or as soon as a given *stopping criterion* is met, which could give eventually an incomplete partition as output. Some examples of constraints we can set as stopping criterion are: (i) a minimum threshold for the remaining nodes within the graph; (ii) a lower threshold (absolute or relative) on the sum of all remaining edge-weights. Thus, let $W_{origin} = \sum a_{ij}$ be the sum of all weights in the original graph, then the procedure could stop when the sum of all remaining edge-weights is less than $(0.01 \cdot W_{origin})$. The clustering algorithm is described in the pseudo-code given in algorithm 3.1. As one can see, the cornerstone of this algorithm is the procedure DOMINANT_SET, which remains to be defined.

In [116] it was proved that there is a tight correspondence between the problem of finding dominant sets in an edge-weighted graph (i.e., finding maximum weighted cliques) and the problem of finding the extrema of a (continuous) quadratic form over the standard simplex. Computationally, this allows us to find dominant sets (i.e., clusters) using straightforward continuous optimization techniques such as *replicator equations*, a class of dynamical systems arising in evolutionary game theory. Note that these systems are also intimately related to Hopfield neural networks ([69]).

Algorithm 3.1 Dominant sets Clustering

Input: weighted graph $G = (V, E, \omega)$
Output: a partition $\mathcal{P}$ (eventually incomplete)

```
P = ∅
while STOPPING_CRITERION(G) do
    S ← DOMINANT_SET(G)
    P ← P ∪ {S}
    V ← V \ S
return P
```

As a result, we can find dominant sets by simply making a particular temporal expression converge. More precisely, consider the following dynamical system represented with

its discrete time equation, where A_k is the adjacency matrix of an edge-weighted graph G_k:

$$x_i(t+1) = x_i(t) \cdot \frac{(A_k\,\mathbf{x}(t))_i}{\mathbf{x}(t)^T A_k\,\mathbf{x}(t)} \tag{3.7}$$

with $i = 1, \ldots, m$. Starting from an arbitrary initial state, this dynamical system will eventually be attracted by the nearest asymptotically stable point. Thus, the procedure DOMINANT_SET simply involves the simulation of the system given in equation 3.7. The solution to this dynamical system is a stable point, called a characteristic vector $\mathbf{x}^S$, which satisfies the conditions of definition 3.2, and thus corresponds to a dominant set (as it has been proved in [115]).

Finally, the assignment of weights to all graph edges gives us a natural measure of the overall similarity of a dominant set. Given a dominant set $S \subseteq V$, we measure its overall cohesiveness with:

$$\mathrm{Coh}(S) = \frac{\sum_{i \in S} \mathrm{awdeg}_S(i)\mathrm{w}_S(i)}{\mathrm{W}(S)} \tag{3.8}$$

Observe that $\max(\mathrm{Coh}(S))$ equals $1 - \frac{1}{|S|}$, and thus the maximal cohesiveness of a dominant set S tends to 1 only for very large clusters. For small clusters, $max(\mathrm{Coh}(S))$ approaches 0.5 (which is achieved for a dominant set of 2 vertices).

3.3.3.4 A short example.

To illustrate the concepts defined here above, let us consider the weighted graph given in Fig. 3.6, which comprises twelve vertices ($m = 12$). As one can see, there are two main groups of nodes that are weakly interconnected by the edge $(8, 9)$. All edges represented with dashed lines have smaller weights (i.e., less than 0.2), while edges in solid lines have weights between 0.5 and 1. By applying algorithm 3.1, we find iteratively three dominant sets (DM), which are given in Table 3.3, along with their characteristics. Interestingly, the algorithm first finds DM1=$\{2, 3, 4, 5, 6\}$, thus it focuses on the largest DM with a high cohesiveness (i.e., the most significant group). Then, even though DM2 has slightly larger values for the cohesiveness and the total weight, this cluster is found at the second iteration because it contains less members.

Finally, the algorithm recovers the last group $\{1, 7, 8\}$. That is, it can separate this set of nodes from those of DM1, even if there are many "weak relations" among members of the two clusters (e.g., edges with weight values between 0.05 and 0.20). Note that we didn't specify any predetermined number of clusters, and that all nodes have been naturally (and correctly) clustered, even with a standard stopping criterion on the remaining weights.

Table 3.3: Characteristics of the dominant sets extracted from the graph in Fig. 3.6.

	Vertices	Cohesiveness ($\mathrm{Coh}(S)$)	Total weights ($\mathrm{W}(S)$)	$\mathrm{awdeg}_S(i)$
DM1	$\{2, 3, 4, 5, 6\}$	0.70	2.52	$\{0.64, 0.73, 0.72, 0.72, 0.65\}$
DM2	$\{9, 10, 11, 12\}$	0.75	4	$\{0.75, 0.75, 0.75, 0.75\}$
DM3	$\{1, 7, 8\}$	0.52	1.88	$\{0.53, 0.54, 0.51\}$

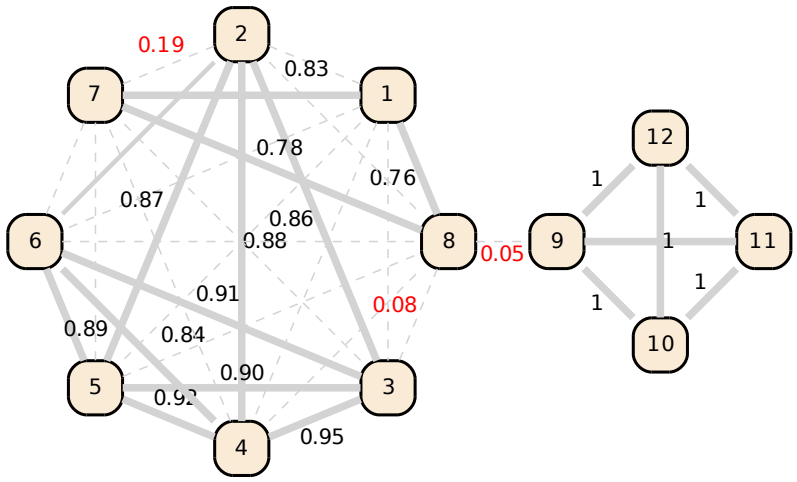


Figure 3.6: An example of edge-weighted graph in which 3 dominant sets are found iteratively by the algorithm : (1) $\{2, 3, 4, 5, 6\}$; (2) $\{9, 10, 11, 12\}$; (3) $\{1, 7, 8\}$.

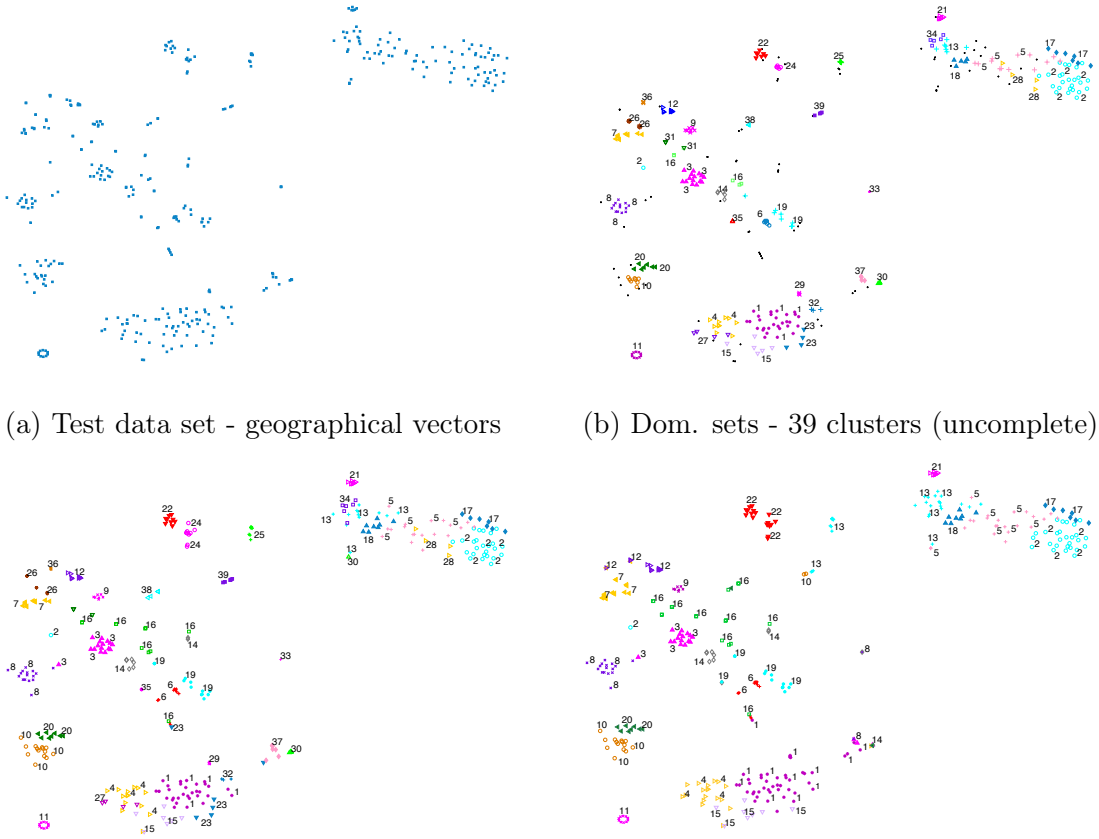
3.3.3.5 Illustration on a real data set.

To further illustrate the use of dominant sets clustering in the context of our attack attribution method, we have applied the DM framework on a real-world data set made of computer attacks. As introduced earlier in Sections 3.2 and 3.3.2, we have observed malicious sources attacking different sensors in the Internet, and we have recorded for each source its geographical location. Certain sources seem to coordinate their actions, so it leads to the observation of so-called *attack events* on the sensors, which are limited in time and comprise malicious sources attacking in a coordinated fashion.

For this illustration, we consider 351 attack events (i.e., $m = 351$) observed in a time period spanning from September 2006 until June 2008, and for which we have computed their respective feature vector to produce $\mathbf{x}_i^{geo} = \{x_{i,1}^{(geo)}, \ldots, x_{i,229}^{(geo)}\}$, for $i = 1, \ldots, 351$. Now, we want to discover whether this data set contains some meaningful clusters of events that do share very strong similarities with respect to their spatial distributions, which could help the analyst to figure out if those events could eventually be linked to a same phenomenon. For this purpose, we just need to build the edge-weighted graph by using an appropriate similarity metric (e.g., Jensen-Shannon in this case), and then to feed the adjacency matrix of the graph to algorithm 3.1.

Fig 3.7 (a) shows a 2D mapping of the data set, where we have used a non-linear dimensionality reduction technique called *t-Distributed Stochastic Neighbor Embedding* (or t-SNE [175]) to embed the high-dimensional feature vectors into a 2-dimensional plot. The aim of dimensionality reduction is to preserve as much of the significant structure of the high-dimensional data as possible in the low-dimensional map. That is, we can verify that two nearby data points have highly similar feature vectors, whereas two distant points should have nothing in common in their respective distributions. This can be helpful to visualize a high-dimensional data set, but also to assess the consistency of clustering

results. So, it is important to keep in mind that each data point on such 2D representations is mapped to a d-dimensional feature vector, like a geographical distribution in this case.



(a) Test data set - geographical vectors

(b) Dom. sets - 39 clusters (uncomplete)

(c) Dom. sets - 39 clusters (complete)

(d) Dom. sets - 22 clusters (complete)

Figure 3.7: Clustering results obtained by applying *dominant sets* on a real data set made of geographical distributions of machines involved in computer attacks in the Internet. The different clusters are represented by a combination of a marker type and a color.

Dimensionality reduction techniques transform a data set $\mathbf{X}$ of dimensionality d into a new data set $\mathbf{Y}$ of dimensionality d^*, with $d^* \ll d$, while retaining the underlying structure of the data set $\mathbf{X}$ as much as possible. *t-SNE* is a non-linear technique that comes from a variation of Stochastic Neighbour Embedding [68]; however, it produces significantly better visualizations than other multidimensional scaling techniques (such as Sammon mapping, Isomaps or Laplacian Eigenmaps) by reducing the tendency to crowd points together in the centre of the map [175].

Fig. 3.7 (b) shows the results of dominant sets clustering on the very same data set, where we used a stopping criterion of 0.01. 39 clusters were obtained, and those clusters account for 279 events (80% of the data set), which means that the output of this clustering is an *uncomplete* partition. That is, the unclustered nodes (depicted by black dots on Fig. 3.7 (b)) have an unsufficient number of meaningful relations (or edges) to be included in the largest and most significant clusters found so far.

Table 3.4: Centroïds of different clusters depicted in Fig. 3.7 (b).

	DM centroïds (average distributions)
DM22	HU(13%), PL(17%), FR(10%), BR(8%)
DM24	PL(39%), DE(19%), ES(6%)
DM30	TW(42%), CN(13%), ES(6%)
DM37	TW(14%), KR(42%), CN(17%)
DM1	CN(56%), CA(8%), US(8%)
DM4	CA(17%), CN(29%), US(15%), FR(7%)
DM15	CN(38%), US(16%), FR(12%)
DM23	CN(54%)
DM32	CN(75%)

Assuming that the analyst wants all feature vectors to be clustered (*complete partitioning*), then we can easily adapt the clustering algorithm so as to assign each unclustered vector to the closest group. This result is illustrated in Fig 3.7 (c); however, we must be aware of the fact that certain vectors may then be misclassified, i.e., they are assigned to clusters in which they do not share strong similarities with the other members.

Another interesting observation is that the dominant sets clustering is able to extract the most significant and informative groups in the early stages of the clustering. That is, if the analyst sets a higher threshold as stopping criterion (e.g., 0.05), then he gets obviously less clusters; however, those clusters are the same as the first ones found with a lower threshold. This is illustrated in Fig 3.7 (d), where we can see the 22 clusters obtained with a stopping criterion of 0.05. As we can see, those 22 clusters are exactly the same as the first 22 clusters obtained with a stopping criterion of 0.01. The dominant sets 23 to 39 have thus been absorbed by the previous ones, for example: DM28 has been absorbed by DM2 and DM5, DM34 has been absorbed by DM13, DM27 by DM4, etc.

Finally, from a semantic viewpoint, we note that dominant sets reflect very strong relationships among their respective members. Even with a data set having a quite complex underlying structure (as the 2D mapping seems to suggest), the DM algorithm is able to separate groups of attack events that have some commonalities in their geographical distributions, but have also some (statistically) significant differences which are worth being underlined and showed to the analyst. To illustrate this point, Table 3.4 provides the centroïds of different clusters depicted in Fig 3.7 (b). As one can see, there are some notable differences even between adjacent clusters, such as DM22 vs DM24, DM30 vs DM37, or among DM 1, 4, 15, 23 and 32. As we demonstrate in the next Section, this kind of results is difficult to achieve with more classical clustering techniques.

3.4 Evaluation of clustering results

In the previous Section, we have intuitively showed that clustering by dominant sets provides naturally meaningful and informative groups, even when applied on a real data set characterized by a complex structure. However, it is important to assess these results by means of more objective criteria, so as to validate the soundness of this clustering technique. There are mainly two approaches to perform this validation. An *external* assessment compares the recovered structure to an a priori structure, which is not feasible in this case,

since we do not have any prior information on the true labels of the data points. On the other hand, we can perform an *internal* evaluation of the clusters validity to confirm that the obtained results cannot reasonably have occurred as an artefact of the clustering algorithm.

In this Section, we will first apply two different techniques that try to estimate the "correct" number of groups, which is mostly useful for clustering techniques that require this number as input, but it can also be used to verify if the number of clusters found by a given technique is consistent with this estimation. Then, we define and apply three different internal validity indices on the partitioning results obtained in previous Section. Finally, based on those objective indices, we compare those previous results against those obtained with two other commonly used clustering approaches, namely Hierarchical Clustering and K-Means.

3.4.1 Estimating the optimal number of clusters

We consider again our test data set introduced in previous Sections, but we are now turning our attention to the estimation of the optimal number of clusters. The dominant sets clustering provided us 39 (meaningful) clusters; however, how can we cross-validate this result?

Upper tail rule.

First, we can apply a technique developed by Mojena in [99], which uses a hierarchy of partitions to estimate the number of clusters. In Hierarchical Clustering (HC), we rely also on a pre-calculated proximity matrix (just like in graph-based clustering), but the clustering process consists of a sequence of steps where data objects and clusters are iteratively merged according to some optimality criterion (i.e., an *agglomerative* process). The *linkage* method can be done in different ways, e.g.: (i) based on the smallest distance between objects (*single linkage*, a.k.a. nearest neighbor), (ii) based on the largest distance (*complete linkage*, a.k.a. furthest neighbor), or (iii) based on the average distance from all points in one cluster to all points in another cluster. The result of HC is a nested set of partitions (i.e., a hierarchy), on which the analyst still has to define an appropriate *fusion level* that provides the final set of clusters (see [49] for more information on this clustering technique).

Mojena [99] proposed a method known as the *upper tail rule* as a way of determining the appropriate number of groups in hierarchical clustering. It is based on the relative sizes of the different fusion levels in the hierarchy. Basically, the Mojena rule consists in plotting the standardized fusion levels as a function of the desired number of clusters. To calculate those standardized fusion levels, we can use the following equation [91]:

$$\frac{(\alpha_{j+1} - \overline{\alpha})}{\sigma_\alpha} \tag{3.9}$$

where α_{j+1} is the fusion level corresponding to a set of $m - (j + 1)$ clusters, i.e., α_0 is the fusion level for m clusters (recall that m is the number of objects in the data set), and thus α_{m-1} is the fusion level corresponding to 1 cluster only. Then, $\overline{\alpha}$ is the average of the j previous fusion levels, and σ_α is the standard deviation of the j previous levels.

So, we have applied this technique on our test data set introduced in Section 3.3.3. The Mojena plot for the three common linkage methods (single, complete and average linkage) is represented in Fig. 3.8. Normally, an elbow in the curve should indicate a reasonable estimate of the number of clusters. As one can see, the single linkage method indicates that there should be at least 5 clusters. However, single linkage suffers from the "chaining effect". This can affect the results by forming elongated clusters in which objects are only weakly connected by some sort of chain between a minority of data points.

Looking at the results of complete and average linkage methods, we can conclude that there should be at least 5 to 10 clusters in the data set according to this evaluation. However, there are no clear break in those curves, and the standardized fusion levels become almost flat only after 40 or 50 clusters. In conclusion, the Mojena method is not very precise on this data set and it provides here only a rough estimation of the number of clusters, which lies apparently between 5 and 50 according to this technique.

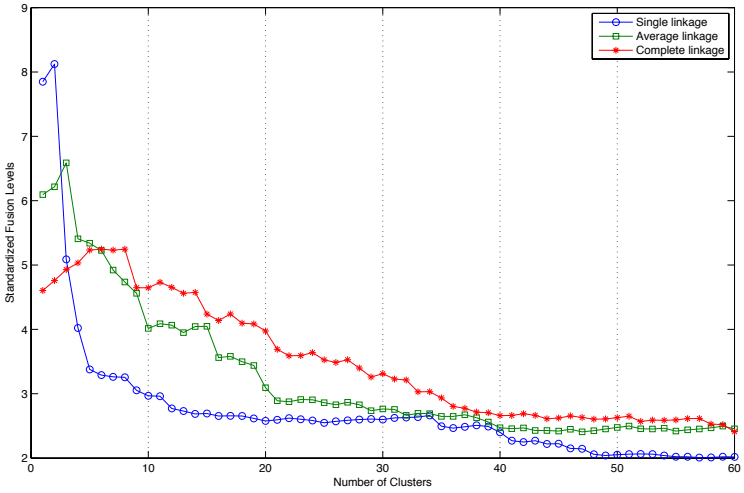


Figure 3.8: Mojena plot of the standardized fusion levels in hierarchical clustering, for three different linkage methods. An elbow in the curves indicates a good candidate for the estimation of the number of clusters.

Gap statistic.

Tibshirani et al. have proposed a method called the "Gap statistic" for estimating the number of clusters in a data set [169]. This technique can be applied to the output of any clustering algorithm. The idea is to compare the change in *within-cluster dispersion* to that expected under an appropriate null distribution used as reference.

Let us assume we have obtained k clusters $C_1, \ldots, C_k$ as output of a given clustering method, and the r^{th} cluster has N_r members. Then, we calculate the sum of all pairwise distances for that cluster C_r with:

$$D_r = \sum_{i,j \in C_r} d_{ij}$$

We define now W_k as

$$W_k = \sum_{r=1}^{k} \frac{1}{2N_r} D_r \tag{3.10}$$

When the distance used is the squared Euclidean distance, then W_k represents the pooled within-cluster sum of squares around the cluster means.

The idea of the gap statistic is to compare the values of the within-cluster dispersions $\log(W_k)$, $k = 1, \ldots, K$, with the expected values under a reference *null distribution* (which should give a priori only 1 cluster). Basically, the gap statistic procedure involves the simulation of B data sets according to a null distribution (e.g., a uniform distribution), and the clustering method is applied to each of them. The same dispersion index $\log(W_k^*)$ can then be calculated for each simulated data set. The gap statistic is calculated using the following equation:

$$Gap(k) = \frac{1}{B} \sum_{b} \log(W_{k,b}^*) - \log(W_k) \tag{3.11}$$

The estimation of the optimal number of clusters is then the value of k for which the observed value $\log(W_k)$ falls the farthest below the expected value $\log(W_k^*)$ obtained under a null distribution. That is, our estimate $\hat{k}$ is in fact the value maximizing $Gap(k)$.

Note that Tibshirani et al. suggest two strategies to generate the reference null distribution [169]: (i) a uniform distribution over the range of observed values for each variable of the data set (i.e., we generate d one-dimensional column-vectors with values that are uniformly distributed over the range $x_{j,min}$ and $x_{j,max}$ for each column j of the data X); (ii) a uniform distribution over a box aligned with the *principal components* of the data.

In Fig. 3.9 we have represented the Gap statistic as a function of the number of clusters by using the dominant sets algorithm on the same data set of attack events used previously in this Chapter. Again, we used Jensen-Shannon as distance metric. The stopping criterion of algorithm 3.1 corresponding to the desired number of clusters is indicated on the second x-axis on top of the plot.

As we can see on the graph, the maximum value of $Gap(k)$ is reached at 59 clusters, which is consistent with the results obtained before. However, it seems that a stopping criterion of 0.01 gives meaningful results in terms of clusters. In fact, decreasing even more the stopping criterion simply generates a large number of small clusters (of size 2 or 3), while providing only small increments on the gap statistic values (which are not really significant).

Observe also that there is a series of local optima in the ranges between $15 - 21$ and $25 - 29$ clusters, which could also be considered as reasonable estimates in this case. In fact, the simulations performed in [169] suggest that the Gap statistic is good at identifying *well-separated* clusters. When data points are not well separated, it was showed that the Gap method becomes as inaccurate as the proportion of overlapping points, which is probably the reason why this method is not really able to provide a clear estimation for this data set. However, since the data is very different from a null distribution, it gives us a reasonable estimate of the *range of values* for the number of clusters present in the data, and it allows us to calibrate the stopping criterion of the dominant sets algorithm. We observe that the gap estimate is consistent with the results obtained by using dominant sets, which demonstrates the meaningfulness of this graph-based approach when applied

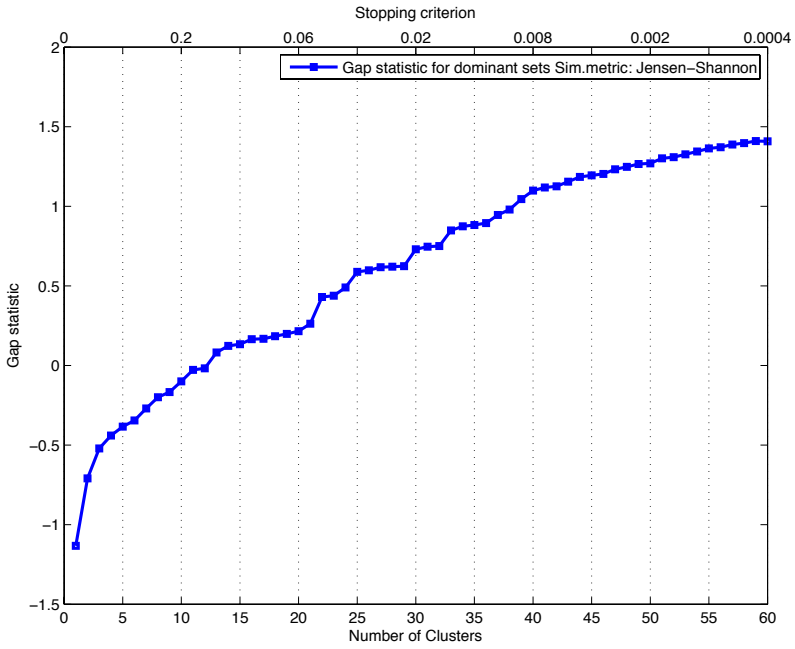


Figure 3.9: Gap statistic in function of the number of clusters in dominant sets clustering. The corresponding value of the stopping criterion is indicated on the second x-axis on top of the plot.

to this example of real-world data set.

3.4.2 Cluster validity indices

We will conclude the evaluation of our clustering results by using some objective indices. Various cluster validity indices have been proposed in the past to assess the quality and the consistency of clustering results ([67]). In graph clustering, most indices are based on the comparison of intra-cluster connectivity (i.e., the compactness of clusters) and the inter-cluster variability (i.e., the separability between clusters). Boutin and Hascoët [20] provide a good review of validity indices that are especially appropriate for assessing graph clustered structures.

In this Section, we define some validity indices that are, in our opinion, well-suited for evaluating the quality of our dominant sets, but also the results obtained via other clustering methods. More precisely, we will use three different validity indices:

- the Graph compactness, which indicates how compact the clusters are;
- the Davies-Bouldin index, which evaluates the inter versus intra-cluster connectivity;
- the Silhouette index, which is linked to the characteristics of nodes' neighborhood.

Graph compactness

The *graph compactness* C_p is a validity index that is very easy to calculate, and which can be helpful to evaluate graph clustering. C_p is mainly based on the characteristics of graphs

connectivity. That is, for any cluster C_k, we can calculate a normalized compactness index, as proposed in [20]:

$$\mathrm{C}_{p_k} = \frac{\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k}\omega(i,j)}{N_k(N_k-1)/2} \tag{3.12}$$

where $\omega(i,j)$ is the positive weight function reflecting node similarities in the graph, and N_k is the number of members within cluster k. Since C_{p_k} only depends upon similarity values and the composition of the clusters, it can be used to evaluate any other clustering method.

We can also define a mean compactness index $\overline{\mathrm{C}_p}$ for a partition made of K clusters, which takes into account the individual compactness values of the clusters, but also their respective sizes:

$$\overline{\mathrm{C}_p} = \frac{\sum_{k=1}^{K}\mathrm{C}_{p_k}N_k}{\sum_{j=1}^{K}N_j} \tag{3.13}$$

Davies-Bouldin index

Davies and Bouldin have proposed an index to evaluate the quality of a graph partition that aims at comparing the inter and intra-cluster connectivity. It is defined as [42]:

$$\mathrm{DB} = \frac{1}{K}\sum_{i=1}^{K}\max_{j\neq i}\left(\frac{\mathrm{diam}(C_i)+\mathrm{diam}(C_j)}{d(C_i,C_j)}\right)$$

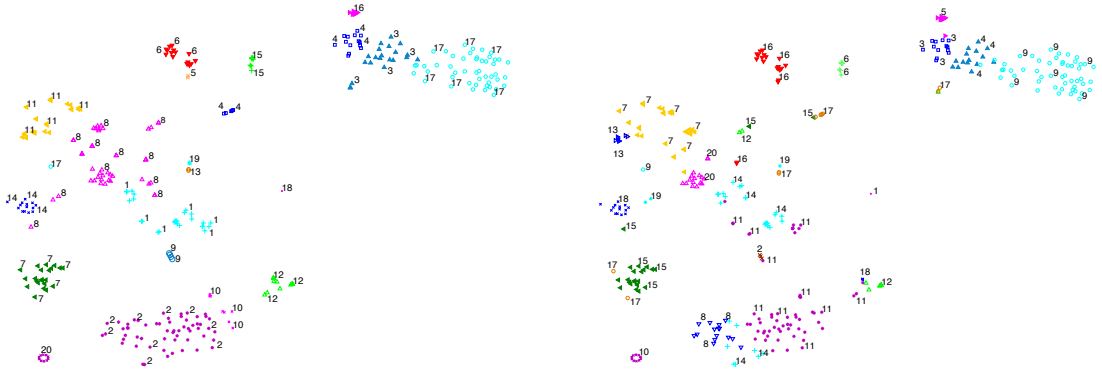
where K is the number of clusters in the partition to be evaluated, $\mathrm{diam}(C_i)$ is the diameter of cluster C_i, and $d(C_i,C_j)$ is the inter-cluster distance between C_i and C_j (e.g., the average distance between two nodes, with one in each cluster, or the distance between clusters centroïds). Small values of DB correspond to compact clusters. Note that Davies-Bouldin index is more robust than Dunn's index, and can be used for the evaluation of any clustering method (not only graph-based).

Silhouette index

The silhouette index is different from the two other indices defined here above, since it is based on node's neighborhood (rather than on a global compactness index). Let us consider a node (i.e., a point of the data set) $\mathbf{x}_i$ that belongs to cluster C_j. Suppose that the closest cluster to node $\mathbf{x}_i$ (according to the average distance) is denoted by C_h. The silhouette index is defined by ([131, 76]):

$$\mathrm{silh}(\mathbf{x}_i) = \frac{d(\mathbf{x}_i,C_h)-d(\mathbf{x}_i,C_j)}{\max(d(\mathbf{x}_i,C_j),d(\mathbf{x}_i,C_h))}$$

Observe that $-1 \leq \mathrm{silh}(\mathbf{x}_i) \leq 1$. Data points with large silhouettes are well clustered, whereas those with small values are likely to be scattered between clusters. Points with negative silhouette values are not well-clustered. Rousseeuw proposed to plot clusters silhouettes on a chart as a graphical aid to estimate the number of clusters in the data [131].



(a) (b)

Figure 3.10: Visualization of clustering results obtained with (a) Hierarchical Clustering and (b) K-Means, applied to the same data set of Fig. 3.7. The different clusters are represented by a combination of a marker type and a color.

For a given cluster C_j we compute its average silhouette S_j as follows:

$$S_j = \frac{1}{N_j} \sum_{i \in C_j} \text{silh}(\mathbf{x}_i)$$

Then, similarly to [20], we define a global (average) silhouette index that takes into account the respective sizes of the clusters:

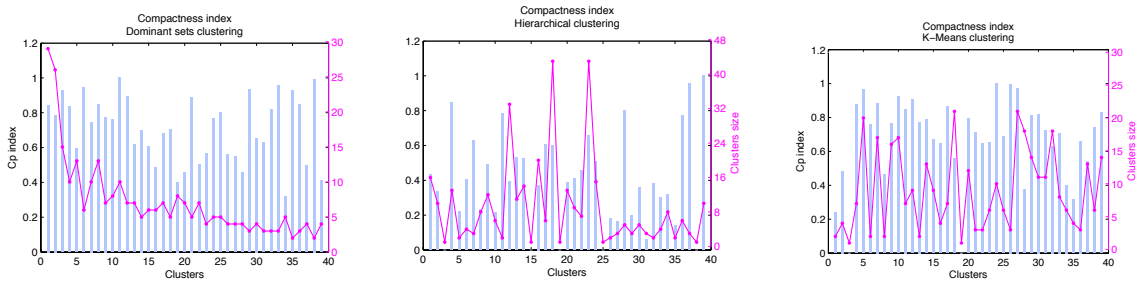
$$\text{GS} = \frac{\sum_{j=1}^{K} N_j S_j}{\sum_{j=1}^{K} N_j} = \frac{1}{m} \sum_{i=1}^{m} \text{silh}(\mathbf{x}_i)$$

Kaufman and Rousseeuw have argued that an average silhouette index of about 0.5 indicates a reasonable partition of the data.

3.4.3 Objective evaluation and comparison with other approaches

To conclude our clustering evaluation, and to demonstrate how well the dominant sets algorithm performs compared to other approaches, we have applied the Hierarchical clustering (HC) and K-Means methods to the very same data set as the one used previously in Sections 3.3 and 3.2. Remember that the feature vectors represent geographical distributions of machines involved in computer attack events in the Internet. The same distance metric has been used for all clustering methods, i.e., Jensen-Shannon divergence (using equation 3.4).

Fig. 3.10 shows the clustering results using HC (a) and K-Means (b) respectively (with 20 clusters as input). These results can be visually compared with those obtained using dominant sets in Fig. 3.7. An in-depth analysis of those results, along with the clusters centroïds, reveals there is a large number of points that are not well clustered in the case of HC and K-Means clustering, compared to the results obtained with dominant sets (even though the distance metric used was the same).



(a) (b) (c)

Figure 3.11: Compactness values by cluster (in blue) and clusters size (in magenta). (a) Dominant sets clustering (b) Hierarchical clustering (c) K-Means clustering.

Table 3.5: Objective evaluation of clustering results using different validity indices.

Clust.method	**Nr clusters**	$\overline{\text{Cp}}$ (↗)	$\overline{\text{DB}}$ (↘)	$\overline{\text{GS}}$ (↗)
DM1 - incomplete	39	0.73	**0.99**	**0.46**
DM2 - complete	39	0.54	1.77	0.32
HC1 - complete link	39	0.50	1.30	0.41
HC2 - complete link	20	0.39	1.49	0.41
K-Means (KM1)	39	**0.74**	2.15	0.33
K-Means (KM2)	20	0.66	1.91	0.39

To perform a more objective evaluation, we have applied the three validity indices defined here above to each clustering method. The values of those indices are given in Table 3.5. Interestingly, the dominant sets approach seems to outperform the two other clustering techniques, even when the number of clusters is set to the same number (i.e., $K = 39$). To further confirm this result, we have represented in Fig. 3.11 the individual compactness values (by cluster) for each clustering method, along with the clusters sizes. As one can see, this chart clearly demonstrates two things: (i) the compactness values are on average higher for clusters found by dominant sets and K-Means clustering, than with Hierarchical clustering (with complete linkage); and (ii) the dominant sets clustering technique seems to focus on the most significant (and probably most informative) clusters, as the first clusters found by this technique are also the largest ones with, at the same time, very high compactness values. Furthermore, assuming the analyst could figure out that about 39 clusters was a reasonable estimate (e.g., using the Gap statistic), then we observe that the result given by K-Means (with 39 clusters as input) performs quite well with respect to the clusters compactness. However, the two other indices (Davies-Bouldin and the global silhouette) are clearly in favor of dominant sets, which again confirms the meaningfulness of this graph-based approach.

3.5 Summary

In this Chapter, we have presented the two first components of our attack attribution method, i.e.: *attack feature selection* and *graph-based clustering*. Those steps implement a

novel graph-based clustering approach that rely on *dominant sets*, which can help security analysts to discover knowledge by identifying groups of highly similar patterns with respect to any attack feature selected from a security data set. In particular, we have demonstrated how to apply the dominant sets approach so as to get meaningful results. Finally, we have also emphasized the strong advantages of this approach compared to more classical clustering methods, such as Hierarchical clustering and K-Means.

However, the results of this Chapter also suggest that clustering real-world data sets is not trivial. In fact, the partitioning results of security data sets look often rather "fuzzy" regarding certain groups of patterns. Even with an effective clustering technique, it can be challenging for an analyst to identify the "correct" number of clusters, or the most appropriate partitioning of the data set.

In many cases, applying a clustering technique to a single attack feature is not sufficient for identifying attack phenomena, or to figure out what can be their root causes. Obviously, it is easy for an analyst to apply iteratively the very same clustering method w.r.t. each attack characteristic that can potentially bring an interesting viewpoint on the phenomena. However, how to combine clustering results in a meaningful way is still an open issue. For example, which *set of features* is the most appropriate one to identify a given phenomenon? Furthermore, observe that the characteristics of certain phenomena may evolve over time. Hence, the analyst can obtain several clusters, representing patterns observed at different moments; but at this point, he has no means to merge them and identify the ones that relate to the same phenomenon.

In the next Chapter, we will show how to systematically combine different attack features, so as to identify global attack phenomena in a meaningful and effective way.

Chapter 4

Attack Attribution using Multi-criteria Decision Analysis

"The purpose of models is not to fit the data but to sharpen the questions".
– Samuel Karlin

In the previous Chapter, we showed how cluster analysis, more precisely a graph-based clustering approach based on *dominant sets*, can reveal interesting patterns within an attack data set, which may help security analysts in investigating root causes of attack phenomena.

Intuitively, we can easily imagine that combining multiple attack features in a systematic way should further improve the identification process of attack phenomena and, perhaps more importantly, give insights into their dynamic behavior and their root causes. In this Chapter, we show that it is possible to use a multi-criteria decision approach to support the *attack attribution* process and to emphasize the *modus operandi* of attackers. As a result, the last component of our method takes advantage of different aggregation techniques inpired by *multi-criteria decision analysis* (MCDA), such that multiple attack features can be effectively combined without requiring a lot of *a priori* knowledge.

4.1 Introduction

4.1.1 Underlying motivation

Previously, we have seen that a graph-based clustering technique (based on dominant sets) can be a useful and effective technique to extract informative patterns from a set of observations. By repeating this process for different attack features, we can obtain one set of clusters w.r.t. each attack feature, and every set of clusters can thus provide an interesting viewpoint on the phenomena under study.

However, similarly to criminal forensics, a security analyst often needs to synthesize different pieces of evidence in order to investigate the root causes of attack phenomena.

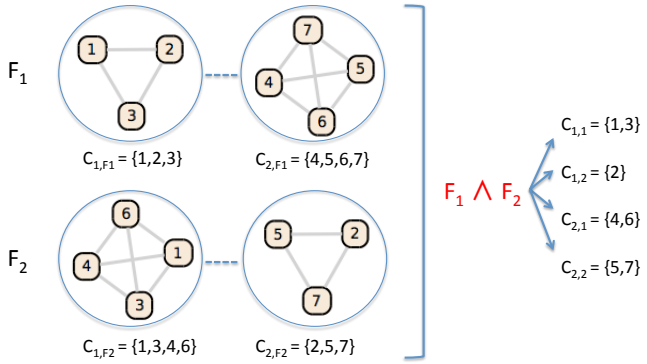
This can be a tedious, lengthy and informal process mostly relying on the analysts expertise. A somehow naïve approach of doing this aggregation of features consists in computing the intersections among all clusters obtained for each feature separately. Even though it could work for fairly simple cases, we observe that this approach does not hold for many attack phenomena we have analyzed in reality. There are two main issues explaining this:

(i) the *uncertainty problem*, which can be seen as the result of the *fuzzy* aspect of real-world phenomena.

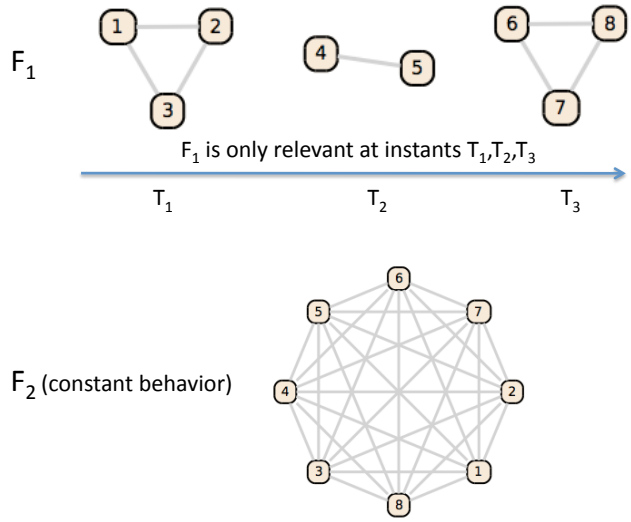
(ii) the *dynamicity problem*, which is due to the evolutive nature of real-world phenomena.

Let us illustrate those issues with two simple examples that we have observed in real-world data sets. Consider for instance Fig. 4.1 (a) where a set of 7 attack events have been clustered w.r.t. 2 different attack features F_1, F_2. In fact, the 7 events are due to the same root phenomenon, and ideally, we should have obtained only 1 cluster for each feature. However, in this case, the events have been separated into two "nearby" clusters, where the members of one cluster are somehow loosely connected to the members of the other cluster (e.g., pairwise similarities between events of different clusters could be around 0.5). This scenario is realistic, because real-world measurements are rarely "black" or "white", and thus real phenomena are often characterized by fuzzy patterns. In other words, any real-world measurement is impregnated by a certain amount of *uncertainty*, which often influences clustering results in an undesirable way. Furthermore, because patterns are most likely different for F_1 and F_2, chances are high that the composition of the two clusters, for F_1 and F_2 respectively, will also be different. Consequently, when the analyst computes the intersection among all clusters, the 7 events are split into 4 smaller groups, instead of one only. When the number of features and the number of observations of the data set increase, one can easily imagine that this can lead to a combinatorial explosion. In the worst case, the total number of groups could achieve $\prod_i^n |\mathcal{P}_i|$, where $\mathcal{P}_i$ is the partition obtained for the attack feature i, and there are n features. So, even if larger phenomena could still emerge from clusters intersections, we note that we get also a large number of small, meaningless clusters, and thus we lose a lot of semantics.

Another issue arises with the naïve approach of intersecting clusters, namely the *dynamicity problem*. Consider now Fig. 4.1 (b) where a set of 8 events have been clustered w.r.t. 2 different features F_1, F_2. All events are still due to the same phenomenon. But this time, the clustering algorithm has correctly split the events into three separate clusters for feature F_1, due to the fact that the phenomenon has been observed at three different instants, and its behavior has evolved regarding F_1. Those three clusters are definitively not linked to each other, as F_1 is only relevant at the different points in time on which we have observed the phenomenon. However, it is quite common to observe another attack feature for which the same phenomenon has a constant behavior, i.e., all events are grouped in the same cluster. By intersecting both dimensions, the analyst still gets 3 separate clusters instead of one only, which complicates again the identification of the global phenomenon.



(a) Illustration of the *uncertainty problem*, which is due to the fuzziness of real-world phenomena.



(b) Illustration of the *dynamicity problem*, which is due to the evolutive nature of real-world phenomena.

Figure 4.1: The two main issues in combining clusters obtained for different attack features.

4.1.2 The MCDA approach

As demonstrated in previous section, we need thus a more flexible way of combining attack features. Actually, the problem looks very similar to typical situations handled in *multi-criteria decision analysis* (MCDA), also called in the literature *Multi-Attribute Utility Theory* (MAUT). In a classical MCDA problem, a decision-maker evaluates a set of *alternatives* w.r.t. different *criteria*, and a global score is then calculated for each alternative using a well-defined *aggregation method* that models the preferences of the decision-maker or a set of constraints.

Generally speaking, the alternatives are evaluated w.r.t different attributes (or features) that are expressed with numerical values representing a degree of preference, or a degree of membership[1]. The two most common aggregation methods used in MAUT are the weighted arithmetic and geometric means.

[1]Of course, some attributes may sometimes be expressed using ordinal or qualitative values. Then, a commonly-used approach consists in converting ordinal values to a numerical scale using utility functions.

Another classical application relates to the *group decision making* problem, where n experts express their evaluations on one (or more) alternatives. The goal is then to combine all experts' preferences into a single score (like in many sports competitions, where the criteria are scores given by different judges). The most commonly used technique for combining experts' scores is the weighted arithmetic mean, since experts may be assigned different weights according to their standing.

Some other typical examples involving an aggregation process can be found in *fuzzy logic* and *rule-based systems*. In this case, the inference engine is made of fuzzy rules in which the rule antecedents model attributes that are subject to vagueness or uncertainty. The aim is to evaluate the "firing strength" of each fuzzy rule using logical connectives, and then to combine all rules' output into a single, crisp value that can be used to make a decision.

4.1.3 Combining edge-weighted graphs using MCDA

The last component of our attribution method aggregates all edge-weighted graphs obtained previously via the clustering component. We consider that each edge within a graph (reflecting a similarity between two nodes) provides a degree of evidence of the relation between a given pair of events of the security data set. We adopt thus a *relational* multi-criteria approach, where all alternatives (i.e., security events) are compared two by two. The aim of this aggregation is to determine, for each pair of events, how likely it is that those events are linked to the same root phenomenon, given the set of relations obtained by assessing different attack features.

Considering the two problems described in the previous paragraph, there are *at least* two important requirements regarding the aggregation method:

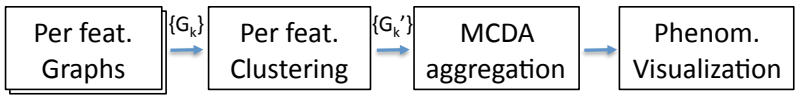
(1) two security events should have a high global score (i.e., be linked to the same phenomenon) when a sufficient number of attack features are correlated to a certain degree. Note that we do not need especially all features to be satisfied (as a result of the dynamicity of phenomena);

(2) conversely, the overall score reflecting the relation between two events should be minimized when the individual relations are so low that they could equally have occurred by chance;

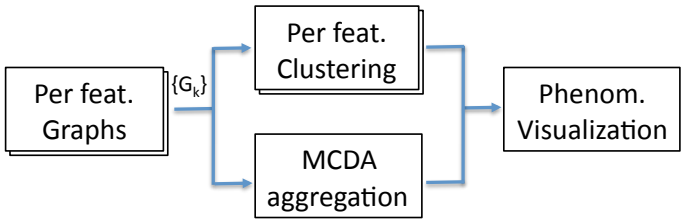
It is also worth mentioning that we propose two different options for the input of this aggregation step:

- **option 1**: *Serial combination*: in each edge-weighted graph (one per attack feature), we keep only all edges of the nodes that have been clustered using the dominant sets, as described in previous Chapter. Since dominant sets reflect strongly connected groups, we keep thus only the most relevant relations among security events w.r.t. every attack feature, and those subsets of relations are then used as input to the aggregation component to produce a set of phenomena (see Fig. 4.2 (a)).

- **option 2**: *Parallel processing*: we reuse the complete edge-weighted graphs as input to the aggregation component (i.e., also the links from or to unclustered events). The



(a) Option 1: Serial combination



(b) Option 2: Parallel processing

Figure 4.2: Illustration of the two possible options in combining edge-weighted graphs and per-feature clustering results. (a) **Serial combination**, where the per-feature clustering results have a direct impact on the input of the multi-criteria aggregation component. (b) **Parallel processing**, where all edges of each edge-weighted graph is reused as input to the multi-criteria aggregation, and per-feature clustering results are only exploited in the visualization step.

output of both components (per-feature clustering and aggregation components) are then combined in the visualization of the resulting phenomena (see Fig. 4.2 (b)).

Actually, both options have advantages and drawbacks: *option 1* will most likely provide cleaner results (in terms of final clusters), but it is possible to loose some semantics due to the fact that certain interesting nodes (or edges) may be excluded, as an artefact of the clustering algorithm. Similarly, *option 2* will most likely provide more detailed results and can help to recover certain events excluded by the per-feature clustering step, but some irrelevant events could be included by accident into final clusters, disturbing the interpretation of those results. However, for most experiments performed in the next chapters, we have used **option 2** (parallel processing), which still offers the more complete picture and helps to circumvent artefacts introduced by clustering algorithms.

The output of the multi-criteria component is thus a *combined graph* in which edges reflect aggregated scores among security events. All nodes that are still linked to others are very likely due to the same root cause, because of the requirements enounced here above. As final step, we just need to present the strongly connected components of the combined graph to the analyst. By visualizing those connected components using several attack features, the analyst gets immediately an overall picture of all important relations among events that are supposedly belonging to the same phenomenon. Maybe even more importantly, we get also insights into how those relations have eventually evolved (e.g., when subgraphs corresponding to different clusters co-exist within the same component of the combined graph).

In summary, Fig.4.3 illustrates graphically the general idea underlying this attack attribution method. Finally, we note some other important requirements of this method:

- the flexibility to include additional attack features when needed, so as to further improve the root cause analysis;

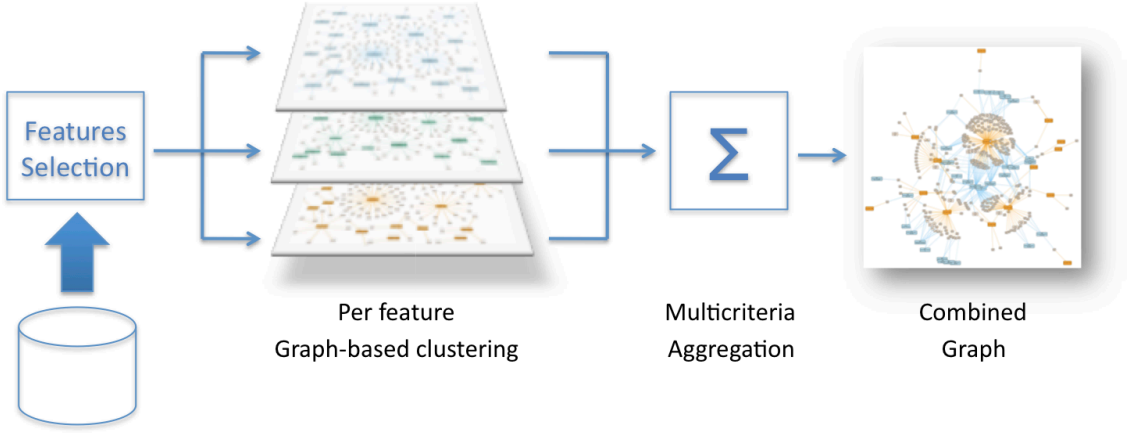


Figure 4.3: An overview of the general idea underlying our attack attribution method, in which multiple graphs are aggregated into a *combined graph* using a multi-criteria decision approach, so as to emphasize phenomena made of events linked by complex relationships.

- the unsupervised aspect of this classification, and because of this, there is a need to include some domain knowledge in a flexible way;
- the precise set of relevant features does not need to be specified in advance, and different combinations of features can apply to different subgroups of events.

As we demonstrate in the rest of this Chapter, MCDA aggregation techniques can satisfy all those requirements.

4.2 Formalizing the problem

4.2.1 Aggregation function

Aggregation functions are used in many prototypical situations where we have several criteria of concern, with respect to which we assess different options. The objective consists in calculating a combined score for each option (or alternative), and this combined output forms then a basis from which decisions can be made. More formally, an aggregation function can be defined as follows.

Definition 4.1. (Aggregation function [17]) *An aggregation function is formally defined as a function of n arguments ($n > 1$) that maps the (n-dimensional) unit cube onto the unit interval: $f_{aggr} : [0,1]^n \longrightarrow [0,1]$, with the following properties:*

(i) $f_{aggr}(\underbrace{0,0,\ldots,0}_{n\text{-times}}) = 0 \quad$ *and* $\quad f_{aggr}(\underbrace{1,1,\ldots,1}_{n\text{-times}}) = 1$

(ii) $x_i \leq y_i$ *for all* $i \in \{1,\ldots,n\}$ *implies* $f_{aggr}(x_1,\ldots,x_n) \leq f_{aggr}(y_1,\ldots,y_n)$

All unit intervals $[0,1]$ are considered here to be continuous, i.e., a variable defined on this unit interval may take any real value between the lower and upper bounds.

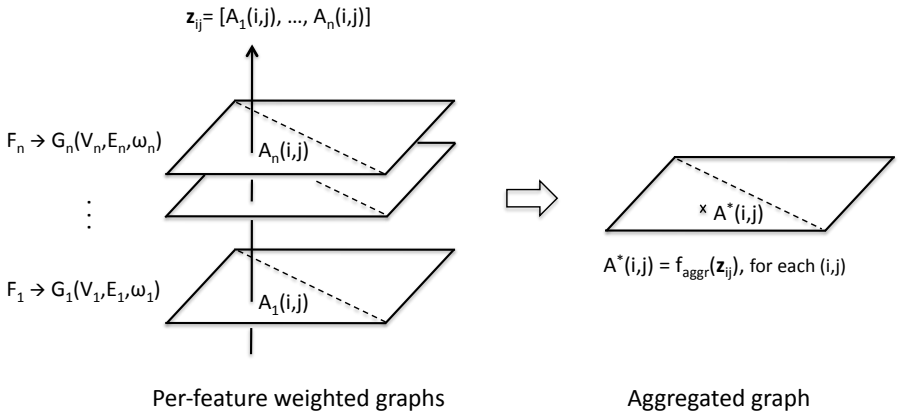


Figure 4.4: Illustration of the aggregation process performed on n edge-weighted graphs (represented by their respective proximity matrix), which leads to the construction of combined graph G^* that takes into account all attack features according to the *semantics* of f_{aggr}, the aggregation function.

In our multi-criteria attribution method, we have n different attack features F_k, whose indices can be put into a set $\mathcal{N} = \{1, 2, \ldots, n\}$. For each F_k, recall that we have built an edge-weighted graph $G_k = (V_k, E_k, \omega_k)$, represented by its corresponding similarity matrix $A_k(i, j) = \omega_k(i, j)$, with ω_k defined as an appropriate distance function.

Thus, for each pair of events (i, j) of the security data set $\mathcal{D}$, a *vector of criteria* $\mathbf{z}_{ij} \in [0, 1]^n$ can be constructed from the similarity matrices, such that:

$$\mathbf{z}_{ij} = [A_1(i, j), A_2(i, j), \ldots, A_n(i, j)] \tag{4.1}$$

Informally, our approach consists in combining these n values of each criteria vector $\mathbf{z}_{ij}$ which reflects the set of all relationships between a pair of security events. This results in the creation of an aggregated graph $G^* = \sum G_k$.

A rather simplistic approach consists in combining the criteria using a simple arithmetic mean, eventually with different weights assigned to each criteria (i.e., a weighted mean). However, this does not allow us to model more complex relationships, such as "most of", or "at least two" criteria to be satisfied in the overall aggregation function, without having to know *which* set of criteria is more relevant for a given pair of objects. So, we need an aggregation method in which the optimal combination of criteria (and the associated weights) is not predetermined in a static way. This multicriteria aggregation is extensively studied in Sections 4.3 and 4.4.

4.2.2 Building a combined graph G^*.

As illustrated in Fig.4.4, the multicriteria aggregation leads finally to the construction of a combined graph G^*, represented by its adjacency matrix A^*, which can be obtained through following operation:

$$A^*(i, j) = f_{aggr}(\mathbf{z}_{ij}),\ \forall (i, j) \in \mathcal{D} \tag{4.2}$$

Finally, we can extract the strongly *connected components* from G^* in order to identify

all subgraphs in which any two vertices are connected to each other by a certain path:

$$\begin{aligned} \mathcal{P} &= components(A^*) \\ &= \{P_1, P_2, \ldots, P_m\} \end{aligned}$$

which gives us our final set of connected subgraphs $\mathcal{P}$, where $P_x \subseteq G^*$, and $\forall (i,j) \in P_x : f_{aggr}(\mathbf{z}_{ij}) \geq \varepsilon$, with $\varepsilon \in]0,1]$.

Actually, this turns out to be again a graph clustering problem very similar to the per-feature clustering problem described in Chapter 3. As a result, the set of subgraphs $\mathcal{P}$ could be obtained by applying the very same *dominant sets* algorithm given in Section 3.3.

However, it is important to emphasize the fact that searching for dominant sets in the combined graph may be too restrictive in certain cases. While looking for fully connected subgraphs (i.e., dominant sets) provides very coherent groups in the case of a single attack feature, it can also become a limitation when several features are combined into a single graph. The reason is pretty simple: intuitively, real-world phenomena are dynamic by nature, which means that attack events can be linked by different *combinations of features*, and thus attack events observed at instant t_0 can have very different characteristics from those of the last observed events.

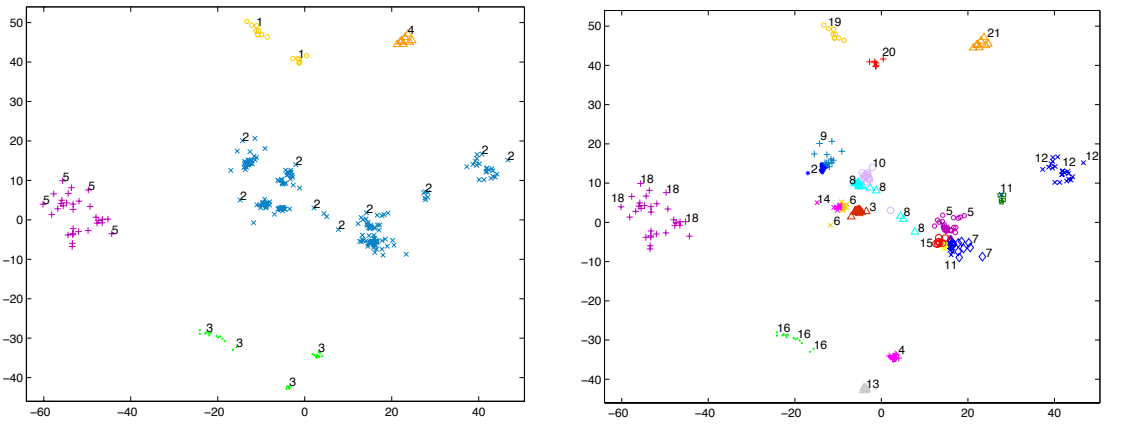
As a consequence of this evolving behavior, clusters in the combined graph G^* can present elongated shapes in which attack events are linked in a sort of *chaining structure*. While this "chaining effect" is usually not desirable in single feature clustering, it becomes really useful in the case of our combined graph G^* resulting from the aggregation of multiple features.

To illustrate this point, let us consider Fig.4.5 where a combined graph representing pairwise relations within a set of 297 attack events has been mapped on a 2-dimensional plot. From the multicriteria analysis, it turns out that those events have been attributed to 5 distinct phenomena only. Fig.4.5 (a) shows the result of applying a *connected components* algorithm to this data set. Via such a method, we are able to recover the 5 clusters corresponding to the behavior of the 5 phenomena, as expected. Observe the elongated shapes of clusters 1, 2 and 3.

In Fig.4.5 (b), we see now the result of applying *dominant sets* to the very same data set. As we can see, this algorithm is not able to recover the chaining structures of dynamic phenomena. For example, the very large and elongated structure of cluster 2 is split into a dozen smaller clusters. Similarly, the previous clusters 1 and 3 have been split into two and three different clusters, respectively.

This illustrates the eventual need for another graph clustering method to identify subgraphs in the combined graph G^*. We propose to extract *connected components* in order to be able to recover chained structures. However, a single linkage algorithm will most probably give similar results. There exist several well-known algorithms to extract connected components from a graph (e.g., depth-first search, breadth-first search, etc). We use here the *Dulmage-Mendelsohn* decomposition of A^*, which is a lightweight and efficient operation on matrices [43].

As we will show in the two next chapters with the help of real-world applications, each subgraph P_x represents quite likely a unique attack phenomenon. This can greatly help the analyst to figure out what are the root causes of the observed events. By analyzing



(a) Connected components (b) Dominant sets (fully connected subgraphs)

Figure 4.5: Illustration of the "**chaining effect**" for 5 real-world phenomena found in the combined graph G^*, which results from the dynamic behavior of those phenomena. (a) A graph clustering method based on *connected components* is able to recover the elongated structures of the 5 phenomena, as expected. (b) An algorithm focusing on *fully connected components* (such as *dominant sets*) is not designed to identify such kind of chaining structures. As a result, certain phenomena are split into several smaller clusters, which can make the root cause analysis harder for the analyst.

and visualizing those resulting phenomena through the clustering results of their individual features, we get immediately a global picture of all important relationships among observations, and hence we get also a better insight into the behavior of the underlying phenomenon.

Finally, it is also important to note that we must define a **decision threshold** ε, which is applied to A^* in order to eliminate combined edges that result from an unfortunate linkage between events having some weak correlation for a number of features (which means they would otherwise end up in the same subgraph while not related to the same root phenomenon). A reasonable default value for this threshold is obviously k/n, with k the minimal number of required criteria to decide keeping a link between two events.

As we show in Chapters 5 and 6, it is necessary (and relatively easy) to perform a *sensitivity analysis* on ε, in order to derive appropriate values for this threshold, and thus to obtain meaningful components.

4.2.3 Choice of an aggregation function

It is quite evident that the choice of the aggregation function f_{aggr} used to combine attack features is fundamental, as this function will model the behavior of the phenomenon under study. In other words, this choice must be guided by the *semantics* that the analyst wants to give to the aggregation procedure, and this procedure must provide appropriate means to model the characteristics of the attack phenomena under scrutiny.

Aggregation functions can be divided into following main classes:

- averaging aggregation functions, where the aggregated value of a vector of criteria $\mathbf{z}$ is bounded by
$$\min(\mathbf{z}) \leq f_{aggr}(\mathbf{z}) \leq \max(\mathbf{z})$$
- conjunctive aggregation functions, where the resulting value is bounded by
$$f_{aggr}(\mathbf{z}) \leq \min(\mathbf{z}) = \min(z_1, z_2, \ldots, z_n)$$
- disjunctive aggregation functions, where the resulting value is bounded by
$$f_{aggr}(\mathbf{z}) \geq \max(\mathbf{z}) = \max(z_1, z_2, \ldots, z_n)$$
- mixed aggregation functions, where the $f_{aggr}(\mathbf{z})$ exhibits different types of behavior on different parts of the domain.

In this Chapter, we provide an extensive study of two families of averaging functions, namely *Ordered Weighted Averaging* functions and the family of *Choquet integrals*, and we show how to take advantage of them to address attack attribution problems. This choice is motivated by the domain knowledge we have acquired by observing and analyzing attack phenomena in the Internet. Indeed, until today we haven't observed any phenomenon with a pure conjunctive (resp. disjunctive) behavior, i.e., a phenomenon in which the global score obtained from the combination of attack features was below (resp. above) all individual scores given for the attack features separately.

4.3 Ordered Weighted Averaging (OWA) functions

OWA functions are a family of averaging functions that basically rely on two main characteristics: (i) a weighting vector (like in a classical weighted mean), and (ii) sorting the inputs (usually in descending order), hence the name of *Ordered Weighted Averaging*. This reordering of the components of the input vector $\mathbf{z}$ introduces a non-linearity in the aggregation function. However, it allows the decision-maker to design slightly more complex modeling schemes when dealing with data fusion tasks.

This Section introduces two different OWA functions that can be helpful in the context of our attack attribution method. The application of these operators and their semantics are illustrated with a basic example on *attack events*. Finally, we briefly discuss the problem of how to determine appropriate values for the weighting vector.

4.3.1 Classical OWA function

In [192], a new type of averaging operator called *Ordered Weighted Averaging* (OWA) was introduced. The OWA operator allows one to include certain relationships among criteria, such as "most of" or "at least" k criteria (out of n) to be satisfied in the overall aggregation process. OWA differs from a classical weighted means in that the weights are not associated with particular inputs, but rather with their *magnitude*. As a result, OWA can emphasize

the largest, smallest or mid-range values. It has become very popular in the research community working on fuzzy sets.

Definition 4.2. (OWA) [192, 17] *For a given weighting vector* $\mathbf{w}$*,* $w_i \geq 0$*,* $\sum w_i = 1$*, the* OWA *aggregation function is defined by:*

$$OWA_{\mathbf{w}}(\mathbf{z}) = \sum_{i=1}^{n} w_i z_{(i)} =< \mathbf{w}, \mathbf{z}_{\searrow} > \tag{4.3}$$

where we use the notation $\mathbf{z}_{\searrow}$ *to represent the vector obtained from* $\mathbf{z}$ *by arranging its components in decreasing order:* $z_{(1)} \geq z_{(2)} \geq \ldots \geq z_{(n)}$*.*

It is easy to see that for any weighting vector $\mathbf{w}$, the result of OWA lies between the classical **and** and **or** operators, which are in fact the two extreme cases when $\mathbf{w} = (0, 0, \ldots, 1)$ (then $OWA_{\mathbf{w}}(\mathbf{z}) = min(\mathbf{z})$) or when $\mathbf{z} = (1, 0, \ldots, 0)$ (then $OWA_{\mathbf{w}}(\mathbf{z}) = max(\mathbf{z})$). Another special case is when all weights $w_i = \frac{1}{n}$, which results in the classical arithmetic mean.

Having defined the OWA operator, we can easily apply it to all vectors of criteria $\mathbf{z}_{ij}$. That is, referring to equation 4.2, the proximity matrix A^* of the combined graph G^* is simply obtained by executing the following operation:

$$A^*(i, j) = OWA_{\mathbf{w}}(\mathbf{z}_{ij}),\ \forall (i, j) \in \mathcal{D} \tag{4.4}$$

We have implemented the OWA operator in a MATLAB® function using the vectorized approach. Assuming that we have m security events and n attack features, this vectorized approach allows the OWA operator to be applied to m^2 elements of n similarity matrices at once. The only limitation of this approach is the amount of memory that is needed (on a single computer) to store the n matrices, certainly when the number of events m becomes large. This said, we could easily perform experiments on data sets comprising up to 10,000 events, using 8 attack features concurrently. Note also that this type of computation is quite easy to parallelize, should scalability become an operational issue.

Obviously, the problem of defining the weights w_i to be used in OWA still remains. Yager suggests two possible approaches: (i) either to use some learning mechanism, with sample data and a regression model (e.g., fitting weights by using training data and minimizing the least-square residual error), or (ii) to give some semantics, or meaning to the w_i's by asking a decision-maker or an expert to provide directly those values, based on *domain knowledge*. In many attack attribution cases, we have to rely on the latter, since the process is mostly unsupervised, and thus we have no training samples for the phenomena we aim to identify. We further discuss the problem of defining OWA weights in Section 4.3.4.

4.3.2 Weighted OWA (WOWA)

Weighted averaging functions, such as OWA or the weighted mean, can be quite convenient aggregation functions when we deal with data fusion tasks, in which criteria of interest are expressed with numerical values (usually, in $[0, 1]^n$). However, the weights used in weighted

mean (WM) and the ones defined for the OWA operator play very different roles. The WM takes into account the *reliability* of each information source (or each expert), whereas the weights used in OWA reflect the importance of the *values*, regardless of their source.

Torra proposed in [171] a generalization of both WM and OWA, called *Weighted OWA* (WOWA). This aggregation function combines the advantages of both types of averaging functions by allowing the user to quantify the reliability of the information sources with a vector $\mathbf{p}$ (as the weighted mean does), and at the same time, to weight the values in relation to their relative position with a second vector $\mathbf{w}$ (as the OWA operator).

The rationale underlying the WOWA function becomes clear in certain intelligent systems in which you have different sensors having a certain reliability (which is known, thus we have the weights $\mathbf{p}$), and where the measurements of those sensors have to be somehow prioritized, irrespective of their reliability. Think, for example, of an automated braking system which assists the driver in a vehicle. In this case, the reliability of the sensors should probably be taken into account (since less reliable sensors may fail or give erroneous measurements); however, the distance measurements to the nearest obstacles may be even more important in certain situations, regardless of which sensors provide the inputs. A WOWA function provides exactly this kind of combination.

In the rest of this section, we start by formally defining the WOWA aggregation function, which can replace the OWA operator in equation 4.4. Then, we illustrate its usefulness in the context of the attack attribution method, and how it performs compared to the classical WM or OWA.

Definition 4.3. (Weighted OWA [171]) *Let* $\mathbf{w}, \mathbf{p}$ *be two weighting vectors with* $w_i, p_i \geq 0$, $\sum w_i = 1$, $\sum p_i = 1$. *The* Weighted OWA *aggregation function is defined by:*

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = \sum_{i=1}^{n} u_i z_{(i)}, \tag{4.5}$$

where $z_{(i)}$ *is the* i^{th} *largest component of* $\mathbf{z}$ *and the weights* u_i *are defined as*

$$u_i = G\left(\sum_{j \in H_i} p_j\right) - G\left(\sum_{j \in H_{i-1}} p_j\right)$$

where the set $H_i = \{j | z_j \geq z_i\}$ *is the set of indices of* i *largest elements of* $\mathbf{z}$*, and* G *is a monotone non-decreasing function that interpolates the points* $(i/n, \sum_{j \leq i} w_j)$ *together with the point* $(0, 0)$*. Moreover,* G *is required to have the two following properties:*

1. $G(i/n) = \sum_{j \leq i} w_j$, $i = 0, \ldots, n$;

2. G *is linear if the points* $(i/n, \sum_{j \leq i} w_j)$ *lie on a straight line.*

In fact, the $WOWA$ operator can be seen as an OWA function with the weights u_i, which are obtained by combining both vectors $\mathbf{w}, \mathbf{p}$ using a generating function G. The mathematical properties of the $WOWA$ operator have been studied and described in [170, 171]. Among others, it has been showed that the weighting vector $\mathbf{u}$ satisfies $u_i \geq 0$, and $\sum u_i = 1$. It is also worth noting that if all $w_i = \frac{1}{n}$, then it turns out that

$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = M_{\mathbf{p}}(\mathbf{z})$, the weighted arithmetic mean. Similarly, when all $p_i = \frac{1}{n}$, then $WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = OWA_{\mathbf{w}}(\mathbf{z})$.

Obviously, the weights $\mathbf{u}$ also depend on the choice of the interpolation function G, also called W^* in [172]. As suggested by different authors, this function can be chosen as a linear spline that interpolates the points $(i/n, \sum_{j \leq i} w_j)$, or it can be also a monotone quadratic spline as was suggested in [171, 16], as long as the function satisfies the properties stated here above (i.e., the straight line condition). In the experiments performed in this dissertation, we have used the interpolation method described in [172], by wrapping the Java implementation of the author in MATLAB® functions.

Again, the WOWA operator can be applied to all vectors of criteria $\mathbf{z}_{ij}$ obtained from n weighted graphs, so as to build a combined graph represented by its proximity matrix A^*. Thus, in equation 4.4, the OWA can simply be replaced by the WOWA function, i.e.:

$$A^*(i,j) = WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}_{ij}),\ \forall (i,j) \in \mathcal{D} \tag{4.6}$$

Clearly, the advantage of using the WOWA operator (instead of OWA) comes from the additional vector of weights $\mathbf{p}$, which provides more flexibility in the decision-making process by quantifying the reliability of each feature (or at least, the way we can measure it).

4.3.3 OWA functions: an illustrative example on attack events.

Let us illustrate the use of OWA functions in the context of our attack attribution method. Like in Chapter 3, suppose we are dealing with a data set made of computer attacks observed in the Internet thanks to a set of distributed sensors. The attack phenomena manifest themselves under the form of numerous *attack events*, which comprise groups of malicious sources targeting in a coordinated fashion other machines. However, such attack events are almost continuously observed in the Internet every day, and thus it is very hard for an analyst to distinguish which events should belong to the same phenomenon. Moreover, identifying groups of events having apparently the same root cause is of great interest for the analyst who can directly focus on the most interesting (or dangerous) phenomena (i.e., "triage" process).

(1) Analysis of attack features

For the sake of illustration, imagine that we can extract four different attack features (characterizing every attack event):

- F_{geo}, which represents the spatial distribution of the attackers in terms of originating countries, as a result of mapping IP addresses to the countries they have been assigned to;

- F_{sub}, which represents the distribution of IP networks of the attackers (e.g., IP addresses grouped by their Class A-subnet);

- F_{time}, which represents the temporal distribution of attacking sources (i.e., a time series representing a number of sources grouped by day, or by hour, etc);

- F_{act}, which represents the type of attack activity performed by attackers (e.g., the targeted ports, the type of exploit, etc).

Clearly, the two first features are not independent of each other, since there is a direct mapping between IP networks and geographical countries where local ISP's are usually responsible for the management of well-defined IP blocks attributed according to a certain logic. In other words, F_{geo} and F_{sub} can be viewed as two different ways of measuring the same concept (the origins of the attackers). Consequently, the two features are somehow *redundant* and two events having the same distributions of origins will most likely exhibit high similarities for both features, although not necessarily with the same amplitude. However, the analyst may consider F_{sub} as a more reliable measurement than F_{geo} for following reasons: (i) due to varying country sizes and significant differences in economic situations, the degree of connectivity is very different from country to country; thus, measuring attackers coming from countries such as US, CA, DE or CN is *a priori* much more likely to occur than for other countries; and (ii) the databases that map IP addresses to geographical location are usually difficult to maintain up to date, so this can be an additional source of error.

Regarding F_{time}, this feature will measure the temporal degree of coordination between groups of attackers that belong to different attack events. As a result, we can consider that this feature provides a more reliable measurement of the likelihood that two events could be due to the same phenomenon. Note that this attack characteristic is naturally obtained by the correlation method used to identify such attack events (see [117] for more details). However, F_{time} *by itself* is not sufficient to link two events, since different botnets or worms can perfectly target the same sensors in the very same interval of time, with roughly the same daily pattern.

Finally, F_{act} represents the activities of the attackers, for example in terms of targeted TCP/UDP ports. Even though two events targeting exactly the same ports might be due to the same root cause, this feature is again not sufficient by itself, considering the fact that certain TCP or UDP ports are much more likely to be exploited than other unusual ports (for which no vulnerabilities have been found in potential services running on those ports). However, the reliability of this measurement is considered as just normal, i.e., the same as for F_{sub}.

Summarizing, after this preliminary analysis of the attack features under consideration, the analyst is able to express the following preference relationships:

$$F_{geo} \prec F_{sub} \sim F_{act} \prec F_{time}$$

(2) Multi-criteria analysis: definition of weighting vectors

Following the multi-criteria relational approach described in sections 4.1 and 4.2, we can build an edge-weighted graph for each F_k by computing all pairwise similarities among attack events (as described in previous Chapter). We are now interested in modeling the most appropriate way of *combining* all those similarities into a single value, from which a decision to keep a link (or not) between a pair of events can be made. It is thus important that the aggregation method can fulfill specific constraints that reflect the preferences of the analyst regarding the behavior of the phenomena he wants to identify.

For the purpose of this illustration, we consider that the analyst wants to model the following behavioral constraints, based on his domain knowledge:

(i) two attack events are most likely linked to the same root phenomenon (and hence, have a high combined score) when at least two **different** attack features are correlated (recall that F_{geo} and F_{sub} are two redundant features, and should ideally account for only one global feature in the aggregation process).

(ii) attack phenomena can have a dynamic aspect, so the analyst does not know in advance which combination of features (among $\binom{2}{n}$ possibilities) is relevant to link two events.

(iii) only one attack feature is not sufficient to link two events to the same phenomenon, as this could be due to chance only (e.g., due to large or popular countries, or IP networks that are largely infested by zombie computers, or some types of activities that are much more often observed than others, etc);

(iv) as described here above, certain attack features are more reliable than others for determining how likely it is that two events are linked to the same root cause.

Intuitively, it is easy to see that a simple arithmetic mean can not model those requirements. However, using OWA functions, we can try to model those requirements in a more effective way. For example, let us define the two following weighting vectors:

$$\mathbf{w} = [0.1, 0.3, 0.4, 0.2] \quad \text{and} \quad \mathbf{p} = [0.15, 0.25, 0.35, 0.25]$$

In fact, the requirements (i) to (iii) are translated into the values of vector $\mathbf{w}$ which is to be used in an OWA function. With those values for $\mathbf{w}$, the importance of the highest score will be lessened by $w[1] = 0.1$, and thus we need **at least two** high scores in order to get a combined value above a predetermined high threshold ε (e.g., above 0.5).

The last requirement (i.e., the reliability of the sources) can then be expressed by vector $\mathbf{p}$, which can be used either in a weighted mean, or in a WOWA aggregation function in conjunction with $\mathbf{w}$.

(3) OWA evaluation

To illustrate how those OWA functions perform compared to classical means, let us consider some typical relations that can possibly link attack events (which are inspired by real-world cases we have observed).

Fig. 4.6 represents a set of events observed at different points in time. The events $\{e_0, e_1, e_2, e_3\}$ are due to a same phenomenon, which is characterized by a set of relations that are summarized in Table 4.1 on page 79 (in columns F_k). Here is the interpretation of those relations:

- in row number 1, (e_0, e_1) is a case where the events differ by only one feature (F_{act}), while the three other features are perfectly correlated.

- for (e_1, e_2) in row number 2, the attack events are highly correlated by F_{time} and F_{act}, but the origins of the attackers have somehow evolved between the two (note

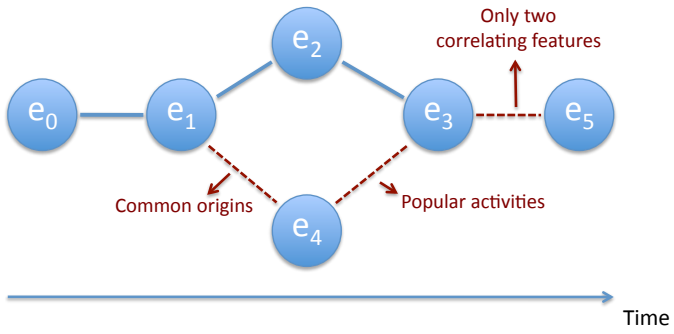


Figure 4.6: A set of events observed at different points in time, and where $\{e_0, e_1, e_2, e_3\}$ are assumed to be due to a same phenomenon. The two other events $\{e_4, e_5\}$ are not linked to the same root cause, however, they interfere with the previous ones. That is, the relations represented by dashed lines should be lessened by the aggregation method, whereas those in solid lines must be reinforced.

the lower similarity values for F_{geo} and F_{sub}). At the extreme, we could have a similar case (row number 3) where $(e_1, e_2)' = [0.1, 0.1, 0.8, 0.8]$, i.e., F_{time} and F_{act} are still strongly correlated but the origins of the phenomenon have changed even more.

- looking at (e_2, e_3) in row number 4, we can see that they have $F_{geo}, F_{sub}, F_{time}$ as common features, but the activities of the attackers have changed between the two events. At the extreme (row number 5), we could have a similar case $(e_2, e_3)'$ where F_{geo} has a lower similarity value, because measuring this feature is less reliable. However, in both cases the analyst wants the events to be attributed to the same root cause since F_{geo} and F_{sub} are substitutable features.

The combined scores of all those cases (rows 1 to 5) are thus required to be **reinforced** (i.e., maximized) by the aggregation function.

Then, we can also observe two events e_4, e_5 that are weakly correlated with the other events. However, those weak relations are only due to chance and can be seen as *interferences*. More precisely, we interprete those interfering relations among events as follows:

- in row number 6, because they both originate from popular countries, e_1 and e_4 are weakly correlated w.r.t. F_{geo} and F_{sub}. Sadly enough, the other two features present also a low similarity score, which gives an average score just below 0.5. At the extreme (row number 7), we could have a similar case where F_{geo} and F_{sub} are strongly correlated (by chance), but the other features are not. This is an interesting case where the **average value is greater than 0.5**, but the analyst still wants those events to be **separated** because there is only one correlated dimension (i.e., the origins), which is not sufficient to deduce that the root cause is the same. Hence, the aggregated value for this case must be minimized, i.e., as far as possible below the average value 0.5.

- regarding (e_3, e_4) in row number 8, those events are strongly correlated by F_{act} only, which is merely due to the fact that certain types of attack are more often observed than others. Unluckily, the other features present again some weak correlation, giving an overall average similarity of 0.5.

Table 4.1: Overview of some typical relations for a set of events where $\{e_0, e_1, e_2, e_3\}$ belong to a same root phenomenon, whereas the other events $\{e_4, e_5\}$ do not but interfere with the previous ones. The columns F_k refer to the similarity scores (i.e., the components z_i of each vector of criteria $\mathbf{z}$) regarding each attack feature, and the last 4 columns give the aggregated score obtained with different averaging functions. Rows 1-5 illustrate cases where the global score must be reinforced (i.e., above 0.5), whereas rows 6-9 illustrate cases that must be minimized (i.e., global score under 0.5). F_{geo} and F_{sub} are two *redundant* features.

Row nr	Event pairs	F_{geo}	F_{sub}	F_{time}	F_{act}	M	$M_{\mathbf{p}}$	$OWA_{\mathbf{w}}$	$WOWA_{\mathbf{w},\mathbf{p}}$
1	(e_0, e_1)	1	1	1	0	0.75	0.75	0.80	0.80
2	(e_1, e_2)	0.4	0.2	1	1	0.65	0.71	0.66	0.74
3	$(e_1, e_2)'$	0.1	0.1	0.8	0.8	0.45	0.52	0.45	**0.54**
4	(e_2, e_3)	0.9	0.7	1	0	0.65	0.66	0.67	0.68
5	$(e_2, e_3)'$	0.1	0.9	1	0	0.50	0.59	0.49	**0.61**
6	(e_1, e_4)	0.9	0.3	0.2	0.3	0.45	0.43	0.36	0.33
7	$(e_1, e_4)'$	0.9	0.9	0.2	0.3	0.58	0.51	0.58	**0.47**
8	(e_3, e_4)	0.45	0.45	0.1	1	0.50	0.47	0.43	0.40
9	(e_3, e_5)	1	1	0	0	0.50	0.40	0.50	**0.33**

- in row number 9, we see a typical case where only the two redundant features F_{geo} and F_{sub} are perfectly correlated; however, the average score of 0.5 must be lessened since those features should be considered as referring to the same aspect of the attacks.

As a result, the aggregation function must be able to lessen those side effects by **minimizing** the global score for those pairs of events (rows 6 to 9).

(4) Discussion of the results

In Table 4.1, we can observe the global scores of all vectors of criteria for the cases described previously, as calculated with the arithmetic mean (M), the weighted mean ($M_{\mathbf{p}}$), the Ordered Weighted Average ($OWA_{\mathbf{w}}$) and the Weighted OWA ($WOWA_{\mathbf{w},\mathbf{p}}$), using the weighting vectors $\mathbf{w}, \mathbf{p}$ defined on page 77. In light of the requirements (at least 2 different features), the analyst could be tempted to take as decision threshold ε a value around 0.5, which is the average for 2 (out of 4) high similarity values.

However, observe that many "difficult" cases have a global score that situates around this value. It is interesting to note that the choosen OWA operator can help to reduce the influence of the highest similarity score, but due to the fact that the analyst is unable to decide **which** highest feature must be given a lower (resp. higher) importance, the OWA operator will fail in certain cases. That is, in rows 3 and 5, we see that eliminating the impact of the highest criterion is too restrictive, whereas in cases indicated by rows 7 and 9, removing the influence of the highest score is not sufficient!

Not surprisingly, the WOWA operator provides here some improvement over the OWA operator. Thanks to the second vector of weights $\mathbf{p}$, which reflects the reliability of the attack features, the analyst can model his requirements in a more effective way. For example, we see that for the case in row number 3, this operator can help to increase the combined score from the average of 0.45 up to **0.54**. Similarly, looking at the result for

the case in row 7, the WOWA operator can help to reduce an unwanted high score (with an average of 0.58) down to **0.47**.

In conclusion, by using OWA aggregation functions, we can observe from the examples given in Table 4.1 that (i) more complex behaviors are modeled in a more effective way; (ii) OWA operators can help to emphasize the gap between some *desired* and *unwanted* cases of events linkage that can occur due to uncertainty or dynamicity issues; and (iii) in the presence of interactions between features (like a **redundancy** or a **synergy**), the analyst still needs more flexible means to model the aggregation of features, such as the ability to model the influence of certain *coalitions* of features. As we demonstrate in Section 4.4, this issue can be solved by using a **Choquet integral** defined w.r.t. a fuzzy measure.

4.3.4 Strategies to determine weights in OWA.

As illustrated previously, the most natural way of defining the weighting vector in OWA consists in giving semantics to individual weights based on *domain knowledge*, e.g., by asking an expert to provide directly those values. However, there are some other strategies that can help the decision-maker when knowledge about the phenomena is absent or quite limited.

A first approach consists to set some constraints on certain characteristics of averaging functions, such as the *weights dispersion* and the *orness* (or degree of disjunction) of the function. Another approach is to use optimization methods to fit weights to some empirical data, assuming we know what is the desirable output that corresponds to every input sample (i.e., an approach based on *supervised learning*). We briefly discuss both approaches in this section.

Orness and Dispersion.

Informally, the *orness* (also called the *degree of disjunction*, or the *attitudinal character*) measures how far a given averaging function is from the max function, which is the weakest disjunctive function. For any given averaging function f_{aggr}, its degree of orness is defined by ([53, 134]):

$$orness(f_{aggr}) = \frac{\int_{[0,1]^n} f_{aggr}(\mathbf{x})d\mathbf{x} - \int_{[0,1]^n} \min(\mathbf{x})d\mathbf{x}}{\int_{[0,1]^n} \max(\mathbf{x})d\mathbf{x} - \int_{[0,1]^n} \min(\mathbf{x})d\mathbf{x}} \tag{4.7}$$

Given this definition, it is clear that $orness(\max) = 1$ and $orness(\min) = 0$, whereas $orness(f_{aggr}) \in [0, 1]$ for any other function. In the specific case of the OWA function, it turns out that the above definition translates into the following simple equation ([192]):

$$orness(OWA_{\mathbf{w}}) = \sum_{i=1}^{n} w_i \, \frac{n-i}{n-1} \tag{4.8}$$

Another useful interpretation of the orness relates to the *degree of compensation* of an averaging function, i.e., the degree of orness measures to what extent a "bad" score (i.e., a small value) for one of the inputs influences the output ([173]). Observe also that the orness of the arithmetic mean (M) and the weighted mean ($M_{\mathbf{w}}$) is always equal to $\frac{1}{2}$ (regardless of the weighting vector $\mathbf{w}$ used for $M_{\mathbf{w}}$).

Next, another important numerical characteristic of an averaging function is the *weights dispersion*, also called the *entropy*. Basically, this quantity measures the degree to which all the information (i.e., all inputs z_i) is used in the aggregation process. For any given weighting vector $\mathbf{w}$, we can quantify its dispersion (or entropy) by using

$$Disp(\mathbf{w}) = -\sum_{i=1}^{n} w_i \log w_i \tag{4.9}$$

with the usual convention that $0 \cdot \log 0 = 0$. The maximal dispersion is equal to $\log n$, which is achieved at $w_i = \frac{1}{n}$ for all weights (i.e., arithmetic mean). The minimal dispersion is equal to 0, which is achieved for $w_i = 0$, $i \neq k$, and $w_k = 1$ (i.e., the order statistic).

Example of orness and entropy.
Turning back to previous illustration of section 4.3.3, it is easy to calculate those indices for the chosen OWA vector $\mathbf{w} = [0.1, 0.3, 0.4, 0.2]$, i.e.:

$$Disp(\mathbf{w}) = 1.28 \text{ and } orness(OWA_{\mathbf{w}}) = 0.46$$

The value of the weights dispersion indicates that all inputs are well taken into account in the aggregation, since the maximum entropy in this case is equal to $\log 4 = 1.38$, and the value obtained, 1.28, is close to that maximum dispersion. On the other hand, the orness value confirms that this $OWA_{\mathbf{w}}$ function compensates slightly more than a simple arithmetic mean (i.e., bad scores have a greater influence on the final output). This can be derived from the orness value 0.43 which is smaller than 0.5, the orness value for the arithmetic mean.

Maximum entropy and minimum variance.

Fullér and Majlender proposed two methods for choosing OWA weights that are based on various measures of weights dispersion (or entropy) [56].

In the first approach, the idea is to determine, for a given n and orness measure α, a vector of weights w_i that maximizes the dispersion $Disp(\mathbf{w})$ as given by equation 4.9. This can be formulated as the following optimization problem:

$$\min \sum_{i=1}^{n} w_i \log w_i \tag{4.10}$$

$$\text{such that } \begin{cases} \displaystyle\sum_{i=1}^{n} w_i \, \frac{n-i}{n-1} = \alpha \\ \displaystyle\sum_{i=1}^{n} w_i = 1, \; w_i \geq 0, \; i = 1, \ldots, n \end{cases}$$

The solution to this problem is given in [56] and is called Maximum Entropy OWA (MEOWA). Using the method of Lagrange multipliers, the authors could find analytical

expressions for the determination of the weights w_i under the conditions stated here above[2]. Although we haven't used this MEOWA operator, it is worth noting that it can help the analyst to perform a preliminary analysis on an unknown data set when the domain knowledge is rather limited (since the only parameter one needs to specify is the orness value α, which reflects the degree of compensation of the aggregation function).

Fullér and Majlender proposed also in [57] to optimize another popular characteristic of weighting vector, namely the *weights variance*, which is defined by

$$D^2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(w_i - M(\mathbf{w}))^2 = \frac{1}{n}\sum_{i=1}^{n} w_i^2 - \frac{1}{n^2} \tag{4.11}$$

where $M(\mathbf{w})$ is the arithmetic mean of $\mathbf{w}$. Here, the idea is to minimize $D^2(\mathbf{w})$ according to a given orness value. The solution to this problem is called Minimum Variance OWA (MVOWA). Since adding a constant value to an objective function does not change the minimizer, the problem reduces to the following optimization:

$$\min \sum_{i=1}^{n} w_i^2 \tag{4.12}$$

$$\text{such that } \begin{cases} \displaystyle\sum_{i=1}^{n} w_i \, \frac{n-i}{n-1} = \alpha \\ \displaystyle\sum_{i=1}^{n} w_i = 1, \; w_i \geq 0, \; i = 1, \ldots, n \end{cases}$$

The optimal solution to this problem can be found in [57] where analytical expressions have been calculated for the weights w_i under the above requirements. Note that for $\alpha = \frac{1}{2}$, the optimal solution is always $w_i = \frac{1}{n}$, $i = 1, \ldots, n$.

Fitting weights to empirical data.

Until now, we have described an attack attribution method that is essentially unsupervised, and thus the only information the analyst could rely on to define aggregation parameters (i.e., weights) was basically his general domain knowledge about attack phenomena. However, it is reasonable to expect that, after a while, certain phenomena discovered by this unsupervised approach can be further validated by new observations. Consequently, some empirical data could be gathered on those specific phenomena. For example, assuming a researcher discovers a new type of botnet and he is able to track the behavior of bots belonging to that botnet, then one could possibly collect measurements (like geographical location of bots, their IP subnets, their activities, etc) on some events attributed to the botnet, and compute similarities with some other events that are knowingly not part of the same phenomenon.

Hence, given a set of training samples $(\mathbf{z}_k, y_k)$, $k = 1, \ldots, K$, we can now examine the problem of finding (or adjusting) OWA weights that best fit the empirical data. The goal is to find a weighting vector that minimizes the differences between the predicted $(OWA(\mathbf{z}_k))$

[2] A different (yet equivalent) representation of the same solution was given in [25]

and observed (y_k) values, under a given optimization scheme.

This can be formulated as an optimization problem, e.g., by using a least squares or least absolute deviation criterion. In the first case, we need then to solve the following problem:

$$\min \sum_{k=1}^{K} \left(\sum_{i=1}^{n} w_i z_{(i)k} - y_k \right)^2 \tag{4.13}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0,\ i = 1, \ldots, n$$

whereas the optimization under a least absolute deviation criterion is formulated as:

$$\min \sum_{k=1}^{K} \left| \sum_{i=1}^{n} w_i z_{(i)k} - y_k \right| \tag{4.14}$$

$$\text{s.t.} \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0,\ i = 1, \ldots, n$$

This problem of fitting OWA weights to empirical data was studied by several authors ([52, 191], among others). It is worth noting that a common feature of all methods is to eliminate the nonlinearity due to the reordering of the input vector $\mathbf{z}$. For this reason, the optimization problems are formulated using an auxiliary data set where all input vectors have been reordered, i.e.: $z_{(1)k} \geq z_{(2)k} \geq \ldots \geq z_{(n)k}$, for every pair $(\mathbf{z}_k, y_k)$ of the training data set. Note also that it is generally assumed $K \gg n$, otherwise multiple optimal solutions exist (due to rank deficiency). Some additional requirements can be incorporated in the optimization problem, such as the desired value of $orness(OWA) = \alpha \in [0, 1]$.

The optimization problems 4.13 and 4.14 can be solved using either linear or quadratic programming techniques, depending on the optimization criterion. Beliakov et al. provide in [17] a more detailed discussion on how to formulate and solve those problems using numerical analysis methods. Unfortunately, in the context of this dissertation we couldn't gather any "good" training data set to further validate this weights fitting approach. So, we leave the experimental validation of this supervised approach as future work.

4.3.5 A final note on OWA functions.

We observe that the family of OWA functions has been further studied and generalized by other authors. Many other types of OWA functions have been developed, such as the Ordered Weighted Geometric function (OWG), the Neat OWA, the Generalized OWA, or OWA functions based on *power functions* (similar to weighted power means). Similarly, yet other methods have been developed to determine optimal OWA weights, for example methods based on weight generating functions (e.g., Regular Increasing Monotone, or RIM quantifiers, such as in [193, 194]).

Note that our method is not *per se* limited to the use of the OWA functions herein presented. However, in this Section we have clearly demonstrated how certain OWA aggregation functions can help a security analyst, in a simple yet effective manner, to combine

evidences and make a decision on the attribution of security events to a common root cause. We hope it will inspire other researchers to explore the utility of other multi-criteria methods and aggregation functions in the realm of *attack attribution*).

4.4 On the use of Choquet integral

4.4.1 Introduction

Until now, we have showed how certain types of aggregation functions, such as OWA, can help to combine attack features which are represented by pairwise relations between attack events. However, in many multi-criteria problems, and by consequent also in attack attribution problems, it happens quite often that certain criteria are not completely independent. For example, certain combinations of criteria may show some sort of *synergy* (or complementarity), whereas other subsets of criteria could have globally less importance due to the presence of *redundancy* (or substitutability) among them.

Fuzzy integrals are a large family of aggregation functions that can provide effective means for modeling this kind of behavior in MCDA problems. Broadly speaking, there are two main families of functions:

- the *Choquet integral*, which is used for the aggregation of criteria defined on cardinal scales. That is, the scores of the criteria are expressed by real numbers reflecting a degree of satisfaction, a preference, or a degree of membership.

- the *Sugeno integral*, which can be viewed as the ordinal version of Choquet, i.e., the scores of the evaluation criteria are expressed on a finite ordinal (or qualitative) scale, and thus the Sugeno integral involves a combination of min-max logical operations on a fuzzy measure.

In this Section, we will only consider the use of Choquet integral for combining attack features, since most criteria we are dealing with are defined on cardinal scales (usually, on $[0, 1]$). When the analyst is confronted with multi-criteria problems where criteria are defined on both ordinal and cardinal scales, a commonly used approach consists in trying to turn the ordinal problem into a cardinal one, or to get cardinal information from the ordinal one, for example by counting the number of times an alternative is better or worse than the other ones on a given criterion. When all criteria are ordinal ones, then obviously, the Sugeno integral provides more appropriate means to aggregate them.

Recall that our attack attribution methodology is based on a *relational approach* where the goal is to combine a set of pairwise relations reflecting similarities among attack events w.r.t different attack features, so as to build a combined graph that takes into account all features (see Section 4.2). This combined graph can then serve to identify groups of events linked to the same root phenomenon. The purpose of this Section is to demonstrate how the Choquet integral may be used to perform this aggregation of features in a more effective way. We will also show that a Choquet integral offers a much greater flexibility in modeling interactions among features.

The rest of this Section is organized as follows: first, we introduce the concept of *fuzzy measure*. Then, we formally define the Choquet integral and we introduce a few essential concepts that are associated to it. Next, we illustrate the use of Choquet integral on a set

of attack events. Finally, we examine the problem of how to determine the coefficients of a fuzzy measure.

4.4.2 Fuzzy measures

Fuzzy integrals, such as Choquet or Sugeno integrals, are defined with respect to so-called *fuzzy measures*. Given a set of objects $\mathcal{N}$ (e.g., criteria, features), a fuzzy measure is simply a set function used to define, in some sense, the importance (or strength) of any subset belonging to the power set of $\mathcal{N}$. It is worth noting that fuzzy measures are not necessarily additive (i.e., the measure of a given set is not necessarily equal to the sum of its subsets). This property of additivity has been somehow relaxed on fuzzy measures by requiring only the *monotonicity* of the measure.

More formally, a fuzzy measure (alternatively called a *capacity* in the literature) is defined by

---

Definition 4.4. (Fuzzy measure or Capacity) *Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be the index set of n criteria. A* capacity [31] *or* fuzzy measure [153] *is a set function $v : 2^{\mathcal{N}} \to [0, 1]$ which is monotonic (i.e., $v(\mathcal{A}) \leq v(\mathcal{B})$ whenever $\mathcal{A} \subset \mathcal{B}$) and satisfies $v(\emptyset) = 0$. The measure is normalized if in addition $v(\mathcal{N}) = 1$.*

In multi-criteria decision making, a fuzzy measure is thus a set of 2^n real values where each value can be viewed as the degree of importance of a combination of criteria (also called a *coalition*, in particular in game theory). In other words, from definition 4.4, any subset $\mathcal{A} \subseteq \mathcal{N}$ can be considered as a coalition of criteria, and thus $v(\mathcal{A})$ reflects the importance of this coalition with a given weight. Note that when new elements are added to a given coalition, it can not decrease its weight (due to the monotonicity condition).

A basic example of a fuzzy measure is

$$v(\mathcal{A}) = \frac{|\mathcal{A}|}{n}, \tag{4.15}$$

for any subset $\mathcal{A} \subseteq \mathcal{N}$, and where $|\mathcal{A}|$ denotes the number of elements in $\mathcal{A}$.

Two mathematical properties of fuzzy measures are particularly of interest. First, a fuzzy measure is *additive* if for all disjoint subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, we have $v(\mathcal{A} \cup \mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B})$. That is, when a fuzzy measure is additive, it suffices to define the n coefficients $v(\{1\}), \ldots, v(\{n\})$ to define the fuzzy measure entirely. Note that in the general case, one needs to define the $2^n - 2$ coefficients corresponding to the 2^n subsets of $\mathcal{N}$, except $v(\emptyset)$ and $v(\mathcal{N})$.

Secondly, a fuzzy measure is *symmetric* if the value $v(\mathcal{A})$ depends only on the cardinality of the set $\mathcal{A}$, i.e., for any subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, $|\mathcal{A}| = |\mathcal{B}|$ implies $v(\mathcal{A}) = v(\mathcal{B})$. The example given by equation 4.15 here above is an example of a fuzzy measure which is both additive and symmetric. Note that this type of measure is usually too restrictive in the modeling of a multi-criteria aggregation (in fact, the integral of such fuzzy measures coincides with the arithmetic mean).

A fuzzy measure can also be transformed into an alternative representation, called the *Möbius representation*, which can be helpful in expressing various concepts or quantities related to aggregation functions. For example, we will see in section 4.4.4 that it is convenient to express certain interaction indices in a more compact form. The Möbius representation of a fuzzy measure can be obtained with the Möbius transform.

Definition 4.5. (Möbius transform) [130] *The* Möbius transform *of a fuzzy measure v, denoted by $\mathcal{M}_v$, is a set function defined for every $\mathcal{A} \subseteq \mathcal{N}$ as:*

$$\mathcal{M}_v(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{A}} (-1)^{|\mathcal{A} \setminus \mathcal{B}|} v(\mathcal{B})$$

The fuzzy measure v can be recovered using the inverse of a Möbius transform, called *Zeta transform*:

$$v(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{A}} \mathcal{M}_v(\mathcal{B}), \quad \forall \mathcal{A} \subseteq \mathcal{N}$$

In addition,

$$\mathcal{M}_v(\emptyset) = 0 \text{ and } \sum_{\mathcal{A} \subseteq \mathcal{N}} \mathcal{M}_v(\mathcal{A}) = 1.$$

Example of fuzzy measure.

A convenient way of representing fuzzy measures consists to use a *lattice form* (i.e., a Hasse diagram) of the inclusion relation defined on the set of subsets of $\mathcal{N}$ ([17]). For example, for $n = 3$, we can represent the 2^n elements of a fuzzy measure v as:

$$\begin{array}{ccc} & v(\{1,2,3\}) & \\ v(\{1,2\}) & v(\{1,3\}) & v(\{2,3\}) \\ v(\{1\}) & v(\{2\}) & v(\{3\}) \\ & v(\emptyset) & \end{array}$$

Let v be a fuzzy measure given by

$$\begin{array}{ccc} & 1 & \\ 0.9 & 0.5 & 0.3 \\ 0.5 & 0 & 0.3 \\ & 0 & \end{array}$$

Then, its Möbius transform $\mathcal{M}_v$ is given by

$$\begin{array}{ccc} & 0.1 & \\ 0.4 & -0.3 & 0 \\ 0.5 & 0 & 0.3 \\ & 0 & \end{array}$$

For example,

$$\begin{aligned}\mathcal{M}_v(\{1,2\}) &= (-1)\cdot v(\{1\}) + (-1)\cdot v(\{2\}) + (-1)^2 \cdot v(\{1,2\}) \\ &= -0.5 - 0 + 0.9 = 0.4\end{aligned}$$

Observe that the sum of all values in the Möbius representation is equal to 1, and the values of v and $\mathcal{M}_v$ coincide on singletons.

4.4.3 Choquet integral

In this section, we present the Choquet integral, which is defined with respect to a fuzzy measure.

Definition 4.6. (Choquet integral) [31] *The (discrete) Choquet integral of an input vector* $\mathbf{z}$ *with respect to a fuzzy measure (or capacity)* v *is given by*

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} z_{(i)} \left[v(\{j | z_j \geq z_{(i)}\}) - v(\{j | z_j \geq z_{(i+1)}\}) \right] \tag{4.16}$$

where $z_{(1)} \leq z_{(2)} \leq \ldots \leq z_{(n)}$*, i.e.,* $z_{(i)}$ *is the* i^{th} *largest component of the input vector* $\mathbf{z}$*.*

By rearranging the terms of the sum here above, the Choquet integral can alternatively be written as ([17]):

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} \left[z_{(i)} - z_{(i-1)} \right] v(H_i) \tag{4.17}$$

where $H_i = \{(i), \ldots, (n)\}$ is the subset of indices of the $n - i + 1$ largest components of $\mathbf{z}$, and $z_{(0)} = 0$ by convention.

For example, let $n = 3$ and $z_2 \leq z_1 \leq z_3$. Then, using equation 4.16, we have

$$C_v(z_1, z_2, z_3) = z_2 \left[v(\{2,1,3\}) - v(\{1,3\}) \right] + z_1 \left[v(\{1,3\}) - v(\{3\}) \right] + z_3 v(\{3\})$$

Special cases.

It is worth mentioning that the class of Choquet integrals generalizes averaging functions, such as those discussed previously. In fact, it turns out that weighted means and OWA functions are just special cases of Choquet integrals with respect to additive and symmetric fuzzy measures respectively. More precisely, when a fuzzy measure v is *additive*, the Choquet integral reduces to a weighted arithmetic mean:

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} v(\{i\})\, z_i$$

When a fuzzy measure v is *symmetric*, the Choquet integral reduces to an OWA function

as introduced in Section 4.3, with weights given by

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} (v_{n-i+1} - v_{n-i})\, z_{\nearrow(i)}$$

with $v_i := v(\mathcal{A})$, such that $|\mathcal{A}| = i$.

Similarly, it can be showed that the WOWA operator is also a special case of Choquet integral [173]. Finally, the Choquet integral with respect to a *symmetric additive* fuzzy measure (as the example 4.15) coincides with the arithmetic mean.

Orness and Entropy.

The general concept of *orness* defined by equation 4.7 can be calculated in the case of the Choquet integral thanks to following formula [89]:

$$orness(C_v) = \frac{1}{n-1} \sum_{\mathcal{A} \subseteq \mathcal{N}} \frac{n - |\mathcal{A}|}{|\mathcal{A}| + 1} \mathcal{M}_v(\mathcal{A}) \tag{4.18}$$

where $\mathcal{M}_v(\mathcal{A})$ is the Möbius representation of $\mathcal{A}$. Quite obviously, this quantity can also be expressed in terms of the fuzzy measure v. The signification of this index is the same as the one given in section 4.3.4 for the OWA aggregation function.

Then, similarly to the important concept of *weights dispersion* defined for OWA functions in equation 4.9, the *entropy* of a fuzzy measure v with respect to the Choquet integral has been defined in [90] as follows:

$$H(v) = \sum_{i \in \mathcal{N}} \sum_{\mathcal{A} \subseteq \mathcal{N} \setminus i} \frac{(n - |\mathcal{A}| - 1)!}{n!} h(v(\mathcal{A} \cup \{i\}) - v(\mathcal{A})) \tag{4.19}$$

with $h(t) = -t \log t$, if $t > 0$ and $h(0) = 0$. This definition coincides with the previous one given for OWA functions when v is symmetric. Again, the maximal value of H is $\log n$ and is achieved for an *additive symmetric* fuzzy measure (such as the one given in the example 4.15). The minimal value 0 is achieved for a Boolean fuzzy measure.

Some other mathematical properties of Choquet integrals have been extensively studied by several MCDA experts; however, we limit our discussion to those mathematical concepts that are really essential to understand the integration of such techniques into our attack attribution methodology. We refer the interested reader to [63, 173, 17] for a more detailed discussion of mathematical aspects.

Note also that Beliakov et al. discuss in [17] the *computational aspects* of various aggregation functions and fuzzy integrals, i.e., how those concepts can be represented and implemented in a programming language. For the purpose of the experiments carried out in this dissertation, the Choquet integral, and all related concepts described in the text, have been implemented under the form of MATLAB® functions.

4.4.4 Interaction indices among criteria

The flexibility of the Choquet integral comes also with a certain complexity, which is mainly due to the fact that a fuzzy measure v must be defined by 2^n values. As a result, the behavior of the decision-making process does not always appear as clearly when looking at all values of v. Moreover, in multi-criteria problems, it is often the case that certain criteria are not independent, i.e., there is some interaction (positive or negative) among the criteria. For example, two criteria may point essentially to the same concept, and hence should be considered as redundant in the aggregation. Therefore, it is interesting to define some indices to measure the importance of a given criterion, or the interactions among criteria.

To do this, we can use the *Shapley value*, which measures the importance of a criterion i in all possible coalitions. It was first proposed by Shapley [142] in the context of cooperative game theory.

Definition 4.7. (Shapley value [142]) *The Shapley index of a criterion $i \in \mathcal{N}$ w.r.t. a fuzzy measure v is given by*

$$\phi(i) = \sum_{\mathcal{A} \subseteq \mathcal{N} \setminus i} \frac{(n - |\mathcal{A}| - 1)!|\mathcal{A}|!}{n!} [v(\mathcal{A} \cup \{i\}) - v(\{\mathcal{A}\})]$$

The Shapley value *is the vector $\phi(v) = (\phi(1), \ldots, \phi(n))$.*

The Shapley value can be interpreted as the average contribution of each criterion alone in all possible coalitions. With the help of the Möbius transform, the Shapley index can be expressed in a more compact form, which can be also more convenient to calculate:

$$\phi(i) = \sum_{\mathcal{B} |\, i \in \mathcal{B}} \frac{1}{|\mathcal{B}|} \mathcal{M}_v(\mathcal{B})$$

Another important measure is the *interaction index*, introduced by Murofushi and Soneda [103], which quantifies the way two criteria i, j interact in all possible coalitions. As mentioned before, a certain criterion may be irrelevant when considered alone, but its importance regarding the overall decision value may sharply rise when taken in conjunction with other criteria.

Definition 4.8. (Interaction index [103]) *The interaction index between two criteria $i, j \in \mathcal{N}$ w.r.t. a fuzzy measure v is given by*

$$I_{ij} = \sum_{\mathcal{A} \subseteq \mathcal{N} \setminus \{i,j\}} \frac{(n - |\mathcal{A}| - 2)!|\mathcal{A}|!}{(n-1)!} [v(\mathcal{A} \cup i, j) - v(\mathcal{A} \cup i) - v(\mathcal{A} \cup j) + v(\mathcal{A})]$$

When $I_{ij} < 0$, we can say that criteria i, j are linked by a negative synergy (redundancy, or substitutability). Inversely, a positive interaction $I_{ij} > 0$ depicts a positive synergy (or complementarity) between criteria i, j ([58, 59]). When $I_{ij} = 0$, we say that criteria i, j

are independent, because the degree of satisfaction due to their combination is equivalent to the sum of the individual contributions of both criteria (i.e., it reflects a null synergy).

This definition of I_{ij} due to Murofushi and Soneda holds for a pair of criteria only, but it was extended by Grabisch et al. [63] for any coalition $\mathcal{A}$ (and not just pairs) of criteria:

$$I(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{N} \setminus \mathcal{A}} \frac{(n - |\mathcal{B}| - |\mathcal{A}|)!|\mathcal{B}|!}{(n - |\mathcal{A}| + 1)!} \sum_{\mathcal{C} \subseteq \mathcal{A}} (-1)^{|\mathcal{A} \setminus \mathcal{C}|} v(\mathcal{B} \cup \mathcal{C}) \tag{4.20}$$

which again can be expressed in a more compact form using the Möbius transform:

$$I(\mathcal{A}) = \sum_{\mathcal{B} |\, \mathcal{A} \subseteq \mathcal{B}} \frac{1}{|\mathcal{B}| - |\mathcal{A}| + 1} \mathcal{M}_v(\mathcal{B}) \tag{4.21}$$

Clearly, $I(\mathcal{A})$ coincides with I_{ij} for $\mathcal{A} = \{i, j\}$, and coincides with $\phi(i)$ for $\mathcal{A} = \{i\}$. As we show in the next section, those interaction indices present a special interest when the analyst must deal with the problem of defining fuzzy measures.

4.4.5 Illustration on attack events

(1) Definition of a fuzzy measure

Let us consider again the example data set made of attack events described in Section 4.3.3. Recall that we have 4 different criteria which are mapped to the selected attack features $F_{geo}, F_{sub}, F_{time}, F_{act}$.

To use a Choquet integral as aggregation function, the analyst needs to define a fuzzy measure v that models the requirements w.r.t. the behaviors of attack phenomena. As usual, the 2^n elements of v can be represented under a lattice form, which is of the following form when n=4:

$$\begin{array}{ccccccc} & & & v(\{1,2,3,4\}) & & & \\ & v(\{1,2,3\}) & v(\{1,2,4\}) & v(\{1,3,4\}) & v(\{2,3,4\}) & & \\ v(\{1,2\}) & v(\{1,3\}) & v(\{1,4\}) & v(\{2,3\}) & v(\{2,4\}) & v(\{3,4\}) & \\ & v(\{1\}) & v(\{2\}) & v(\{3\}) & v(\{4\}) & & \\ & & & v(\emptyset) & & & \end{array}$$

To define the numerical values for each coalition of a fuzzy measure v, we have to consider the requirements imposed by the domain knowledge (as described previously in Section 4.3.3). For the values of the singletons $v(\{i\})$, $i = 1, \ldots, 4$, we can start for example from the values of the weighting vector $\mathbf{p} = [0.15, 0.25, 0.35, 0.25]$, as defined previously for the WOWA operator. Recall that those values quantify the degree of reliability of each attack feature.

Next, for each coalition of criteria, we define the corresponding value of the fuzzy measure by considering the positive, negative or null effect of the interactions among all possible subsets of the coalition. In this case, we just need to reduce the value for each coalition of criteria involving a combination of the two redundant features F_{geo}, F_{sub} (i.e., $\{1, 2\}$, $\{1, 2, 3\}$ and $\{1, 2, 4\}$).

This leads us to define following fuzzy measure v:

			1		
	0.75	*0.65*	0.85	0.95	
0.25	0.60	0.50	0.70	0.60	0.70
	0.15	0.25	0.35	0.25	
			0		

(2) Evaluation of the Choquet integral on attack events

We are now in the position of calculating the Choquet integral C_v w.r.t. the fuzzy measure v defined here above. We consider again the same set of attack events as described in the previous illustration in Section 4.3.3.

Table 4.2: Aggregated scores obtained with Choquet integral applied to the same set of event pairs and criteria vectors as in 4.3.3 . The Choquet integral can help to further emphasize the gap between the desired and unwanted cases, thanks to its greater flexibility in modeling behaviors or preferences.

Row nr	Event pairs	F_{geo}	F_{sub}	F_{time}	F_{act}	M	$WOWA_{\mathbf{w},\mathbf{p}}$	C_v
1	(e_0, e_1)	1	1	1	0	0.75	0.80	0.75
2	(e_1, e_2)	0.4	0.2	1	1	0.65	0.74	**0.79**
3	$(e_1, e_2)'$	0.1	0.1	0.8	0.8	0.45	0.54	**0.59**
4	(e_2, e_3)	0.9	0.7	1	0	0.65	0.68	**0.69**
5	$(e_2, e_3)'$	0.1	0.9	1	0	0.50	0.61	**0.67**
6	(e_1, e_4)	0.9	0.3	0.2	0.3	0.45	0.33	0.36
7	$(e_1, e_4)'$	0.9	0.9	0.2	0.3	0.58	0.47	**0.41**
8	(e_3, e_4)	0.45	0.45	0.1	1	0.50	0.40	0.46
9	(e_3, e_5)	1	1	0	0	0.50	0.33	**0.25**

The results are given in Table 4.2 where we can compare the performance of C_v with the classical mean M and the WOWA operator (that has provided the best results until now). We observe that the Choquet integral outperforms any other aggregation function, which comes obviously from its greater flexibility in modeling more complex interactions (such as a synergy or a redundancy between two features). For example, on the cases given in rows 3 and 5 in Table 4.2, the Choquet integral helps to increase the global score of those cases (up to **0.59** and **0.67** respectively), despite the fact that the average value is equal or even less than 0.5. Similarly, on the cases given in rows 7 and 9, we can observe that the Choquet integral is able to lessen the undesired effect of the redundant features, so that the global score is minimized (down to **0.41** and **0.25** respectively) although the average value is equal or even greater than 0.5. Thanks to this flexibility, the Choquet integral can thus model more closely the requirements of the analyst regarding the behaviors of phenomena.

Finally, applying formula's 4.18 and 4.19, we find that the orness of this fuzzy measure v is equal to 0.55 while its entropy is equal to 1.25, which means that all inputs are well taken into account during the aggregation.

(3) Computing interaction indices

The concepts of interaction index defined in Section 4.4.4 can be illustrated by applying the definitions of $\phi(v)$ and $I(\mathcal{A})$ to the fuzzy measure v of previous example.

For $I(\mathcal{A})$, we obtain the following results :

		0.15			
	-0.125	**-0.125**	0.025	0.025	
-0.15	0.025	0.025	0.025	0.025	-0.05
	0.11	0.21	0.39	0.29	
		0.52			

As expected, the fuzzy measure has **negative** interaction indices for all coalitions involving the combination of the redundant features $\{1, 2\}$, which reflects the negative synergy between those criteria.

Finally, we can calculate the Shapley values for each attack feature:

$$\phi(v) = [0.1125,\ 0.2125,\ 0.3875,\ 0.2875]$$

which illustrates also the equivalence of $\phi(i)$ with $I(\mathcal{A})$ for each $\mathcal{A} = \{i\}$. This Shapley value shows that the second feature (F_{sub}) and the fourth feature (F_{act}) have on average approximately the same contribution to the global score, whereas F_{time} and F_{geo} have globally the largest and smallest influence respectively. This is clearly consistent with the requirements of the analyst described previously in Section 4.3.3.

4.4.6 Determination of fuzzy measures

The flexibility of the Choquet integral has also a major drawback, which is related to its exponential complexity (remember that $2^n - 2$ real values must be defined in a fuzzy measure). Another related consequence is the difficulty to interprete those values (certainly when $n \nearrow$), and thus to analyze the behaviour of the aggregation model. To deal with those issues, several particular families of measures have been proposed.

λ-fuzzy measures

Sugeno [153] has first proposed a simplified submodel based on λ-fuzzy measures as a way of reducing the complexity of a fuzzy measure[3]. The idea is to define the values of the fuzzy measure v only for individual criteria, and then to solve a linear system to determine all other values for the coalitions, based on some constraints imposing a sub- or superadditivity on the fuzzy measure.

Definition 4.9. (λ-fuzzy measure [153]) *Given a parameter $\lambda \in]-1, \infty[$, a λ-fuzzy measure is a fuzzy measure v that, for all disjoint sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, satisfies*

$$v(\mathcal{A} \cup \mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B}) + \lambda v(\mathcal{A}) v(\mathcal{B})$$

[3]For this reason, λ-fuzzy measures are also called Sugeno measures.

Under these conditions, all values $v(\mathcal{A})$ can be immediately calculated from the n independent values $v(\{i\})$, $i = 1, \ldots, n$, by using the following formula

$$v(\bigcup_{i=1}^{m}\{i\}) = \frac{1}{\lambda}\left(\prod_{i=1}^{m}(1+\lambda v(\{i\})) - 1\right),\ \lambda \neq 0 \tag{4.22}$$

where the coefficient λ is determined from the boundary condition $v(\mathcal{N}) = 1$, and involves solving following equation on $(-1, 0)$ or $(0, \infty)$

$$\lambda + 1 = \prod_{i=1}^{n}(1+\lambda v(\{i\})) \tag{4.23}$$

A λ-fuzzy measure is either sub- or superadditive, when $-1 < \lambda \leq 0$ or $\lambda \geq 0$ respectively. Note that a λ-fuzzy measure is an example of a distorted probability measure [106].

k-additive fuzzy measures

To decrease the exponential complexity of fuzzy measures, Grabisch proposed another submodel called k-order additive fuzzy measures, or shorter *k-additive* fuzzy measures [60]. The idea is to construct a fuzzy measure where the interaction among criteria is limited to groups of size k (or less). For example, in a 2-additive fuzzy measure, we can only model pairwise interactions among criteria, but no interactions in groups of 3 or more. In fact, all values of the fuzzy measure for groups of size larger than k are determined by various linear constraints.

k-additive fuzzy measures provide a good trade-off between complexity and flexibility of the model. Instead of $2^n - 2$ values, they require only $\sum_{i=1}^{k}\binom{n}{i}$ values to be defined. 1-additive fuzzy measures are just ordinary additive measures (for which only n values are needed), but they are usually too restrictive for an accurate representation of complex problems[4]. In practice, 2-additivity seems to be the best compromise between low complexity and richness of the model [60]. In this case, only $n(n+1)/2$ values need to be defined.

Definition 4.10. (k-additive fuzzy measure [60]) *A fuzzy measure v is said to be k-additive ($1 \leq k \leq n$) if its Möbius transform verifies*

$$\mathcal{M}_v(\mathcal{A}) = 0$$

for any subset $\mathcal{A}$ with more than k elements, $|\mathcal{A}| > k$, and there exists a subset $\mathcal{B}$ with k elements such that $\mathcal{M}_v(\mathcal{B}) \neq 0$.

A fundamental property of k-additive fuzzy measures is

$$\begin{cases} I(\mathcal{A}) = 0, \text{ for every} \mathcal{A} \subseteq \mathcal{N} \text{ such that } |\mathcal{A}| > k \\ I(\mathcal{A}) = \mathcal{M}_v(\mathcal{A}), \text{ for every} \mathcal{A} \subseteq \mathcal{N} \text{ such that } |\mathcal{A}| = k \end{cases}$$

[4] Recall that the Choquet integral w.r.t. 1-additive fuzzy measures is a weighted arithmetic mean.

Note there are a few other methods that have been developed for building models with reduced complexity, such as p-symmetric fuzzy measures [98] or k-tolerant and k-intolerant fuzzy measures [89]. However, we will limit our discussion to the two aforementioned methods, and we illustrate their application in the context of attack events here after.

Reducing the complexity of a fuzzy measure: a practical example.

Let us illustrate the use of the two methods explained here above to construct new *sub-models* with the help of fuzzy measures of reduced complexity.

First, to define a λ-fuzzy measure v_λ, we just need to define $v_\lambda(\mathcal{A})$ for every individual criterion $\mathcal{A} = \{i\}$ according to the *global behavior* we want to give to the measure. That is, if the analyst wants a subadditive (resp. superadditive) measure, then he has to define the values for the singletons such that $\sum_{i=1}^{n} v_\lambda(\{i\}) > 1$ (respectively $\sum_{i=1}^{n} v_\lambda(\{i\}) < 1$). The second characteristic that can influence the definition of the fuzzy measure v_λ is the relative ratio between each pair of criteria.

Turning back to our illustration on attack events, we need thus to define a subadditive fuzzy measure, due to the presence of the two redundant features F_{geo}, F_{sub}. Intuitively, starting from the values of weighting vector $\mathbf{p} = [0.15, 0.25, 0.35, 0.25]$, we have to increase the relative importance of criterion $\{3\}$ (and eventually also $\{4\}$), while decreasing the relative importance of the two redundant criteria $\{1, 2\}$. For example, this can lead to the definition of following values for the singletons: $\{0.10, 0.20, 0.50, 0.25\}$ (there exist obviously many other possibilities).

Then, we can solve the linear system given by 4.22 using those n independent values, and λ can be determined from equation 4.23, which gives us $\lambda = -0.14$. The solution to this linear system gives immediately all values of the new fuzzy measure v_λ, which is then defined by:

$$
\begin{array}{ccccccc}
 & & & 1 & & & \\
 & & 0.78 & 0.54 & 0.82 & 0.91 & \\
0.29 & & 0.59 & 0.35 & 0.69 & 0.44 & 0.73 \\
 & & 0.10 & 0.20 & 0.50 & 0.25 & \\
 & & & 0 & & &
\end{array}
$$

This λ-fuzzy measure can now be used to calculate the Choquet integral on the set of attack events introduced previously in the illustrations. The results are given in Table 4.3. As one can see, the results given by C_{v_λ} are in most cases as good (or even better) as the ones given by previous fuzzy measure v we had defined manually.

A second approach to reduce the complexity of a fuzzy measure is to use a k-additive measure. Suppose we want to define a 2-additive fuzzy measure called v_2. A 2-additive measure is apparently a good trade-off between complexity of the model (in terms of the number of values to determine) and its effectiveness. To do this, we can simply define the interaction indices $I_2(\mathcal{A})$ of v_2, for all combinations of 2 criteria or less. In other words, for all $\mathcal{A} \subseteq \mathcal{N}$ such that $|\mathcal{A}| > 2$, it suffices to set the values of $I_2(\mathcal{A}) = 0$ (by definition of a 2-additive measure).

Based on our domain knowledge about the considered attack features, let us define following interaction indices $I_2(\mathcal{A})$ for this new fuzzy measure v_2 (note the strong negative index for the interaction between the two redundant features, and a weak synergy for all other 2-coalitions):

			0		
	0	0	0	0	
-0.20	0.05	0.05	0.05	0.05	0
	0.10	0.20	0.40	0.30	
			0.50		

It is then possible to convert $I_2(\mathcal{A})$ to its corresponding fuzzy measure v_2 (which will be by definition 2-additive). To do this transformation $I_2(\mathcal{A}) \rightarrow v_2(\mathcal{S})$, we just need to use the conversion formula given by Grabisch in [61]:

$$v(\mathcal{S}) = \sum_{\mathcal{A} \subset \mathcal{N}} \beta^{|\mathcal{A}|}_{|\mathcal{S} \cap \mathcal{A}|} I(\mathcal{A})$$

where β is a quantity related to the Bernouilli numbers B_k, and is given by

$$\beta^l_k = \sum_{j=0}^{k} \binom{k}{j} B_{l-j}, \; k, l = 0, 1, 2, \ldots$$

The resulting 2-additive fuzzy measure v_2 obtained using this method is defined by the following values

			1		
	0.65	0.55	0.85	0.95	
0.20	0.55	0.45	0.65	0.55	0.60
	0.15	0.25	0.35	0.25	
			0		

The results of the Choquet integral w.r.t. v_2 (given in Table 4.3) demonstrate clearly that the 2-additive measure v_2 is at least as performant as the manually defined measure v (or even better in many cases). For example, C_{v_2} helps to emphasize the gap between *desired* and *unwanted* cases of events linkage, such as in rows 3,5 and 7,9 respectively. However, the analyst needs to define far less coefficients (only $\sum_{i=1}^{2} \binom{4}{i}$, instead of $(2^4 - 2)$ values). We can easily imagine how convenient 2-additive measures can be when $n \nearrow$. Moreover, we observe also that v_2 is at least as good as the λ-fuzzy measure v_λ. Although convenient, it is worth noting that λ-fuzzy measures are merely distorted probabilities, and it was showed that it can become too restrictive in certain MCDA problems [61].

Fitting fuzzy measures to empirical data

A last approach for determining fuzzy measures consists obviously in fitting values to empirical data. Similarly to the problem described in section 4.3.4, and assuming we

Table 4.3: Aggregated scores for the same set of event pairs and criteria vectors as in 4.3.3. Here, the values of Choquet integrals were calculated w.r.t. fuzzy measures with a reduced complexity, namely: (i) v_λ, obtained using a subadditive λ-fuzzy measure (with $\lambda = -0.14$), and (ii) v_2 obtained by defining a 2-additive fuzzy measure.

Row nr	Event pairs	F_{geo}	F_{sub}	F_{time}	F_{act}	M	C_v	C_{v_λ}	C_{v_2}
1	(e_0, e_1)	1	1	1	0	0.75	0.75	**0.78**	**0.65**
2	(e_1, e_2)	0.4	0.2	1	1	0.65	0.79	**0.80**	**0.73**
3	$(e_1, e_2)'$	0.1	0.1	0.8	0.8	0.45	0.59	**0.61**	**0.52**
4	(e_2, e_3)	0.9	0.7	1	0	0.65	0.69	**0.62**	**0.60**
5	$(e_2, e_3)'$	0.1	0.9	1	0	0.50	0.67	**0.68**	**0.62**
6	(e_1, e_4)	0.9	0.3	0.2	0.3	0.45	0.36	**0.32**	**0.34**
7	$(e_1, e_4)'$	0.9	0.9	0.2	0.3	0.58	0.41	**0.43**	**0.38**
8	(e_3, e_4)	0.45	0.45	0.1	1	0.50	0.46	**0.43**	**0.43**
9	(e_3, e_5)	1	1	0	0	0.50	0.25	**0.29**	**0.20**

are able to observe some phenomenon and collect a set of training samples $(\mathbf{z}_k, y_k)$, $k = 1, \ldots, K$, then it is possible to determine a fuzzy measure that best fits the empirical data according to some objective criterion.

Again, this can be formulated as an optimization problem. If we use a least squares criterion to minimize the error, we need to solve the following problem:

$$\min \sum_{k=1}^{K} \left(C_v(z_{1k}, \ldots, z_{nk}) - y_k\right)^2 \tag{4.24}$$

If we use a least absolute deviation criterion, the problem becomes

$$\min \sum_{k=1}^{K} \left|C_v(z_{1k}, \ldots, z_{nk}) - y_k\right| \tag{4.25}$$

From a mathematical viewpoint, these optimization problems are well-defined and can be solved quite easily using either linear or quadratic programming techniques, depending on the optimization criterion ([17]).

However, the problem of collecting meaningful training data remains. From a practical viewpoint, it is not so evident to gather sound experimental measurements that can be used as training data set, since we usually do not know the "ground truth" about the underlying phenomena. Even if we could validate the final clusters and the interconnections between them (corresponding to the behavior of certain phenomena obtained by our unsupervised method), it is still difficult to define precisely the individual values to be used as links in the different edge-weighted graphs. By lack of such training data set, we couldn't validate experimentally this supervised approach in the context of attack attribution, and thus we leave the study of this option as future work.

4.5 Summary

This Chapter has demonstrated how an intelligent combination of multiple attack features can effectively help a security analyst in the process of identifying attack phenomena, and perhaps more importantly, how it helps to model their dynamic behaviors. We have formally presented different methods to aggregate attack features using a *multi-criteria decision analysis* (MCDA) approach. In particular, we have extensively studied different aggregation methods, from rather basic (yet effective) ones, such as OWA functions, to more advanced methods like the *Choquet integral*. Our study has showed that a Choquet integral provides a greater flexibility in modeling the behaviors of attack phenomena by quantifying the influence of *coalitions of criteria*, e.g., a synergy or a redundancy between two or more features. By means of practical examples, we have illustrated how those various methods can be applied to a data set of security events, with the purpose of attributing attacks to the same root cause based upon a series of common features that do not have to be necessarily predetermined in a rigid manner.

In the next Chapter, we will apply this multi-criteria attribution method to an extensive data set of network attack traces collected during more than 2 years in the Internet by a set of distributed honeypot platforms. We illustrate the application of each step of the method to this specific data set, and we describe in detail some experimental results to demonstrate the kind of insights an analyst can obtain into the behaviors of malicious sources involved in Internet-scale attack phenomena.

Chapter 5

Application to Network Attack Traces

"The true method of knowledge is experiment."
– William Blake

This Chapter illustrates our *attack attribution* method with a first application on real-world attack traces collected by worldwide distributed honeypots during more than two years. As showed through the empirical results, the method could identify several large-scale phenomena composed of IP sources that are linked to the same root cause, which constitute a type of phenomenon that we have called *Misbehaving Cloud*.

We start by explaining how we have applied each step of the method to this specific data set. Then, we provide an in-depth analysis of several instances of misbehaving clouds to demonstrate the utility and meaningfulness of the approach, as well as the kind of insights we can get into the behaviors of malicious sources involved in these clouds.

5.1 Introduction

5.1.1 Honeynet

For the experiments presented in this Chapter, we have used a data set that is made of real-world attack traces. This unique data set has been collected on the Internet thanks to the *Leurré.com* Project ([86, 85]), a globally distributed honeynet. As a reminder, a *honeypot* is "a security resource whose value lies in being probed, attacked, or compromised" ([149]). Honeypots have no production value and hence should not see any legitimate traffic or network activity. Whatever they capture can thus be considered as malicious, or at least suspicious. By extension, a network of interconnected honeypots has been termed a *honeynet*.

The main objective of the Leurré.com project is to get a more realistic picture of certain classes of threats occurring in the Internet, by collecting unbiased quantitative data over a

long-term perspective. Since 2004, a set of identical, low-interaction honeypot platforms, based on *honeyd* [123], has been deployed in different countries and on various academic and industrial IP subnets. A platform runs three virtual honeypots, each one has its own public IP address and they emulate different operating systems (two Windows and one Linux machine) with various common services faking to be running on the machine (i.e., the ports 21, 23, 80, 137, 139, 445 are open on Windows honeypots, and ports 21, 22, 25, 80, 111, 514, 515 and 8080 are configured as open on Linux responders). The collected traffic, including the payloads of the packets, is automatically stored into a central database. The network traces are also enriched with contextual information (e.g., geographical location of the attackers, ISP's, domain names, P0f fingerprinting, etc).

More recently (in 2008), the second phase of the project was launched with the deployment of new honeypots with a higher degree of interaction. Those new sensors are based on the *ScriptGen* technology ([84, 83]). Thanks to novel protocol learning algorithms, ScriptGen honeypots are able to continue the network conversations with the attackers (usually, self-spreading malwares or bots) up to the point where the *code injection* attack is performed, which can lead in case of success to the execution of shellcode. Consequently, ScriptGen sensors can also collect malicious binaries that attackers want to propagate in an automated way. We refer the interested reader to [85, 122] for an in-depth presentation of the data collection infrastructure[1] .

5.1.2 Terminology

We start by introducing some important terms defined in the Leurré.com jargon, which will be used throughout the experimental part of this Chapter:

1. **Platform**: A physical machine running three virtual honeypots, which emulate three distinct machines thanks to *honeyd* ([123]). A platform is connected directly to the Internet and collects *tcpdump* traces that are gathered on a daily basis in a centralized database. Note that the term **sensor** is sometimes used interchangeably throughout this chapter.

2. **Source**: an IP address that has sent at least one packet to, at least, one platform. An IP address remains associated to a given Source as long as no more than 25 hours[2] elapse between two packets sent by that IP. After such a delay, the IP will be associated to a new source identifier if we observe it again.

3. **Attack**: refers to all packets exchanged between a malicious source and a platform.

4. **Cluster**: all the sources that execute the same attack against any of the platforms constitute a so-called *(attack) Cluster*. In practice, such a cluster groups all malicious sources that have left highly similar network traces on our platforms. How to identify clusters and how those clusters look like are issues that have been explained in [121, 85].

[1] The Leurré.com dataset is publicly available for any researcher under the condition of a Non-Disclosure Agreement that aims at protecting the privacy of the partners involved in the deployment of those honeypot platforms.

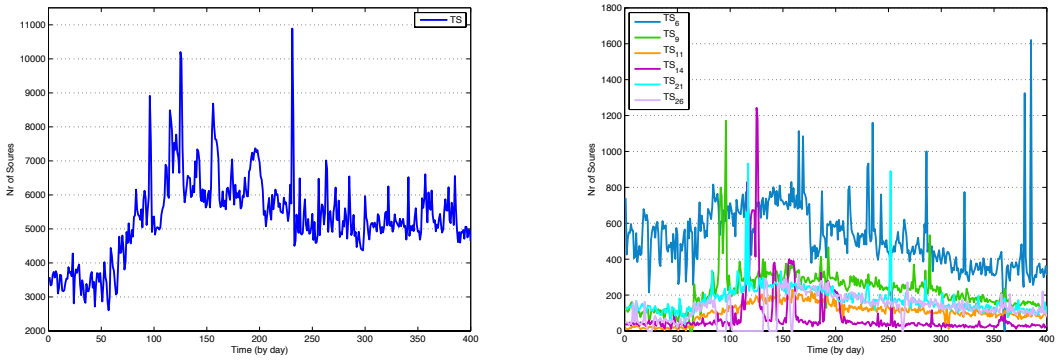
[2] By grouping packets by originating sources instead of by IPs, we minimize the risk of mixing together the activities of two distinct physical machines (as a side effect of the dynamic address allocation implemented by ISP's).

5.1.3 Experimental dataset

Machines used in the Leurré.com project are maintained by partners all over the world, on a voluntary basis. Some of these platforms can thus become unavailable. For this dissertation, we have selected a subset of 40 stable platforms from all platforms at our disposal (i.e., machines that have not been down more than 10 times over the whole period, and each of them has been up continuously for at least 100 days). The platforms are located in 16 different countries and belong to 22 different class A-subnets.

Our analysis period spans from Sep 2006 until November 2008, i.e., 800 days of monitoring. A total of 3,477,976 attacks have been observed by those platforms over this observation period. We can represent the total number of attacks observed on a daily basis as a time series denoted by $TS = \{TS(x)\}$, where $x = 1, \ldots, 800$ represents the observation day (see Fig. 5.1(a)). To refine the analysis, we can represent the number of attacks observed on each platform. This leads to the definition of 40 distinct attack time series denoted by TS_{p_j} where p_j represents a platform identifier (see Fig. 5.1(b)).

We can go further in splitting our time series in order to represent which type of attack was observed on which platform. So, we split each TS_{p_j} into as many time series as there are *attack clusters*, as defined before. These newly obtained time series are represented by $TS_{c_i,p_j} \forall$ cluster c_i and $\forall$ platform p_j. That is, a given point $TS_{c_i,p_j}(x)$ represents the amount of sources attacking, on day x, the platform p_j by means of the attack defined by the cluster identifier c_i. Note that we have a total of 395,712 time series TS_{c_i,p_j}.



(a) (b)

Figure 5.1: Illustration of (a) the global time series TS of the honeynet data set, with the total number of attacking sources grouped by day; (b) some attack time series TS_{p_j} composing TS, as observed by 6 (out of 40) platforms individually. For the sake of clarity, only 400 days are represented on these graphs.

In [118], it has been shown that a large fraction of these time series barely vary in amplitude on a daily basis. This continuous, low-intensity activity is also referred to as the Internet *background radiation* [114]. In this experiment, we do not consider those flat curves, and we instead focus on time series that show some significant variations over time (for at least several days), indicating the existence of some ephemeral phenomenon. To automatically identify these time series of interest, we have applied the method presented in [118], which finally gives our experimental data set denoted by $\mathcal{D}$ that contains now only 2,127 distinct time series of attacks. However, $\mathcal{D}$ still accounts for a total of 2,538,922

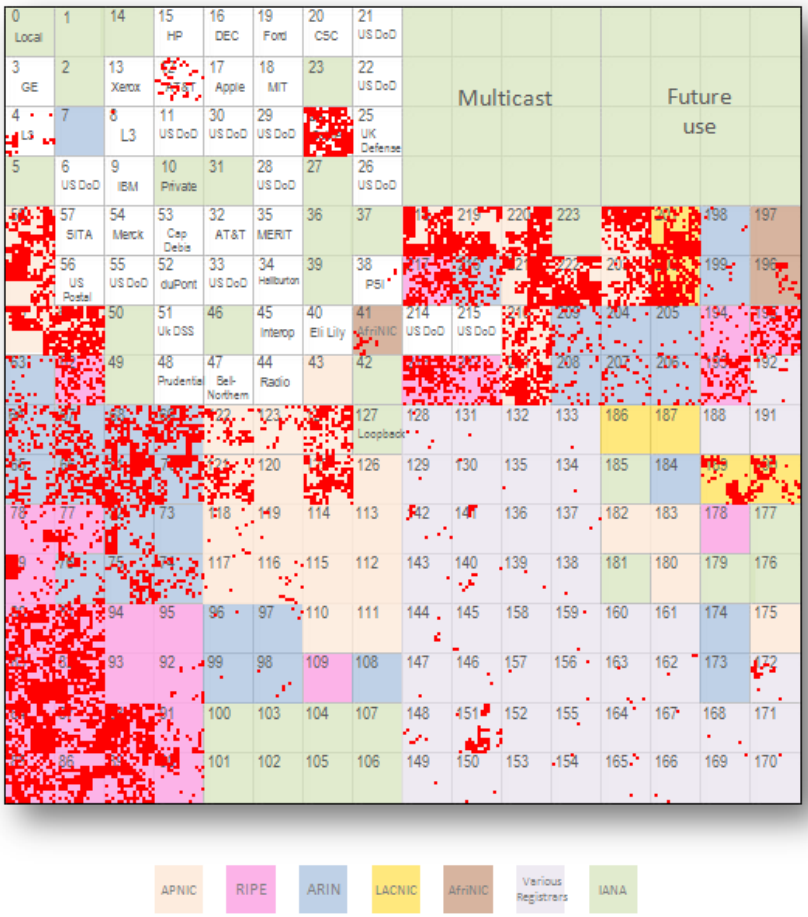


Figure 5.2: Visualization of malicious sources in the IPv4 space for an experimental data set collected by 40 platforms of the *Leurré.com* Project from Sep 2006 until Nov 2008. Backscatter traffic and spoofed IP sources have been filtered out in this map.

malicious sources, whose network activity has been grouped according to 320 different types of attack clusters c_i. Fig. 5.2 provides a visualization of all those malicious sources under the form of an IP map showing the entire IPv4 space[3].

Using a fractal mapping, each square in Fig. 5.2 represents a whole Class A-subnet (i.e., a $x/8$ accounting for 2^{24} addresses), and each red pixel represents a Class B-subnet from which at least one malicious source has been observed by the honeypots. As we can observe, the spatial distribution of attacking machines is highly uneven, and there seems to be some IP networks that are largely infested with compromised machines. Almost every region in the world is involved in the generation of this malicious traffic, although the greatest part of the traffic seems to come obviously from European IP networks (RIPE), North-American networks (ARIN) and Asian networks (APNIC). Observe also the largely infected IP block 24.0.0.0, which is owned by a Canadian cable operator.

Note that, while illustrative, this type of global visualization does not provide any

[3]This type of layout was first proposed by Randall Munroe with a hand-drawn map of allocated Internet address blocks published on http://www.xkcd.com/195/

information about *which phenomena* (e.g., how many worms, botnets, spammers, etc) could be responsible for all this unsolicited traffic. Moreover, we still have no insights into the behaviors of those large-scale attack phenomena and how they evolve over time.

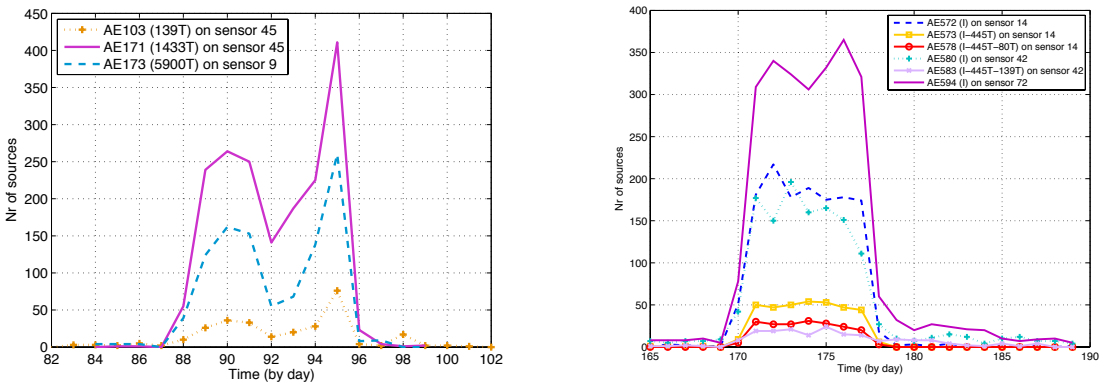
Dataset preprocessing: identification of attack events

In most data mining (DM) applications, there is a preliminary preprocessing step which is needed to present the raw data set under a form that is more usable by the DM algorithms. This is no different for honeypot attack traces. As a result, we have applied signal processing techniques on our set of time series contained in $\mathcal{D}$ in order to identify so-called *attack events*, which are defined at two different levels of aggregation:

Definition (μ-event): A *micro attack event* (or μ-event) is defined by a tuple $(\mathcal{T}_{x_1,x_2}, TS_{c_i,p_j})$ where $\mathcal{T}_{x_1,x_2}$ represents a limited period of time (typically a few days), starting on day x_1 and finishing at x_2, during which a *significant* attack activity is observed, and TS_{c_i,p_j} represents the time series corresponding to cluster c_i observed on the platform p_j.

Definition ($\mathcal{M}$-event): A set of micro attack events observed over the same period of time, and during which the corresponding time series are strongly correlated is defined as a *macro attack event* (or $\mathcal{M}$-event).

In other words, a μ-event represents a given segment of a time series TS_{c_i,p_j} that corresponds to an intense period of activity observed on a single platform p_j, and where the involved attacking sources have an activity profile defined by c_i. When different μ-events are coordinated in time and observed on different platforms, then we identify those events as being part of the same $\mathcal{M}$-event. Figure 5.3 illustrates this concept by representing two different $\mathcal{M}$-events composed of a certain number of μ-events that are correlated in the same time interval.



(a) (b)

Figure 5.3: Illustration of two different $\mathcal{M}$-events (a) an event made of 3 μ-events that are correlated on 2 different sensors, and are targeting 3 different ports (observed in April 2008); (b) an event composed of 6 μ-events that are correlated on 3 different sensors, and are targeting 4 different port sequences (observed in January 2007.

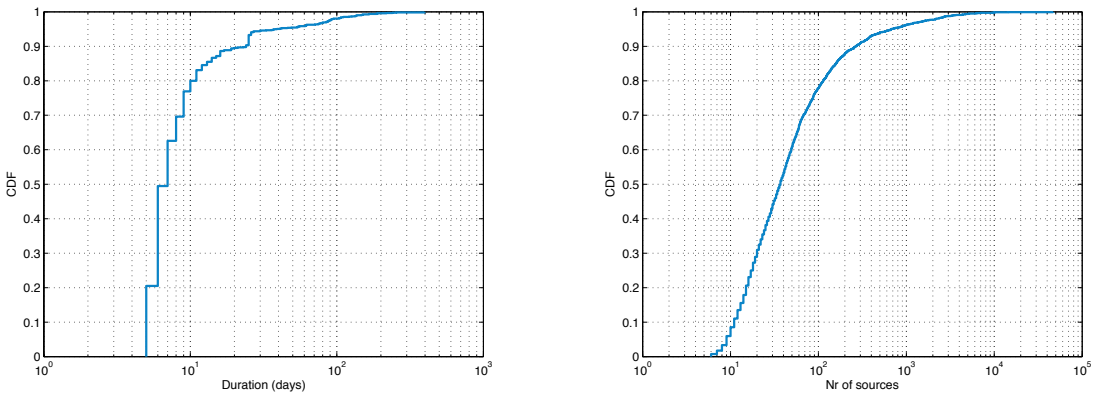
The identification of μ-events relies mostly on some well-known signal processing techniques. The goal is to segment the time series into periods of interest. Such periods are characterized by some *intense period* of activities isolated by periods of very stable or non existent activities. Several techniques exist to detect abrupt changes in a signal [14]. For this data set, the method used is the one that has been precisely presented in [117]. Then, from the set of μ-events of interest, we identify all those that are strongly correlated over the same period of time, which form thus a $\mathcal{M}$-event. Here too, we used the results that have been presented and discussed in [118, 117].

Consequently, our data set $\mathcal{D}$ has been split into 2,454 μ-events, which have been extracted automatically from 2,127 distinct time series of attacks. Table 5.1 provides an overview of the characteristics of both datasets. Fig. 5.4 shows the cumulative distributions of the duration Δt_μ (in days) and the size (in terms of number of sources) for the complete set of μ-events contained in the data set. As we can see, most events are rather ephemeral since 80% of them have a duration of 10 days or less. Regarding the volume of attacking sources, about 90% of the μ-events comprise less than 300 sources. However, about 5% of the events comprise more than 1,000 sources.

More detailed statistics on the traffic collected by *Leurré.com* platforms (such as distributions of protocols, port sequences, traffic share, top attacking countries, most targeted platforms, etc) have been published in [85] for the monitoring period spanning from 2004 until 2008.

Table 5.1: High-level characteristics of the honeynet data set $\mathcal{D}$, which was preprocessed to provide 2,454 attack events. c_i and p_j refer to the attack cluster and the honeypot platform respectively (as defined on page 100). $\overline{\Delta t_\mu}$ refers to the mean duration of the μ-events (in days) and $\overline{\text{size}_\mu}$ represents their mean size (in nr of sources).

Sources	**μ-events**	**$\mathcal{M}$-events**	c_i	p_j	$\overline{\Delta t_\mu}$	$\overline{\textbf{size}_\mu}$
2,538,922	2,454	691	320	40	13.4	226



(a) (b)

Figure 5.4: Cumulative distributions of (a) the duration Δt_μ (in days); and (b) the size; for the 2,454 μ-events contained in the honeynet data set (the x-axis is set to logarithmic scale).

5.2 Selection of attack features

Following the very first step of our attribution method, we need to define some relevant attack features that may reveal interesting patterns and bring meaningful evidence, so as to attribute different events to the same root phenomenon. In this Section, we describe the attack features we have considered as useful to analyze the global phenomena observed by the sensors.

Note, however, that we do not pretend that these are the only ones to be used in network (or threat) monitoring. Other features might certainly prove as relevant in the future, and for this reason, as we have showed in the previous Chapters, our attribution method offers a great flexibility regarding this selection of features, and it is built in such a way that additional features can be easily included if necessary.

5.2.1 Spatial distributions of attackers

Some key features we have selected for this experimental validation deal with certain *external characteristics* of malicious sources, namely their spatial distributions in terms of countries (denoted by F_{geo}) and IP subnets (denoted by F_{sub}). Looking at these statistical characteristics may reveal attack activities having specific distributions of originating countries or IP networks. The information provided by F_{geo} can be important to identify, for instance, botnets that are located in a limited number of countries. It is also a way to confirm the existence, or not, of so-called safe harbors for cybercriminals or hackers.

The source IP network is a property that nicely complements the geolocation. Instead of giving insight into possible geostrategic decisions made by the hackers, they can typically reveal some strategies in the propagation model of the malwares. Moreover, distributions of IP subnets can give a good indication of the spatial "uncleanliness" of certain networks, i.e., the tendency for compromised hosts (e.g., zombie machines) to stay clustered within unclean networks ([32]). Previous studies have also demonstrated that certain worms show a clear bias in their propagation scheme, such as a tendency to scan machines of the same (or nearby) network, in order to optimize their propagation [29].

So, for each μ-event, we create a feature vector representing either the distribution of originating countries, or of IP addresses (grouped by their Class A-prefix, to limit the vector's size). An example of geographical distribution for F_{geo} has been given in Chapter 3 (on page 39). Regarding F_{sub}, we just have to create, for each μ-event, a vector that represents the distribution of IP addresses grouped by Class A subnet (or grouped by /8, which means grouped by the first byte of the IP address). An illustration of such an IP subnet distribution for a given μ-event is showed in Fig. 5.5. Alternatively, the very same feature vector can also be represented under the form of relative frequencies, e.g.: 222(17%), 221(10%), 60(9%), 121(7%), 58(5%), 218(4%), others(47%).

As previously explained in Chapter 4, the two features F_{geo} and F_{sub} are not really independent of each other, since there is a direct mapping between IP networks and geographical countries where local ISP's are usually responsible for the management of well-defined IP blocks that have been assigned to them. Thus, those two features can be viewed as two different ways of measuring the same contextual characteristic (i.e., the origins of the attackers). Consequently, the two features are somehow *redundant* and two events having the same distributions of origins should exhibit high similarities for both features

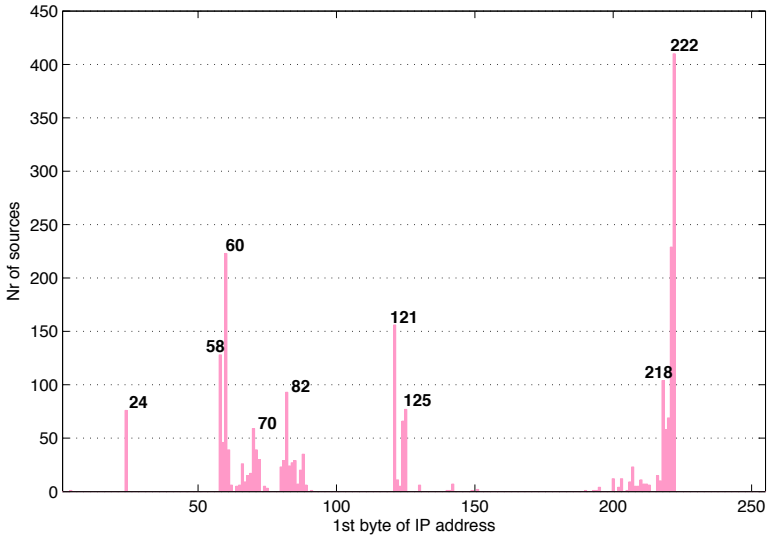


Figure 5.5: Illustrative example of feature vector for F_{sub}. The vector represents the IP subnet distribution of attacking machines for a given μ-event (e_{590}) observed by a sensor located in Serbia in the period Jan-Feb 2007, and involving a total of 2,364 IP sources.

(although not necessarily with the same amplitude). However, we will consider F_{sub} as a more reliable measurement than F_{geo}, for the very same reasons as those explained on page 75 (i.e., various country sizes and significant differences in network connectivity between countries).

5.2.2 Temporal correlation on the sensors

Next, we have selected an attack feature, denoted by F_{time}, that is related to the way malicious sources are targeting the *monitoring platforms*. For this purpose, we take advantage of the events identification technique described previously (see Fig.5.3 on page 103), which comes from the signal processing method developed in [117]. In fact, botmasters can send commands at a given time to a set of zombie machines to instruct them to scan (or attack) one or several IP subnets, which creates time-correlated attack events on specific sensors due to the coordination among those bots that belong to the same botnet.

This *degree of coordination* seems to be an important attack feature. Moreover, this feature does not require the creation of a new feature vector because this information can naturally be obtained from the result of the events identification technique, since (by definition) $\mathcal{M}$-events are composed of μ-events that are strongly correlated in time (see Fig. 5.3 for some examples of temporal patterns). However, due to the very large number of compromised machines in the Internet, F_{time} is probably not sufficient *by itself* to link two events to the same root phenomenon, as different botnets or worms can target the very same sensors in the same interval of time (with roughly the same daily pattern). Regarding the reliability of F_{time}, this attack feature still provides a more reliable measurement of the likelihood that two μ-events could be linked to the same phenomenon.

5.2.3 Type of activity

Besides the origins and the temporal coordination on the targeted sensors, the *type of activity* performed by the attackers seems also relevant. Indeed, bot software is often crafted with a certain number of available exploits targeting a given set of TCP or UDP ports. In other words, we might think of each bot having its own *attack capability*, which means that a botmaster will normally issue scan or attack commands only for vulnerabilities that he might exploit to expand his botnet (i.e., exploits that are implemented in the bot source code).

As a consequence, it seems to make sense to take advantage of similarities between *sequences of ports* that have been probed or attacked by malicious sources. Again, this attack feature by itself is definitively not sufficient to attribute two attack events to the same root cause, considering the fact that certain TCP or UDP ports are *a priori* more likely to be targeted or exploited than other unusual ports (for which no vulnerabilities have been found in potential services running on those ports). However, the "reliability" of the measurement regarding this feature is considered as normal.

We create thus a new set of feature vectors that are related to the port sequences targeted by malicious sources involved in μ-events, and we denote this new attack feature by F_{ps}. The information about which ports have been sequentially probed or exploited by every source is directly provided by the low-level network classification performed in the *Leurré.com Project* (which results in the creation of the so-called attack clusters c_i defined on page 100). This operation is done on a daily basis by means of automated scripts which are used for collecting, classifying and inserting new attack traces into the central database. Some examples of feature vectors for F_{ps} are given in Table 5.2.

Table 5.2: Examples of feature vectors (i.e., port sequences) for some μ-events, as defined with respect to F_{ps}. In the components of the port sequences, each number refers to the targeted port and the letter refers to the protocol (**T** = TCP, **U** = UDP, **I** = ICMP).

μ-event	c_i	Port sequence
e_{185}	155552	|1026**U**|1027**U**|1028**U**|1026**U**|1027**U**|1028**U**
e_{291}	17470	|1026**U**
e_{214}	34594	|5554**T**|9898**T**
e_{316}	75851	|**I**|445**T**|139**T**|445**T**|139**T**|445**T**
e_{353}	147436	|**I**|445**T**|80**T**
e_{394}	17718	|**I**|445**T**
e_{488}	175309	|2967**T**
e_{842}	14647	|445**T**
e_{848}	60231	|5900**T**
e_{1347}	15715	|1433**T**

5.2.4 Common IP addresses

Finally, we have decided to compute, for each pair of μ-events, the ratio of common IP addresses. We denote this attack feature by F_{cip}. We are aware of the fact that, as time passes, certain zombie machines of a given botnet might be cured while others may get

infected and join the botnet. Additionally, certain ISPs implement a quite dynamic policy regarding the allocation of IP addresses to residential users, which means that bot-infected machines can have different IP addresses when we observe them at different moments.

Nevertheless, considering the huge size of the IPv4 space, it is still reasonable to expect that two attack events are likely to be linked to the same root phenomenon when those events share a high percentage of IP addresses. Obviously, F_{cip} is not a new independent feature, since it is probably related to F_{geo} and F_{sub}, which introduces again a sort of redundancy among those features. However, from our own experience in analyzing manually many attack events collected by honeypots, we may consider that F_{cip} is even more reliable than F_{sub} (i.e., because of the vastness of the IPv4 space, sharing a high percentage of IP addresses is very unlikely to be due to chance only).

As feature vectors for F_{cip}, we could of course simply create, for each μ-event, the set of distinct IP addresses of all attacking sources, and then use a set function to measure the intersection between two sets. However, since certain events may comprise thousands of IP addresses, it is more efficient to compute directly the ratio of common IP addresses between all pairs of μ-events by running custom SQL queries on the *Leurré.com* database.

Summarizing, the complete set of attack features $\mathcal{F}$ that we consider for the experiments is:

$$\mathcal{F} = \{F_{geo}, F_{sub}, F_{time}, F_{ps}, F_{cip}\}$$

where

$$\begin{cases} F_{geo} = \text{geolocation, as a result of mapping IP addresses to countries;} \\ F_{sub} = \text{distribution of sources IP addresses (grouped by Class A-subnet);} \\ F_{time} = \text{degree of temporal coordination on the targeted platforms;} \\ F_{ps} = \text{port sequences probed or exploited by malicious sources;} \\ F_{cip} = \text{feature representing the ratio of common IP addresses among sources;} \end{cases}$$

After a preliminary analysis of these attack features, we consider the following preference relationships:

$$F_{geo} \prec F_{sub} \sim F_{ps} \prec F_{time} \sim F_{cip}$$

where $a \prec b$ indicates a preference relationship of b over a, in terms of reliability of the measurement provided by those features.

5.2.5 Other possible features

In this section, we suggest some additional features that could be used either in the clustering process, or as additional descriptive means to assist the root cause analysis. Examples of such features may include:

- proximities in IP addresses, between attackers' addresses and platform's address (eg, bias in the propagation vector)

- temporal patterns aggregated at different scales: hourly patterns, distributions by days of the week, etc

- other characteristics related to malicious sources: OS and hostnames (note: domain names do not seem to bring something meaningful)
- distributions of ISP's or ASN's, although this is closely related to the IP blocks that we have already used
- if we use medium-interaction honeypots (e.g., SGNET [83, 82]): we could add exploit (ε, π), and malware (μ) information to the graph-based clustering.

We leave the exploration of those additional features and data sets as future work.

5.3 Graph-based clustering

We now turn to the application of the second component of our attribution technique, i.e., the *graph-based clustering* component. Recall that this clustering is applied iteratively to every attack feature on the complete set of attack events. Since the dominant set framework is based on a pairwise clustering approach, we need first to calculate all pairwise distances with an appropriate metric for each feature.

In this Section, we start by describing which distances we have used, and how we have transformed those distances into similarities. For certain types of distances (e.g., statistical divergences), we also present a calibration procedure for doing this transformation in a meaningful way. Finally, we present the experimental results obtained from applying the graph-based clustering technique to our honeynet data set comprising 2,454 μ-events.

5.3.1 Distance metrics for measuring pattern proximities

In Section 5.2, we have seen that feature vectors created for F_{geo} and F_{sub} are spatial distributions. As described in Chapter 3, we need to use statistical distance functions to compare such vectors in a meaningful way. In Section 3.3, we have also illustrated the use of different metrics (or divergences), such as Kullback-Leibler, Jensen-Shannon and Bhattacharyya. Drawn from our own experience, we have used here Jensen-Shannon for the better results it has provided in comparing geographical and IP subnet distributions. How to transform those divergences into similarity scores is described in the next subsection.

Measuring pairwise similarities for features representing sets of objects (like F_{ps} and F_{cip}) is more straightforward, since in those cases we can use simple distance metrics, such as the *Jaccard similarity* coefficient. Let S_1 and S_2 be two sample sets, then the Jaccard coefficent is defined as the size of the intersection divided by the size of the union of the sample sets, i.e.:

$$s(i,j) = \frac{|S_1 \bigcap S_2|}{|S_1 \bigcup S_2|} \tag{5.1}$$

For F_{cip}, we have used the Jaccard similarity coefficient to compute the ratio of common IP addresses between attack events.

For F_{ps}, we have combined two different metrics. The first one is the Jaccard similarity between the two sets of targeted ports. However, not only the sets of ports have a semantic

meaning, but also the order (or the sequence) in which ports have been targeted. So, to take this aspect into account, we have also computed the longest sequence match between two targeted ports sequences. Then, we just take the average between the two metrics.

For example, let us consider the following two port sequences:

$$\begin{aligned} S_1 &= |\mathbf{I}|445\mathbf{T} \\ S_2 &= |\mathbf{I}|445\mathbf{T}|139\mathbf{T}|445\mathbf{T}|139\mathbf{T}|445\mathbf{T} \end{aligned}$$

They are quite similar in terms of targeted ports, but for some reason the ports sequence is much shorter for S_1 (which can indicate a malware that has stopped attacking because of a failed exploit). The Jaccard coefficient between S_1 and S_2 is 2/3; but the longest match between the two sequences is only 2/6. So, the combined similarity is here equal to:

$$s(S_1, S_2) = 0.5 \cdot \frac{2}{3} + 0.5 \cdot \frac{2}{6} = 0.5$$

Regarding F_{time}, we use a simple weighted means to combine three scores when comparing a pair of μ-events:

(i) a score $\in \{0, 1\}$ indicating whether two μ-events belong to the same $\mathcal{M}$-event (which means the events are coordinated in time);

(ii) a score $\in \{0, 1\}$ given by the simple comparison of the platforms that have been targeted by the two μ-events (1 = the events have been observed by the same platform);

(iii) a score $\in \{0, 1\}$ given by the simple comparison of the IP subnets (Class A) that have been targeted by the two μ-events (1 = the events have been observed in the same Class A-subnet);

Obviously, the most important score is the first one (time correlation). Then, observing two μ-events on the same platform or in the same subnet could mean something, but not necessarily. It is only the combination of those different features that are, in our opinion, really meaningful.

Based on our own experience, this leads us to define the following weighting vector to combine the three scores described here above: [0.6, 0.2, 0.2]. As an illustration, let us consider some typical examples:

- two time-correlated μ-events, observed on the same platform, yield a similarity of 1;

- two time-correlated μ-events, observed on different platforms but still in the same subnet, yield a similarity of 0.8;

- two μ-events having no time correlation, but observed on the same platforms, yield a similarity of 0.4.

Important remark

As a side note, one could wonder why we use such an *ad-hoc* combination, instead of using three separate features (e.g., F_{time}, $F_{platform}$, F_{classA}) and let them aggregate using the more general techniques described in the previous Chapter. The reason is that $F_{platform}$ and F_{classA} represent, by themselves, very weak features of the attack events. When

considered separately, those features have, in fact, no particular meaning. It is very usual to observe different attack events (having nothing to do with one another) targeting the very same platform at different points in time ($F_{platform}$), and it is even more likely to see this in the same Class A-subnet (F_{classA}).

Introducing many weak features has thus a negative impact on the aggregation process, as the expected behavior becomes harder to model. For instance, in this case a large number of attack events are likely correlated by $F_{platform}$, F_{classA}, and most probably also by F_{ps} since certain ports, like the ports used for Windows network sharing, are commonly used as a target. However, those three features together are not sufficiently discriminant to identify phenomena reliably. Consequently, the analyst has to define an OWA aggregation function that models a behavior where at least four attack features must be correlated. However, such an aggregation will fail to identify phenomena characterized by only three stronger features, such as F_{geo}, F_{sub} and F_{time}.

As a result, we observe that mixing many strong and weak features introduces a sort of *imbalance* in the aggregation process, which makes it harder to model (perhaps unnecessarily). In this example, using a simple OWA operator is automatically forbidden, as it is not flexible enough to model the interactions among several strong and weak features. The analyst must then rely on more complex aggregation schemes, e.g. by using a Choquet integral. However, we have seen that this kind of aggregation function comes also with an exponential complexity related to the definition of the fuzzy measure composed of 2^n coefficients. Furthermore, they are computationally more expensive.

In conclusion, we observe that it is usually better, whenever possible, to define attack features that are more representative of the phenomena under study. This enables us to use simpler aggregation functions such as weighted means, OWA or Weighted OWA operators, which are, generally speaking, more efficient to compute.

5.3.2 On mapping distances to similarities

When we rely on a distance function to measure a dissimilarity between pairs of events, it is still necessary to transform those measures into similarities, so it can be used by the clustering algorithm. Probably the most straightforward way to do this transformation is actually to retrieve the distance d from the value 1 (or eventually another constant), i.e.: $s = 1 - d$.

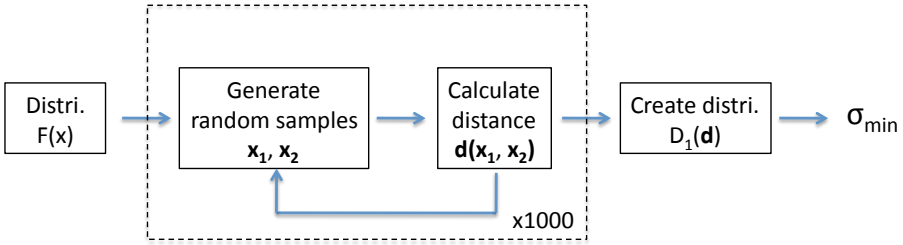
However, this can only work when the distance function is linear on the considered interval, e.g. $[0, 1]$. If we use statistical measures (e.g., statistical divergences, like for F_{geo} and F_{sub}), this is obviously not the case. Previous studies found that the similarity between stimuli decay exponentially with some power of the perceptual measure distance ([143]). Consequently, it is customary to use the following functional form to do this transformation:

$$s_{ij} = \exp(\frac{-{d_{ij}}^2}{\sigma^2}) \tag{5.2}$$

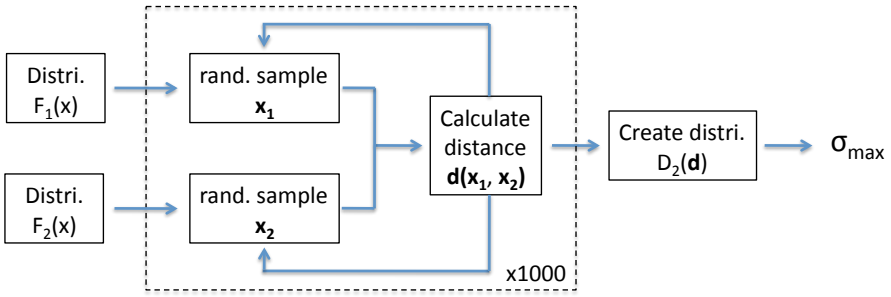
where σ is a positive real number which affects the decreasing rate of s.

Obviously, the question on how to determine this quantity σ remains. In this Section, we propose our own calibration procedure to determine some appropriate ranges of values for σ, according to the properties of the data set and the expected behavior of the similarity function.

The idea of this calibration procedure is as follows. First, we consider an empirical distribution $F(x)$ as typical pattern of our data set (for example, the average of all distributions in the data set). If we generate from $F(x)$ two random samples $\mathbf{x}_1, \mathbf{x}_2$ (using a uniform random distribution over all possible categories of $F(x)$), and we measure the pairwise distance between them (using the statistical divergence that we want to calibrate), then this distance d should be as low as possible, since both samples come from the very same distribution. Conversely, if we create two random samples $\mathbf{x}_1, \mathbf{x}_2$ from two different distributions $F_1(x), F_2(x)$ (which have to be chosen by the analyst), then the distance d between those samples should be as high as possible.



(a) Calibration test 1: determining σ_{min}



(b) Calibration test 2: determining σ_{max}

Figure 5.6: Calibration procedure of a distance metric to be transformed into similarity measure. (a) Statistical test with random samples generated from the same typical distribution; (b) Statistical test with random samples generated from two slightly different distributions that the analyst still wants to differentiate.

Fig. 5.6 illustrates the different steps of this calibration procedure. In the first test (Fig. 5.6 (a)), we repeat a large number of times the measurement of the distance d between two identical distributions randomly created from the same $F(x)$. Then, we compute the distribution of all distances, that we denote by $D_1(d)$. Similarly, in the second test (Fig. 5.6 (b)), we measure the distance between two different distributions (randomly created from two empirical distributions considered as different), and we compute the distributions of those distances that we denote by $D_2(d)$.

Based on $D_1(d)$, we can now determine a value for σ_{min}. We consider the value of d that lies in the upper quantile of D_1, since those values are the most unfavorable ones to the comparison of two identical distributions. For this "worst-case" value, we still require that the similarity is above a given minimum value, typically above 0.5. Then, we can determine σ_{min} by transforming equation 5.2 to:

$$\sigma_{min} = \sqrt{\frac{-d_{max}^2}{\log(s_{min})}} \tag{5.3}$$

with $d_{max} = \text{quantile}(D_1, 0.95)$ and $s_{min} = 0.5$ (and log(x) is the natural logarithm). In other words, when $\sigma \geq \sigma_{min}$, the similarity value $s \geq s_{min}$ for at least 95% of the cases.

Quite similarly, we can determine σ_{max} using $D_2(d)$. We consider then the value of d that lies in the lower quantile of D_2, since those values are the most unfavorable ones to the comparison of two different distributions. In this case, we impose that the similarity is under a given maximum value, typically under 0.05. Then, we can determine σ_{max} by using the formula:

$$\sigma_{max} = \sqrt{\frac{-d_{min}^2}{\log(s_{max})}} \tag{5.4}$$

with $d_{min} = \text{quantile}(D_2, 0.05)$ and $s_{max} = 0.05$. In other words, when $\sigma \leq \sigma_{max}$, the similarity value $s \leq s_{max}$ for at least 95% of the cases.

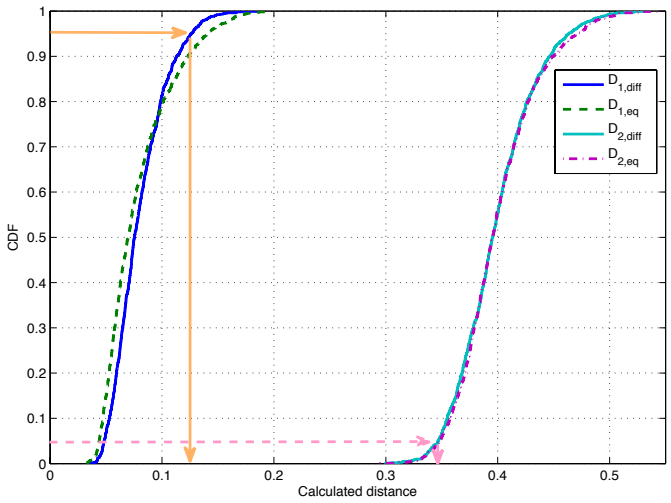


Figure 5.7: Cumulative distributions of the distances calculated using the two calibration procedures depicted in Fig. 5.6. The solid arrow (in orange) indicates the distance value d_{max} corresponding to the 0.95 quantile of D_1 (i.e., for identical sampled distributions), whereas the dashed arrow (in pink) indicates the distance value d_{min} corresponding to the 0.05 quantile of D_2 (i.e., for different distributions). The indices *diff* (resp. *eq*) indicate that samples of different (resp. equal) sizes were generated for the tests.

We have performed this calibration procedure with F_{geo} to calibrate the Jensen-Shannon divergence. The results for D_1 and D_2 are given in Fig. 5.7. On this graph, the solid arrows (in orange) show the distance value d_{max} corresponding to the 0.95 quantile of D_1, which is equal to 0.126 in this case. The dashed arrows (in pink) indicate the distance value d_{min} corresponding to the 0.05 quantile of D_2, equal to 0.347 in this case. Note that we have repeated the very same procedure using samples of equal size, and of different sizes (this is indicated by $D_{i,eq}$ and $D_{i,diff}$ respectively). However, the distributions of distances are very similar in both cases.

Based on those results, we can finally derive an appropriate range of values for σ (using the equations given here above):

$$0.15 \leq \sigma \leq 0.18$$

5.3.3 Cluster analysis of attack events

Having defined all necessary distances and similarities, we are now in a position to perform a cluster analysis of the set of attack events using the dominant sets technique. We start by giving an overview of the clustering results for every feature (e.g., number of clusters, average cluster size, and consistency of the results), followed by some detailed examples showing the meaningfulness of the results and the kind of insights they can offer.

Results overview

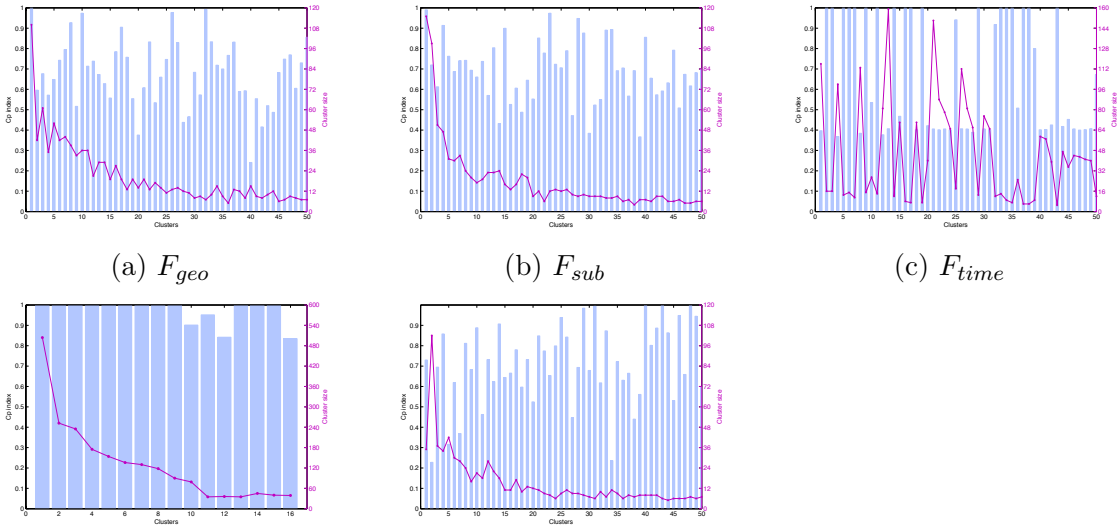
Table 5.3 gives an overview of the clustering results for each feature. For F_{geo} and F_{sub}, the results are very similar, and about 50 to 60% of the events data set could be clustered. The overall quality of the results seems to be fairly good, as the mean compactness of all clusters (calculated using equation 3.13 defined in Chapter 3 on page 58) lies around 0.7. Similar results were obtained for F_{cip}, but apparently the clusters are on average smaller and they are in greater number obviously. F_{time} and F_{ps} seem to correlate attack events even more, with approximatively 85 to 90% of the events being clustered w.r.t. those features. However, there are very few clusters of port sequences (only 16), and these seem to be pretty large. Note also the high compactness value for clusters obtained w.r.t. F_{ps}.

Table 5.3: Overview of the graph-based clustering results for the honeynet data set.

Feature	Nr clusters	Nr μ-events	$\overline{\text{size}}$	$\overline{C_p}$
F_{geo}	134	1,478 (60%)	11.0	0.69
F_{sub}	139	1,310 (53%)	9.4	0.69
F_{time}	50	2,231 (91%)	44.6	0.47
F_{ps}	16	2,103 (86%)	131.4	0.99
F_{cip}	247	1,508 (61%)	6.1	0.66

It is worth noting that the clustering results for F_{ps} seem to indicate that grouping by "port" only leads to global results that are definitively not representative of the *individual phenomena* responsible for targeting those ports. However, we note that this is usually the most commonly-used way of analyzing intrusions or attack traffic collected by IDS sensors and darknets. Most projects collecting data over Internet intrusions and malicious traffic (such as CAIDA [23], Team Cymru [164], ATLAS [164], IMS [6], DShield [45], and many others) tend to group intrusions or attack events *by port* only. They can only provide basic statistics over the way that ports are *globally* more or less targeted, and eventually the overall temporal trends (e.g., an increase of 'x' % of the traffic targeting port 'y'). Generally speaking, this is only useful for observing huge phenomena occurring at Internet-scale (such as flash spreading worms).

We have then further evaluated the validity of those results by representing, for each feature, the compactness (C_p) of the first 50 clusters, and their respective sizes. This is illustrated in Fig. 5.8, where we can clearly see the very high C_p values for F_{ps}, but also the typical behavior of the dominant sets algorithm that extracts the most significant groups (i.e., the largest and most compact ones) in the first stages of the graph clustering. Note that for F_{time}, the clustering appears to be quite different than for the other features, which can be due to the definition of the similarity metric that has only a limited range of discrete values (i.e., $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$). As a result, the optimization process of the algorithm is forced into making harder decisions when grouping events. However, the overall results are still satisfactory, as no cluster has a compactness value below 0.4.



(a) F_{geo} (b) F_{sub} (c) F_{time}

(d) F_{ps} (e) F_{cip}

Figure 5.8: Compactness values of the clusters (in blue) and their respective sizes (in magenta).

Some detailed results

Let us consider some detailed results of clusters for every feature. For the sake of clarity (and to avoid visual clutter in the graphical representations), we have only considered the μ-events that are common to the first 50 clusters of each feature. It turns out that 422 μ-events were involved in this subset of clusters.

To visualize the clusters, we have created 2-dimensional maps for every feature using the same dimensionality reduction technique as used before in Chapter 3, i.e., t-distributed Stochastic Neighbor Embedding, or *t-SNE* [175]. In Fig. 5.9 (a), we see the clusters obtained for F_{geo}. On this map, each data point represents a geographical distribution of a given μ-event, and the coloring refers to the cluster membership.

To illustrate the patterns found by the clustering, the centroïds of certain clusters are indicated by arrows on the map. Interestingly, the analyst gets immediately a nice overview of the underlying structure of the attack events (and the inter-relationships) **as viewed w.r.t. a given viewpoint**, in this case the geographical origins. For example, we can easily observe *commonalities* among patterns of *nearby clusters* (involving in most cases

several popular or large countries, from which many attacks can be observed), and thus we can also understand the logical structure behind the positioning of the clusters on the map.

Similarly, the map in Fig. 5.9 (b) represents the distribution of IP subnets for the very same set of 422 μ-events. Here too, we can observe some well-separated clusters of events, for which the subnet patterns are rather unique and share apparently nothing in common with other groups of events. Conversely, we observe also fuzzier patterns for certain groups of events (probably related to attacks coming from largely infected IP subnets), forming thus clusters that are close to each other, which illustrates again the *fuzziness* of real-world attack phenomena.

Then, Fig. 5.9 (c) shows a map of events obtained for F_{time}. We see that there are just a few groups of different clusters, which are apparently clearly separated from each other. On this map, we have indicated the patterns of certain events with the following scheme:

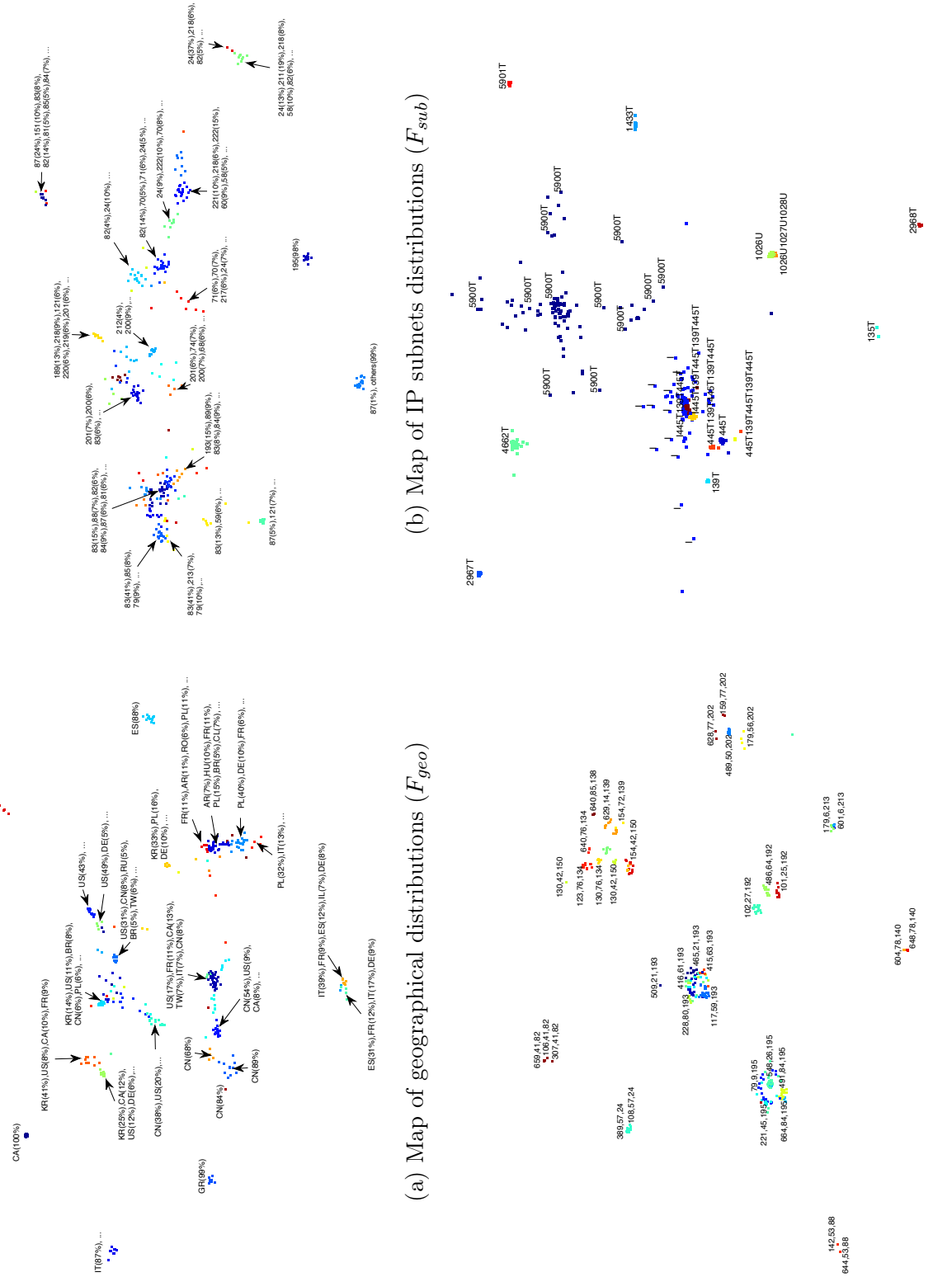
$$\{\mathcal{M} - \text{event id}\}, \{\text{sensor id}\}, \{\text{targeted IP subnet}\}$$

In each group of well-separated clusters, we observe that all events are linked to the same set of platforms or the same targeted subnet, and the μ-events belonging to the same cluster (i.e., data point in the same color) are usually also involved in the very same $\mathcal{M} - event$.

Finally, Fig. 5.9 (d) shows the 2D map obtained for clusters of port sequences (F_{ps}). For this map, we have intentionally stopped the t-SNE algorithm after a few iterations only, so as to be able to distinguish the different data points and clusters. Otherwise, all points belonging to the same cluster would end up being mapped to a single (x,y) coordinate, due to the high similarities among them. For this reason, certain clusters of ports sequences can still appear as somehow spread over a given area (like for 5900T and ICMP, for example). Here too, we can see that μ-events with similar port sequences are being mapped to nearby clusters, whereas events targeting completely different ports are mapped to distant points.

As the clustering results suggest, each attack feature seems to bring an interesting viewpoint on the attack phenomena that have been observed by the honeypots. Intuitively, we can easily imagine how the combination of several viewpoints can further emphasize those phenomena. For example, this would enable us to get better insights into a botnet phenomenon for which the attack events are strongly correlated by the origins, but they target different sensors located in different IP subnets and use different exploits (and thus target different ports). The problem is: *how* to combine all those features, so as to get meaningful results? *Which features* do we have to combine to observe phenomenon X or Y? In fact, the attribution of different events to the same phenomenon does not always appear so clearly, and performing such a multi-criteria analysis *by hand* is a very tedious process, due to the many possible combinations to be investigated.

To illustrate this point, let us visualize in Fig. 5.10 (on page 118) a given phenomenon as viewed through four different attack features. An in-depth analysis has revealed that all μ-events highlighted with larger pixels (and enclosed in dashed rectangles) are due to the same botnet that has targeted a subset of platforms located in different subnets, at different moments (for a total duration of about 3 months), and using several exploits on ports 445T and 139T. As we see on the maps, those attack events are thus highly correlated w.r.t. F_{ps} and F_{time}; however, they still form several smaller clusters which are somehow



(a) Map of geographical distributions (F_{geo})

(b) Map of IP subnets distributions (F_{sub})

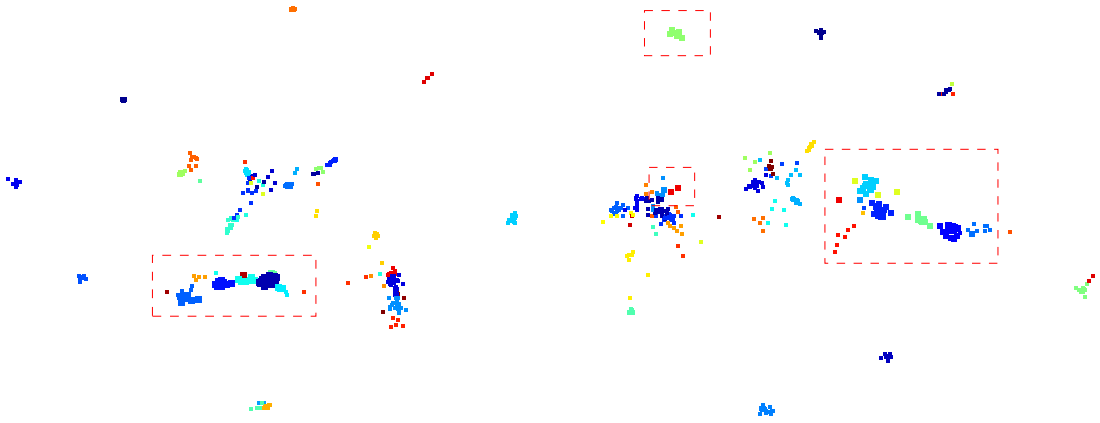
(c) Map of temporal correlations (F_{time})

(d) Map of Port sequences (F_{ps})

Figure 5.9: Visualizing clustering results for 422 μ-events of the honeynet data set.

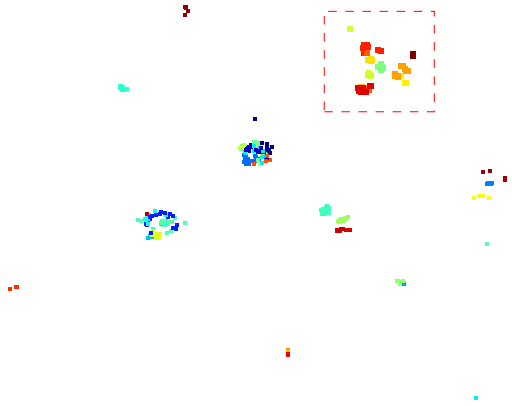
close to each other. Regarding the other two features (F_{geo} and F_{sub}), we see that the origins of the phenomenon have apparently evolved over time, as the μ-events are grouped into a series of 8 to 10 different clusters which together form an elongated shape on the maps.

This example illustrates again the two fundamental issues described in Section 4.1, i.e., the *fuzzy* and *dynamic* aspects of attack phenomena, which complicates the root cause analysis of security events. In the next Section, we will perform a systematic multi-criteria aggregation of those attack events, by leveraging the aggregation techniques described in Chapter 4. As we show in the experimental results, this can effectively help us to combine different viewpoints and get insights into complex behaviors of attack phenomena.

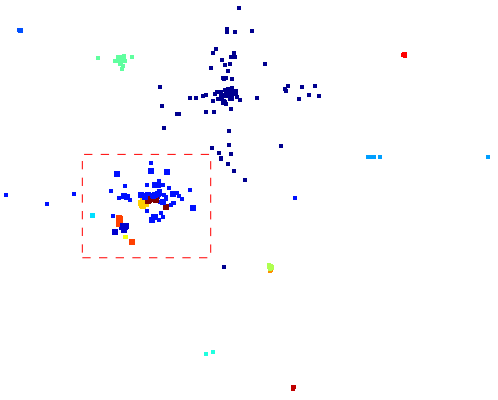


(a) Phenomenon viewed w.r.t. F_{geo}

(b) Phenomenon viewed w.r.t. F_{sub}



(c) Phenomenon viewed w.r.t. F_{time}



(d) Phenomenon viewed w.r.t. F_{ps}

Figure 5.10: Visualization of the **same phenomenon** (i.e., the events of larger size highlighted by the dashed rectangles), **as viewed w.r.t. four different attack features**. In this illustration, we can clearly observe the *fuzzy* and *dynamic* aspects of the phenomenon.

5.4 Aggregation of all features

5.4.1 Defining parameters

We are now in the position of combining all attack features by using an *aggregation function*. Recall that the purpose of this final step consists in finding which subsets of μ-events (which can belong eventually to different clusters w.r.t. individual features) are very likely due to a same root cause.

To perform this aggregation, we have extensively described in Chapter 4 two main classes of functions: (i) Ordered Weighted Averaging functions (i.e., OWA and WOWA), and (ii) an aggregation based on the Choquet integral. We have thus applied each of these aggregation functions to our 2-year data set of attack events. For OWA functions, we simply need to define two weighting vectors, namely $\mathbf{w}$ (used in OWA to quantify the importance of the largest or smallest similarity scores), and $\mathbf{p}$ (used in WOWA, in conjunction with $\mathbf{w}$, to quantify the reliability of the features).

Based on our domain analysis of the selected attack features (that we have done in Section 5.2), and considering the behavior of those OWA functions (described in Chapter 4), we can derive some reasonable sets of values to model the behavioral requirements of the phenomena that we want to identify. Hence, we define $\mathbf{w}$ and $\mathbf{p}$ as follows:

$$\begin{cases} \mathbf{w} &= [\,0.05,\ 0.10,\ 0.35,\ 0.40,\ 0.10\,] \\ \mathbf{p} &= [\,0.10,\ 0.15,\ 0.30,\ 0.15,\ 0.30\,] \end{cases}$$

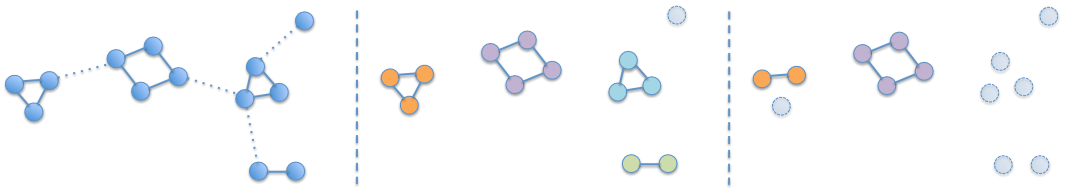
The weighting vector $\mathbf{w}$ expresses that **at least three features** must be strongly correlated in order to obtain a global score close or above 0.5, whereas vector $\mathbf{p}$ quantifies the reliability of each feature, drawn from our domain knowledge.

Regarding the Choquet integral, we have to define a fuzzy measure v. However, we should define in this case 32 different values corresponding to the 2^5 possible combinations of features. As described in Chapter 4, when the number of features n becomes too excessive to define manually such a fuzzy measure, we can rely on different methods to reduce this complexity, for example, we can simply define a λ-fuzzy measure v_λ.

In this application, we have in fact three *redundant features* (which are F_{geo}, F_{sub} and F_{cip}). As a result, it is probably appropriate to define v_λ so that the global behavior of the fuzzy measure is *subadditive*. Moreover, we have to keep also the relative importance of each feature. So, for the definition of each singleton value of v_λ, we can simply start with the following values: [0.10, 0.20, 0.40, 0.20, 0.40]. Those values are consistent with the preference relations between criteria expressed previously (in Section 5.2), and the sum of the values is greater than 1, which leads to a subadditive fuzzy measure by resolving the equation 4.22 (i.e., we obtain then $\lambda = -0.396$).

A last thing we need to define is of course the **decision threshold** (ε) that is used to identify connected components within the combined graph. This is a quite important step, since the properties of the resulting components identified within the graph highly depend on this threshold.

One way to define an appropriate range of values for ε is to perform a *sensitivity* analysis. By analyzing the impact of this decision threshold on the components within the graph, we have observed the following phases as the threshold increases (Fig 5.11):



(a) Phase 1 (b) Phase 2 (c) Phase 3

Figure 5.11: Illustration of the 3 phases of formation when searching for connected components (i.e., phenomena) within a graph by increasing the decision threshold ε. (a) "Phase 1" is where phenomena are being formed by breaking the weak and unwanted relations among them (represented with dashed lines). (b) "Phase 2" is where the number of phenomena becomes stable. (c) "Phase 3" is where an excessive threshold begins to split and eliminate phenomena.

(i) during the first phase, the number of phenomena increases as well, since phenomena are being formed by removing all weak (and unwanted) relationships among them (Fig 5.11 (a));

(ii) at the beginning of the second phase, we observe a stable number of phenomena (as all weak edges have been removed), shortly followed by a new increase of this number because the increasing threshold starts to split certain phenomena into smaller parts (Fig 5.11) (b));

(iii) finally, in the last phase, the decision threshold becomes excessive, and thus the number of phenomena starts to decrease as too many edges (even strong ones) are being removed (Fig 5.11 (c)).

Applying this approach to our data set, we can now look at the evolution of the number of *largest phenomena*, $|\mathcal{P}| = \sum |P_i|$, as a function of ε. By "largest", we mean phenomena comprising at least 10 μ-events (i.e., $|P_i| \geq 10$). This is illustrated in Fig. 5.12 (a) for the OWA aggregation, and in Fig. 5.12 (b) for the Choquet integral (the sensitivity analysis of the WOWA aggregation is very similar to those curves). We have indicated on the graphs the three regions of interest in the determination of ε.

Interestingly, in the second region ("Zone 2"), we can observe a sort of small *plateau* around $\varepsilon = 0.6$, which seems to indicate some good values for the decision threshold as the number of phenomena is fairly stable in this region. Increasing further the threshold leads to a slight increase of $|\mathcal{P}|$ (up to a maximum value), which could *a priori* also provide meaningful results to the analyst. However, we have observed that this maximum number of phenomena is generally not as good as those obtained with the first plateau, for the simple reason that several components P_i are then split into smaller subgraphs.

The last region ("Zone 3") is obviously not indicated, since the decision threshold becomes excessive and we eliminate too many μ-events. Thus, we lose some semantics about the phenomena.

Observe also that the appropriate range of values for ε usually lies around k/n, with k the desired number of correlating features (in this case, $3/5$).

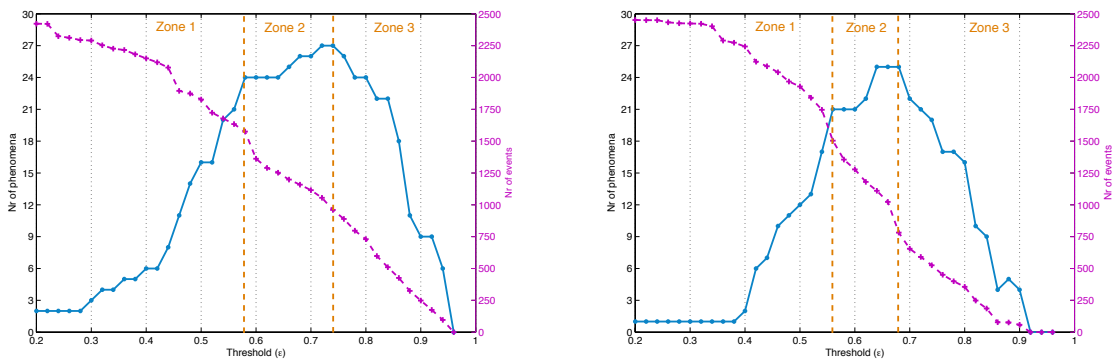


Figure 5.12: Sensitivity analysis of (a) OWA aggregation; and (b) Choquet integral. The regions 1, 2 and 3 indicated on the graphs correspond to the 3 phases described in Fig. 5.11. The axis on the right (in magenta) shows the total number of μ-events (i.e., graph nodes) being grouped into phenomena.

5.4.2 Results overview

Let us briefly compare the results given by each aggregation method. Looking at Table 5.4, we see that the decision threshold ε was approximatively the same for each aggregation technique. The total number of phenomena $|\mathcal{P}|$ found by each technique is rather limited, and this number is also roughly the same for each method. However, the Choquet integral can apparently find more correlated μ-events, and can assign them to the same number of phenomena.

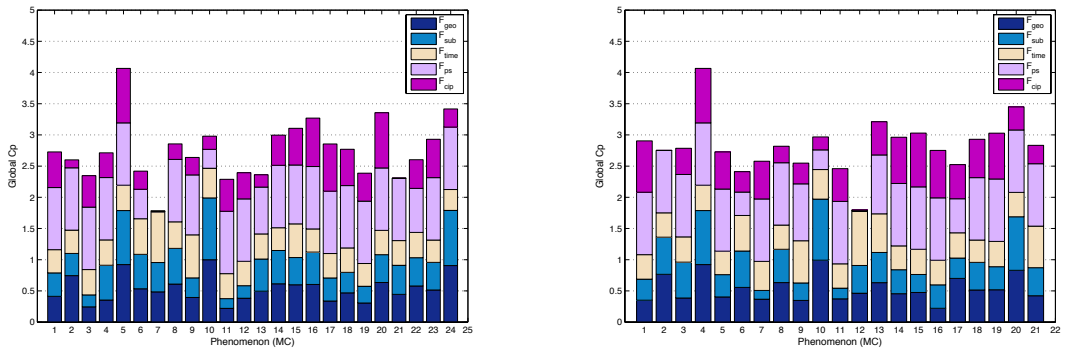
Table 5.4: Comparison of different aggregation methods.

Characteristic	OWA	WOWA	Choquet
Threshold ε	0.58	0.59	0.56
$\lvert\mathcal{P}\rvert$	73	84	73
$\lvert\mathcal{P}\rvert$, with $\lvert P_i\rvert \geq 10$	24	21	21
Nr of μ-events	1,685	1,685	1,748
Average C_p	0.54	0.57	0.58

The sets of phenomena found by the three techniques are fairly consistent with each other. Overall, we found approximatively 20 large phenomena (i.e., containing at least 10 μ-events), and the average compactness of each set of phenomena is just under 0.6, which is fairly good. The best results were obtained using the Choquet integral, with admittedly a rather incremental improvement (w.r.t. C_p) over the other two techniques. However, the Choquet integral could attribute more events to phenomena, thanks to its greater modeling capability.

To further evaluate the consistency of the results, we have also represented in Fig. 5.13 the global graph compactness C_p for the largest phenomena, as calculated individually *by feature*. This view is also interesting to figure out which features tend to group attack events

together within each P_i. As one can see, F_{ps} has in general a very high C_p, whereas F_{time} has usually the smallest compactness. For each P_i, we observe that it can be characterized by varying degrees of correlation regarding each feature, but overall there are always at least three different features with a high correlation.



(a) OWA (b) Choquet

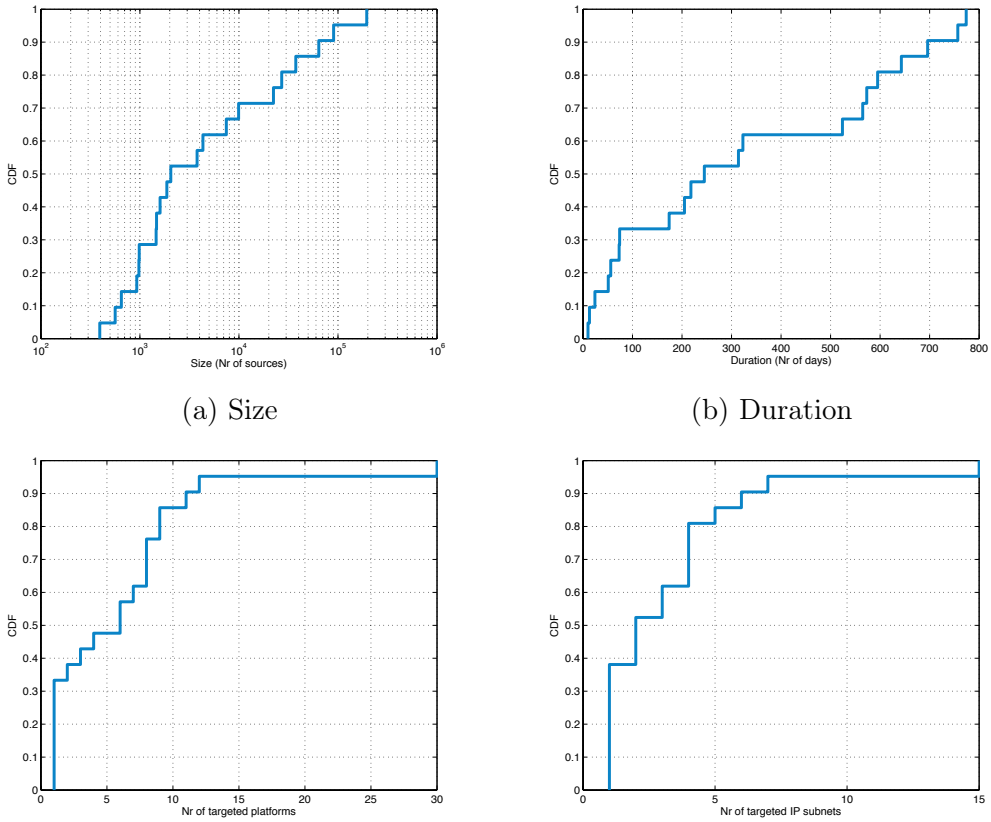
Figure 5.13: Evaluation of the results consistency with the global compactness (C_p) of the largest phenomena found using (a) the OWA aggregation; and (b) the Choquet integral. Each color refers to the C_p index of each feature individually.

Finally, let us consider some global statistical properties of the 21 largest phenomena (as found by the Choquet integral). Fig. 5.14 (a) represents the cumulative distribution (CDF) of the phenomenon's size, in terms of number of sources involved in each phenomenon. Some phenomena contain rather few sources, however, there are still 30% of them that consist of more than 20,000 (observed) sources.

Fig. 5.14 (b) shows the CDF of the phenomenon's lifetime (or duration). Such lifetime is defined as the time interval, in days, between the very first and the very last attack event of a given P_i. As we can see, the CDF is rather linear, and the average lifetime of an attack phenomenon is about 330 days. Certain phenomena have even been observed for more than 700 days!

Looking at the CDF of the number of targeted platforms, in Fig. 5.14 (c), and the CDF of the number of targeted IP subnets, in Fig. 5.14 (d), we can conclude that most phenomena are seen on less than 10 platforms that are in general located in maximum 5 different subnets.

These various characteristics suggest that the root causes behind the existence of these attack phenomena are fairly stable, localised attack processes. In other words, different places of the world do observe different kind of attackers, but their modus operandi remain stable over a long period of time. We are, apparently, not so good at stopping them from misbehaving.



(a) Size (b) Duration

(c) Targeted platforms (d) Targeted IP subnets

Figure 5.14: Overview of some statistical characteristics of the largest attack phenomena.

5.5 Behavioral analysis of attack phenomena

5.5.1 Introduction

In this final Section, we give a more in-depth analysis of 5 illustrative case studies, in order to show the kind of insights our attribution method can offer, in a systematic manner, into the behavior of the identified phenomena.

In fact, the attack phenomena we have found within this 2-year honeynet data set, seem to form some sort of "clouds of malicious sources", each cloud showing a given type of behavior. For this reason, we have called those attack phenomena "**Misbehaving Clouds**" (or *MC*), which seems an appropriate term to describe them.

The five *MC*'s that are studied in this Section have been summarized in Table 5.5. Note that these experimental results are based on case studies that have been presented in [167, 168, 40].

Those Misbehaving Clouds involve several common services such as NetBios (ports 139/TCP, 445/TCP), Windows DCOM Service (port 135/TCP), Virtual Network Computing (port 5900/TCP), Microsoft SQL Server (port 1433/TCP), Windows Messenger Service (ports 1025- 1028/UDP), Symantec Agent (port 2967/TCP), and some others. Not surprisingly, those services are amongst the ones that are commonly exploited by well-known

Table 5.5: High-level characteristics of five Misbehaving Clouds (MC's) under study. The colon *Root cause* refers to the presumed type of phenomenon, based on the results of the attack attribution method. Δt_{MC} indicates the *lifetime* of MC's.

MC	# events	# sources	Δt_{MC}	Root cause	Targeted ports
1	143	64,054	323	Botnet cloud	ICMP, 445T (Microsoft-DS), 139T (Netbios)
2	578	90,386	774	Worm-behaving cloud	1433T (MSSQL), 1025T (RPC), 5900T (VNC), 135T-139T (Netbios), 2967T-2968T (Symantec)
3	63	48,438	634	UDP spammers (botnet)	1026U (Windows Messenger)
4	112	195,234	696	UDP spammers (botnet)	1026U, 1027U, 1028U (Windows Messenger)
5	38	35,588	759	Allaple worm	ICMP, 139T, 445T
6	147	27,030	573	P2P	Unusual ephemeral ports (TCP)

families of bot software, such as SDBot, Spybot, Agobot, GT Bot and RBot [136, 79].

Regarding the origins of MC's, we can observe some very persistent groups of IP subnets and countries of origin. To illustrate this point, we have represented in Fig. 5.15 the CDF of the IP addresses involved in those five MC's, where the x-axis represents the first byte of the IPv4 address space. Observe that MC3 and MC4 have apparently a very singular behavior (i.e., uniform distribution), which will be explained later in Section 5.5.4.

Clearly, malicious sources involved in those phenomena are highly unevenly distributed, and form a relatively small number of tight clusters that are responsible for a large deal of the observed malicious activities. This is consistent with other prior work on monitoring global malicious activities, in particular with previous studies related to measurements of *Internet background radiation* [30, 114, 195].

However, we can show here that there are still some notable differences in the spatial distributions of those misbehaving clouds, even though there is some overlap between "zombie-friendly" IP subnets (such as the distributions of MC1, 2 and 5). Moreover, because of the dynamics of those phenomena, we can even observe different spatial distributions within the *same cloud* at different moments of its lifetime. This is an advantage of our analysis method, which can be more precise and enables us to distinguish *individual* phenomena, instead of some global trends.

5.5.2 Coordinated Botnet

The Misbehaving Cloud $MC1$ is a first interesting case involving a *coordinated botnet* that has mostly targeted the Windows ports (445T and 139T). An in-depth analysis of the shapes of the time series, and the arrival rate of the sources involved in the 143 μ-events, has led us to conjecture that MC1 is quite likely due to a botnet phenomenon.

On Fig. 5.16, we see that this cloud had four main *waves of activity* during which it was randomly scanning different subnets (actually, always the same 5 subnets). Note also the perfect coordination of the time series. When inspecting the IP subnet distributions of the sources belonging to those different attack waves, we could also observe a *drift in the origins*, probably as certain machines were infected by (resp. cleaned from) the bot software. In fact, this drift in the origins can be visualized in Fig. 5.10, where the events of this phenomenon have been highlighted by dashed rectangles.

What is also of interest is that $MC1$ has showed a **coordination** among its sources, with apparently *two different communities* of machines. In fact, a very large community

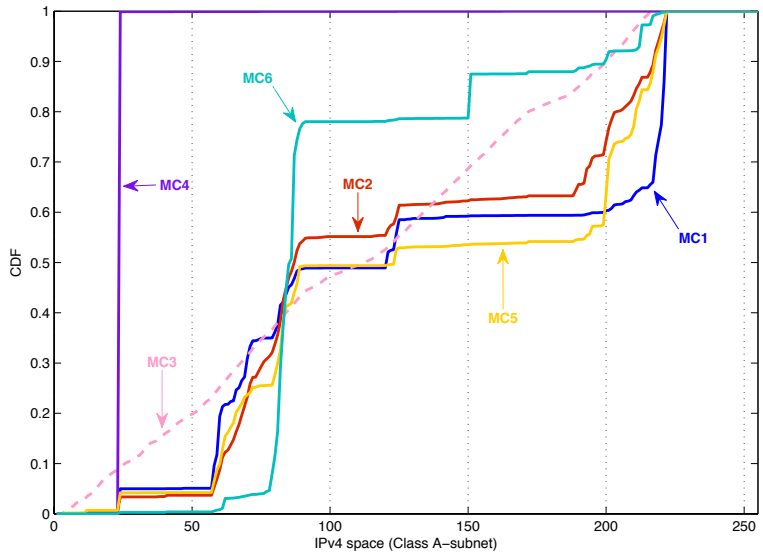


Figure 5.15: Cumulative distributions (CDF's) of originating IP subnets for the largest phenomena.

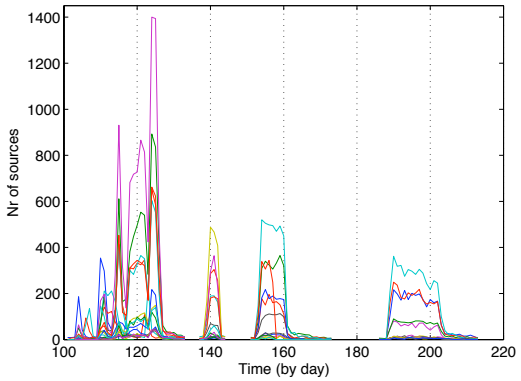


Figure 5.16: Coordinated time series of μ-events belonging to MC1.

of machines within $MC1$ was simply scanning (in a random fashion) the IP subnets where our sensors are located, using innocuous ICMP packets. On the other hand, we found out there was another (smaller) group of machines, also involved in $MC1$, but those were directly attacking the Windows honeypots on specific ports (139T and 445T), like if they were controlled to attack only specific IP addresses. Furthermore, this group of attacking machines had very different origins from those of the ICMP scanners.

To further confirm this assumption, we have computed the distribution of targeted honeypots, for each μ-event. The result is quite surprising: the "scanning sources" are equally targeting the three honeypots of each platform (i.e., 33% of the scanners have been observed on every honeypot IP address), whereas the attacking machines are only targeting the two Windows honeypots, with exactly 50% of the sources observed on the two first IP's of each platform (and none on the third honeypot, which emulates a Linux OS).

We hypothesize that the group of attackers probably took advantage of the scanning results given by the larger community of scanners of the same phenomenon (scanners were probably relying on some OS fingerprinting technique, such as *P0f* or *Xprobe*). It is interesting to see how bots of the same botnet are thus able to coordinate their efforts.

In Fig. 5.17, we provide another visualization of this misbehaving cloud using a *node-link graph*, so that an analyst can visualize, in a single picture, multiple attack features and all relevant relationships among events.

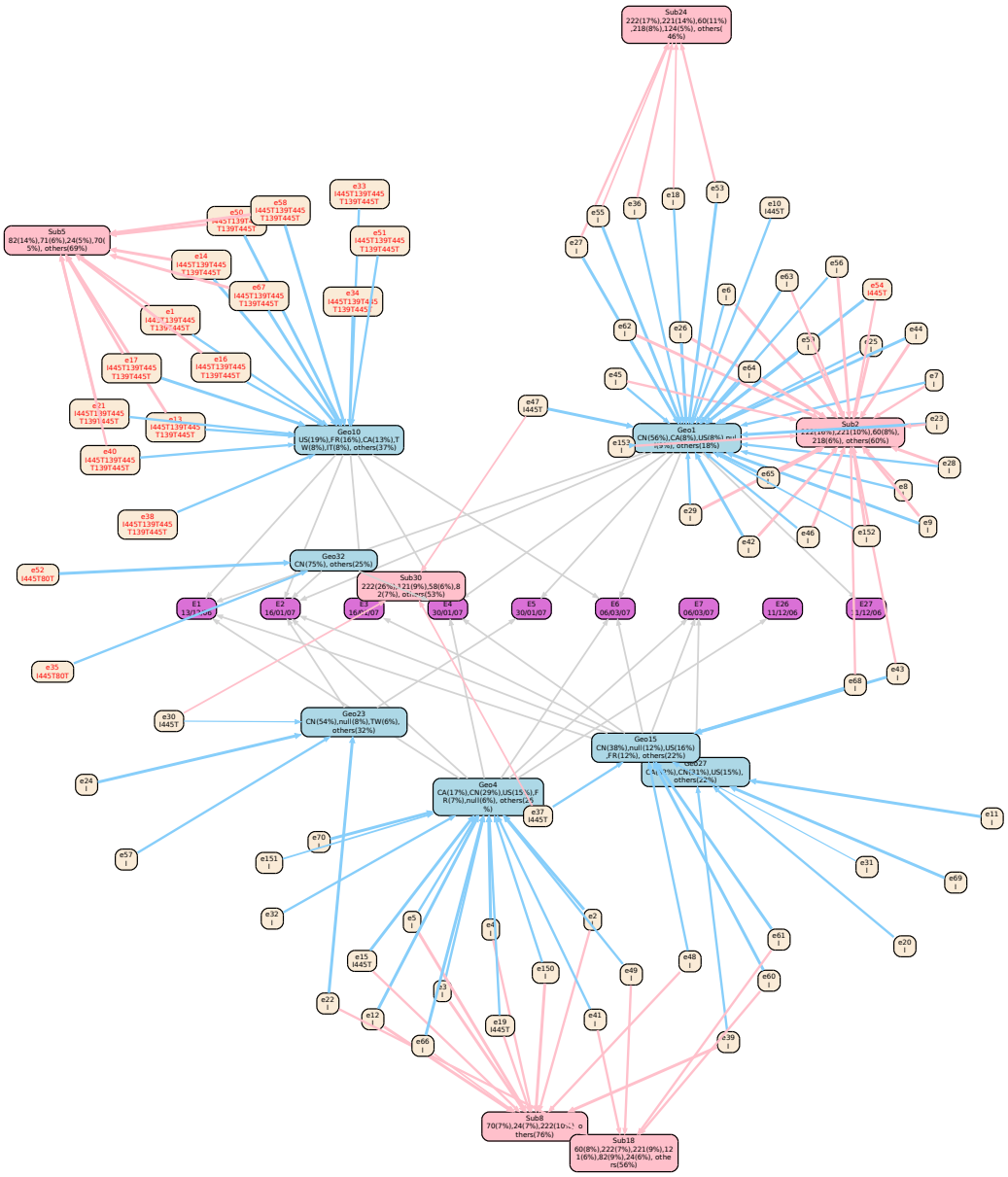


Figure 5.17: Node-link graph for $MC1$, which represents relationships between μ-events (in antique white) w.r.t. different attack features.

In this type of visualization, we have represented the μ-events of $MC1$ with nodes in "antique white" color. The μ-events are then linked to other nodes representing the *clusters* they belong to, w.r.t. different features. In this case, clusters represented with blue nodes refer to F_{geo}, clusters in pink refer to F_{sub}. The centroïds of the clusters (i.e., the average distributions) are written as labels in the nodes. In the middle of the graph, we have represented a time axis with the starting dates of the $\mathcal{M}$-events that contain the μ-events, so we have an idea of how many "waves" of attacks this MC has performed.

This node-link graph further illustrates the typical behavior of $MC1$, i.e., the *separation of duties* between scanners and attackers. In this cloud, μ-events involving machines attacking or exploiting directly the Windows ports are highlighted with port sequences written in red inside the node labels. We can also observe the *drift* in the origins of the sources, which leads to multiple geographical clusters having quite different distributions of countries and IP subnets.

5.5.3 Worm-behaving Cloud

$MC2$ consists of 578 μ-events. The temporal shape of those events is fairly similar to the one left by a typical worm: its trace exists for several days, it has a small amplitude at the beginning but grows quickly, exhibits important drops that can correspond to subnets being cured or blacklisted, and it eventually dies slowly (see [117] for a more formal description of this class of phenomena).

The interesting thing with $MC2$ is that it is made of a sequence of *worm-like* shaped μ-events, and the lifetime of this MC is fairly long: 774 days! It is composed of μ-events that have targeted a number of distinct services, including 1025T (RPC), 135T-139T (Netbios), 1433T (SQL), 2967T-2968T (Symantec) and 5900T (VNC). Many different exploits have thus probably been included in the worm codebase, so as to maximize its propagation.

The results of the multi-criteria fusion algorithm indicate that those μ-events have been grouped together mainly because of the following three features: the origins, the targeted platforms, and the port sequences. Moreover, it seems that an important amount of IP addresses is shared by many μ-events composing this MC. Fig. 5.18 shows another visualization of this phenomenon using a node-link graph and multiple attack features.

To illustrate the kind of μ-events found in this MC, Figures 5.19 (a) and (b) represent four μ-events time series. Figure 5.19 (a) represents two of them, namely e626 and e628, consisting of activities against Microsoft SQL Server (1433/TCP). Whereas Figure 5.19 (b) represents the other two, namely e250 and e251, consisting of activities against a Symantec Service (2967/TCP). Figure 5.19c zooms on these last two μ-events from day 100 to day 150. We can observe the slow increase of the two curves that are apparently typical of worm-related attacks [117, 196].

The two μ-events on the left (resp. middle) share 528 (resp. 1754) common IP addresses with each other. Given these elements, we are tempted to believe that e626 and e628 (resp. e250 and e251) are generated by the same worm, called $WORM\_A$ (resp. called $WORM\_B$). Both worms, $WORM\_A$ and $WORM\_B$, target the same two platforms: 25 and 64. Furthermore, we found that these four μ-events share an important amount of common compromised machines. This could indicate that both worms, before having contacted our honeypots, had contaminated a relatively similar population of machines. A plausible explanation could be that both had been launched from the same initial set of

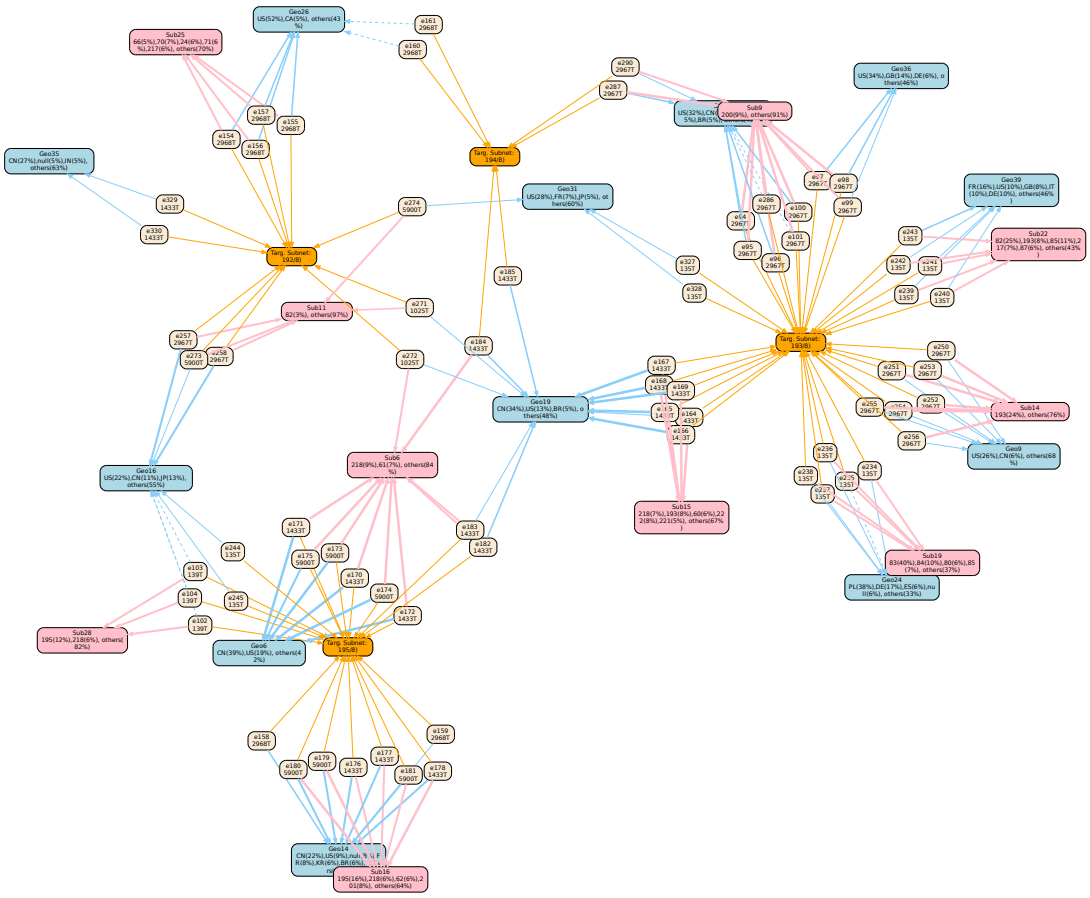


Figure 5.18: Visualization of the μ-events composing $MC2$, using a node-link graph. The following attack features are represented: F_{geo} (in blue), F_{sub} (in pink), and F_{ps} (in the node labels). To avoid visual clutter, in the yellow nodes, we have grouped the targeted platforms located within a same Class A-subnet (this information is derived from the feature F_{time}).

machines and that they were using the same, or similar, code to choose their targets.

From the attack vector viewpoint, these two worms have nothing in common since they use very different types of exploits. Moreover, they have been active in different periods of time. However, the analysis reveals that they exhibit a very similar pattern, both in terms of propagation strategy and in terms of success rates. Thus, even if the infection vector differs between the two, the starting point of the infection as well as the code responsible for the propagation are, as explained, quite likely very similar. This reasoning can be generalized to all 578 μ-events, revealing the high probability that all these different attacks have some common root cause(s).

This does not mean *per se* that all these attacks are due to the very same person or organisation -even if this is likely- but it indicates that the same core piece of code has probably been reused, from a very similar starting point to launch a number of distinct attacks. This reveals some aspect of the *modus operandi* of those who have launched these attacks, and this is an important piece of information for those who are in charge of identifying these misbehaving groups, as well as their tactics.

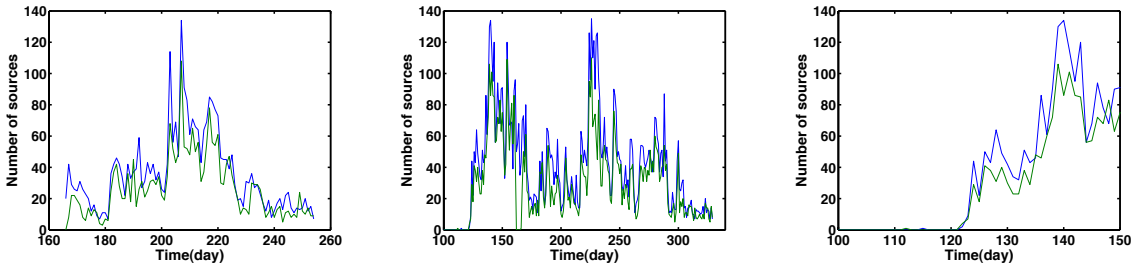


Figure 5.19: Attack time series (nr of sources by day) of some μ-events from $MC2$, targeting (a) MS SQL Server (1433/TCP), (b) Symantec agent (2967/TCP). Fig. (c) is a zoom on (b).

5.5.4 Botnet serving Messenger Spammers

In this case study, we look at two distinct clouds, namely $MC3$ and $MC4$. Both are made of μ-events that have exclusively tried to send spam to innocent victims thanks to the Windows Messenger service, using UDP packets. This is a rather complex but interesting phenomenon that, as far as we know, very few people have investigated at the time of writing this dissertation. A possible reason is that people consider those UDP packets just as "junk traffic", which can be easily spoofed and thus also difficult to analyze. Consequently, security practitioners simply prefer to disregard or filter out those packets, instead of investigating further the root causes of this phenomenon.

However, this won't allow us to stop UDP spammers from misbehaving. As our experiments demonstrate hereafter, our method offers some unique insights into this UDP phenomenon.

The two clouds $MC3$ and $MC4$ have been observed over a large period of time, more than 600 days in both cases. Even if they, conceptually, look similar, there are important differences between the two clouds. First, the targeted ports are not identical: in $MC4$, UDP packets are being sent to three different UDP ports, namely 1026, 1027 and 1028, while in $MC3$ packets are sent exclusively to the 1026 UDP port.

Then, as illustrated in Fig.5.15 where we can see the cumulative distribution (CDF) of sources IP addresses (grouped by /8 blocks of addresses), we observe that $MC3$ is uniformly distributed in the IPv4 space. This result is absurd, since large portions of the IPv4 space can not be allocated to individual machines (due to multicast, bogons, unassigned, etc.) and, in all these regions, it is impossible to find compromised machines sending spams. If we find these IPs in packets hitting our honeypots, it clearly means that these are spoofed IP addresses. Furthermore, the uniform distribution of all the IP addresses in that MC leads us to believe that all other IPs are also spoofed.

On the other hand, $MC4$ has a constant distribution pointing exclusively to a single /8 block owned by an ISP located in Canada[4]. A likely explanation is that those spammers have also used spoofed addresses to send UDP messages to the Windows Messenger service, and they have been able to do so for 600 days without being disturbed!

To further validate those results, we have also looked at the payloads of the UDP pack-

[4]Actually, a closer inspection of sources IP addresses reveals they were randomly chosen from only two distinct /16 blocks from this same /8 IP subnet.

ets by computing a hash for each packet payload. What we discovered is quite surprising: all payloads sent by the sources have exactly the same message template, but the template was different for the two clouds. Fig.5.20 and Fig.5.21 show the two different templates used by spammers of $MC3$ and $MC4$ respectively. Regarding $MC3$, we also observe many alternate URL's, such as: `32sys.com`, `Fix64.com`, `Key32.com`, `Reg64.com`, `Regsys32.com`, `Scan32.com`, etc, whereas spammers in $MC20$ use apparently almost[5] always the same URL (`www.registrycleanerxp.com`).

```
SYSTEM  ALERT  - STOP! WINDOWS REQUIRES IMMEDIATE ATTENTION.
Windows has found CRITICAL SYSTEM ERRORS.

To fix the errors please do the following:
1. Download Registry Cleaner from: http://www.wfix32.com
2. Install Registry Cleaner
3. Run Registry Cleaner
4. Reboot your computer
FAILURE TO ACT NOW MAY LEAD TO DATA LOSS AND CORRUPTION!
```

Figure 5.20: Spam template used in $MC3$.

```
Local System User
CRITICAL ERROR MESSAGE! - REGISTRY DAMAGED AND CORRUPTED.

To FIX this problem:
Open Internet Explorer and type:  www.registrycleanerxp.com
Once you load the web page, close this message window

After you install the cleaner program
you will not receive any more reminders or pop-ups like this.

VISIT www.registrycleanerxp.com IMMEDIATELY!
```

Figure 5.21: Spam template used in $MC4$.

All this knowledge derived from the observation of the MCs illustrates already the richness and meaningfulness of the analyses that can be performed. At this point, there are still two questions left unanswered when we look at those two UDP spam phenomena:

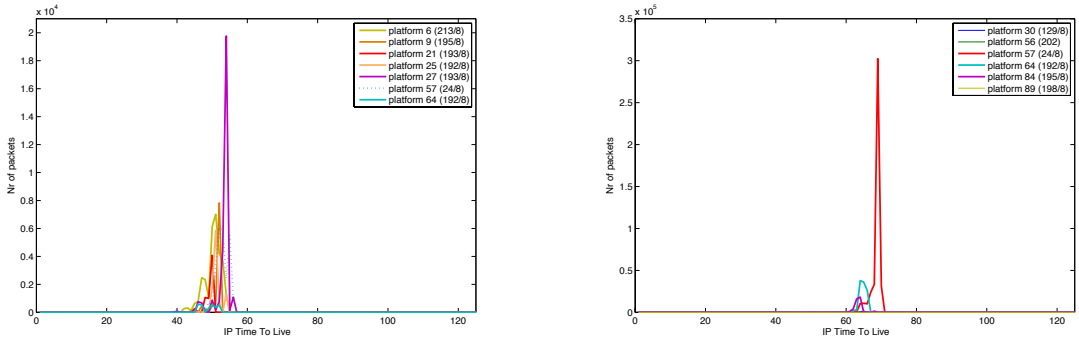
(i) Do all those UDP packets really use **spoofed** IP addresses, and how were they sent (e.g., from a single machine in the Internet, or via a large botnet)?

(ii) Could it be that those two phenomena have in fact the **same root cause**, i.e., the same (group of) people running in parallel two different spam campaigns?

To answer the first question, we have extracted from the UDP packets the Time To Live (TTL) value of their IP headers. We have computed the distributions of these TTL values

[5] For $MC20$, only a few instances of spam messages were observed with a different URL: `nowfixpc.com`

for both phenomena, grouped by targeted platform. The results, illustrated in Fig.5.22, seem to confirm our intuition about spoofed UDP packets, since these TTL distributions are too narrow to originate from a real population of physical machines. In both cases ($MC3$ and $MC4$), we observe that the TTL distributions have a width of about 5 hops, whereas TTL distributions for non-spoofed packets are normally much larger, as malicious sources are, generally speaking, largely distributed in several AS's. As a sanity check, we retrieved the TTL distributions for another phenomenon, which has been validated as a botnet of Windows machines (actually, $MC1$). As one can see in Fig.5.23, the TTL distributions are much larger (around 20 hops) than for spoofed UDP packets.



(a) (b)

Figure 5.22: TTL distribution of UDP packets for $MC3$ (a) and $MC20$ (b) (grouped by targeted platform)

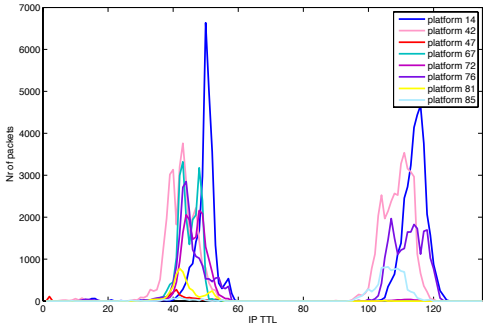


Figure 5.23: TTL distribution of TCP packets for a phenomenon (MC28) attributed to a botnet of Windows machines targeting ports 445T and 139T (grouped by targeted platform).

Another finding that we can presumably derive from Fig.5.22 is that some unusual initial value was used as TTL's. The default initial value for TCP and UDP packets sent from Windows platforms is 128 for most recent versions of the OS, i.e., starting from Windows NT 4.0 and XP [95, 96]. For ICMP packets, the default TTL is usually 64. However, previous research has showed that the average number of hops grows logarithmically with the size of the network in networks that are both scale-free and small-world, which are at the heart of systems like the Internet and the World Wide Web [161, 124]. Furthermore,

several authors have showed, by means of active measurements of the Internet topology, that the average number of hops between two devices in the Internet lies usually around 17 hops [66, 97]. Another measurement study in [50] has showed that, in the continental US, more than 90% of hosts can be reached within 18 hops, whereas the results of the same study for international measurements (i.e., US to Europe and US to Asia) show that the average number of hops was rarely above 20 hops. Consequently, we can assume that the initial TTL value used for UDP packets in MC3 (Fig.5.22 (a)) was probably 64, while the initial TTL value for MC20 (Fig.5.22 (b)) lies probably around 80. In conclusion, this TTL viewpoint seems to confirm the assumption that those UDP packets were probably forged using raw sockets, instead of using the TCP/IP protocol stack of the operating system.

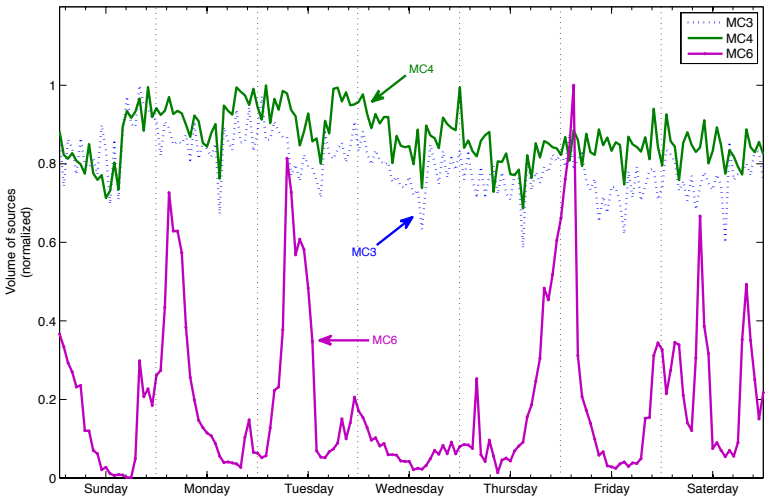


Figure 5.24: Distribution of malicious sources grouped by weekdays. For each MC, a data point represents the accumulated number of sources observed for a given day and hour of the week.

Finally, trying to answer the last question (same root cause or not), we looked at one additional feature of the attacks. We generated a distribution of sources by grouping them based on the *day and hour of the week* they have been observed by our platforms (using the same universal time reference, which is GMT+1 in this case).

As one can see in Fig.5.24, the result is very intriguing: although there is no privileged day or time interval in the week on which we observe a specific pattern, the UDP traffic created by $MC3$ (in dashed) and $MC4$ (in green) look apparently *synchronized*. Since both phenomena have lasted more than 600 days, it is quite unlikely that such correlation could be due to chance only. So, while we have no true evidence to verify this, we can reasonably assume that both phenomena have been orchestrated by the same people, or at least using the same software tool and sets of compromised machines.

One could argue that this temporal correlation is possibly an artefact due to groups of compromised machines lying in the same subnetwork, which are up and running all day and thus exhibit the very same diurnal pattern with respect to their availability to UDP spammers. However, from what we have observed experimentally, such a day-night pattern appears usually in a clearer fashion, similarly to what can be observed, for example, in the

case of MC6 on Fig.5.24.

5.5.5 Cloud of Allaple Worms

$MC5$ is another worm-related cloud. Indeed, the arrival rate (on a daily basis) of the sources involved in the 38 μ-events have a temporal pattern typical of a network worm: there is a *worm outbreak* day, on which we see a steep rise in the number of observed sources, followed by an exponential growing trend up to a maximum number of infected machines. After this, we observe a slow decreasing rate, corresponding to the period where the number of machines that are being cured is larger than the number of new infections. This is somehow visible on Fig. 5.25 that represents the time series of some μ-events involved in $MC5$, as observed on 7 different platforms. According to our measurements, the outbreak of this worm started around November 5th, 2006.

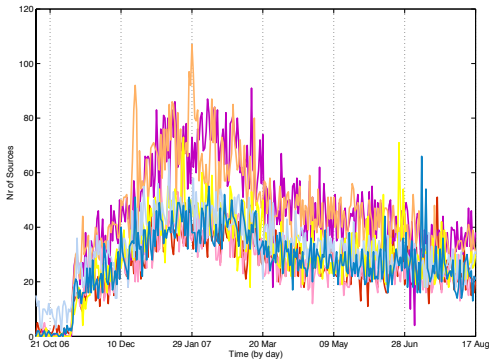


Figure 5.25: Some time series of μ-events belonging to MC5 (related to W32/Allaple.B), observed on 7 platforms.

The targeted ports and services of this cloud are mainly ICMP, sometimes followed by exploits on 139T and 445T (note that the *port sequence* is different from those performed by $MC1$: in this case, 139T is targeted first by the worm). A search on the Web quickly revealed that a network worm called **W32/Allaple.B** [148] (or RaHack.W [156]) was propagating using those different ports. *Allaple* is a polymorphic worm for Windows platforms that spreads to other computers by exploiting common buffer overflow vulnerabilities, including: SRVSVC (MS06-040), RPC-DCOM (MS04-012), PnP (MS05-039) and ASN.1 (MS04-007), and by copying itself to network shares protected by weak passwords.

A closer look at the network traffic collected by our honeypots quickly confirmed this intuition, since the worm has a very specific signature within the payload of the ICMP packets sent as probes to find other alive hosts (i.e., **B**abcdefgh. . . - see the Snort signature in Fig. 5.26 as provided by EmergingThreats[6]). However, it is worth noting that no predefined signature was needed to group all those μ-events in the same misbehaving cloud, thanks to our method that relies on the correlation (and combination) of multiple features, even contextual ones such as spatial distributions. Furthermore, all ICMP packets involved in other MC's have different payloads, which confirms the effectiveness of the approach for the attribution of events to phenomena having the same root cause.

[6] http://www.emergingthreats.net/rules/emerging-virus.rules

The lifetime of this Allaple-related cloud is about 750 days. More than 35 thousand sources infected by Allaple have been observed by 14 different platforms (located in 8 different Class A-subnets), to be finally attributed to the same root cause by our multi-criteria clustering technique. The global geographical distribution of infected machines revealed that a majority of machines was coming from following countries: KR(13%), US(13%), BR(8%), CN(6%), PL(6%), TW(3%), CA(5%), FR(4%), and JP(3%). Another strong advantage of our approach is that, once identified, we can easily follow the evolution of such a worm-related phenomenon (i.e., how it propagates in the Internet).

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET WORM Allaple ICMP
Sweep Ping Inbound"; icode:0; itype:8; content:"Babcdefghijklmnopqrstuvwabcdefghi";
threshold: type both, count 1, seconds 60, track by_src; classtype:trojan-activity;
reference:url,www.sophos.com/virusinfo/analyses/w32allapleb.html;
reference:url,isc.sans.org/diary.html?storyid=2451;
reference:url,doc.emergingthreats.net/2003294;
reference:url,www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/VIRUS/WORM_Allaple;
sid:2003294; rev:6;)
```

Figure 5.26: Snort IDS signature for ICMP Ping packets sent by the Allaple worm.

5.5.6 P2P aberrations

*MC*6 is a very interesting, yet intriguing, cloud. Our technique has grouped together 147 μ-events that have been observed over a period of 573 days. All these events share a number of common characteristics that we have some difficulty to explain:

- The vast majority of these μ-events target a single platform, located in China. A very few μ-events have also hit another platform in Spain.
- The vast majority of these μ-events originate from Italy and Spain only.
- All these μ-events are like *epiphenomena*, i.e., they exist on a single day, and then disappear.
- All these μ-events target a single high, unusual TCP port number, most of them not being assigned to any particular protocol or service (e.g. 10589T, 15264T, 1755T, 18462T, 25618T, 29188T, 30491T, 38009T, 4152T, 46030T, 4662T, 50656T, 53842T, 6134T, 6211T, 64264T, 64783T, 6769T, 7690T)
- These μ-events share a substantial amount of source addresses among them.
- A number of high port numbers correspond to port numbers used by well known P2P applications (e.g., 4662/TCP, used by eDonkey P2P network).

This last remark leads us to hypothesize that this extremely weird type of attack traces may have something to do with *P2P traffic aberrations*. It can be a misconfiguration or, possibly, the side effect of a deliberate attack against these P2P networks, as explained in [105, 46], in which the authors argued that it is possible to use P2P networks to generate DDoS attacks against any arbitrary victim.

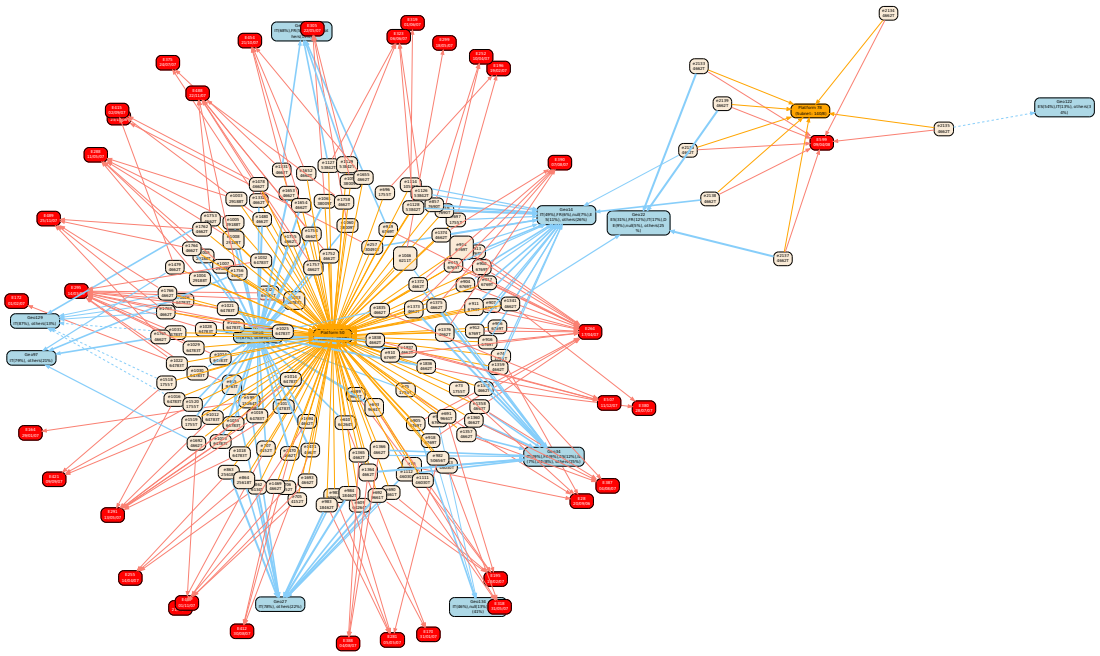


Figure 5.27: Visualization of the μ-events (nodes in antique white) composing $MC6$. Geographical clusters of μ-events (F_{geo}) are depicted in blue, clusters of originating subnets (F_{sub}) are in pink, and the targeted ports (F_{ps}) are in the node labels. In the yellow nodes, we have represented the targeted platforms, and in the red nodes, the dates of the corresponding $\mathcal{M}$-events (which is obtained from the feature F_{time}).

In Fig. 5.27, we have created a node-link graph to easily visualize $MC6$. As one can see, almost all μ-events are targeting a single platform (number 50), at many different points in time. The μ-events are grouped into 10 geographical clusters; however, the distributions of countries share many commonalities (the sources originate mostly from Italy and Spain).

Finally, Fig. 5.24 highlights the fact that these 147 μ-events are not randomly distributed over the hours of the week but instead, they seem to exist on a limited number of recurrent moments.

All these elements tend to demonstrate the meaningfulness of grouping all these, apparently different, attack events. Even if we are not able, at this stage, to provide a convincing explanation related to their existence, our method has, at least, the merit of having highlighted the existence of this, so far unknown, phenomenon.

It is our hope that other teams will build upon this seminal work and our preliminary findings, to help all of us to better understand these numerous threats our approach has helped to identify.

5.6 Summary

This Chapter has illustrated the application of our attack attribution method to a data set of network attack traces collected by worldwide distributed honeypots during more than two years. We have described in details each step the method goes through, and we have presented several experimental results to demonstrate the utility and meaningfulness of the

approach. In particular, we have performed a more in-depth analysis of several instances of *Misbehaving Clouds* found by our multi-criteria clustering approach.

In the next Chapter, we will show how the very same method can be used to analyze a completely different data set made of malicious web domains that aim at selling fake anti-virus software, also called *rogue AV domains*. Thanks to different website features that we have observed from more than five thousand rogue domains in the Internet, we will show how a multi-criteria analysis of this data set can give insights into the *modus operandi* of cybercriminal organizations which are responsible for setting up rogue AV *campaigns*.

Chapter 6

Application to Rogue Security Software

"Experience without theory is blind,
but theory without experience is mere intellectual play."
– Immanuel Kant

One of the requirements in the design of our attack attribution method is its applicability to a broader set of problems relating to Internet threats, intelligence analysis, or more generally to the analysis of any security data set.

In this Chapter, we demonstrate how the very same method can be used to address another emerging security problem, namely *rogue security software*, which is very different from the phenomena that we have previously analyzed by means of honeynet data. A rogue security software is a type of misleading application that pretends to be legitimate security software, such as an antivirus scanner. In reality, these programs provide little or no protection and, in fact, may actually install the very malicious code it purports to protect against.

In the following Sections, we describe how we leveraged our multi-criteria clustering method to analyze the *campaigns* through which this type of malware is distributed, i.e., what are the underlying techniques, server infrastructure and coordinated efforts employed by cyber-criminals to spread their rogue software. In the experimental results, we give a more in-depth presentation of some typical networks of rogue domains that are likely linked to the same campaign, which helps to reveal the *modus operandi* of the criminal organizations behind them.

6.1 Introduction

The previous Chapter has presented an in-depth analysis of a data set collected by a worldwide honeypot deployment for the observation of server-side attacks. Thanks to our

attribution technique, we have identified some large groups of malicious sources forming so-called *Misbehaving clouds*, and having a common root cause (e..g, botnet, network worm, etc).

However, in the recent years security analysts have witnessed a certain shift of attention from server-side to *client-side* attacks, with a specific focus on attacks targeting vulnerabilities in Web browsers. According to the Symantec ISTR [157] - Volume XIII (2008), over half of the patched medium- and high-severity vulnerabilities in the second half of 2007 were browser and client-side vulnerabilities. In Volume XIV of the same series of reports, it appears that drive-by downloads are becoming an increasingly dominant vector of attack.

This has motivated researchers of the **WOMBAT** Project[1] to collect detailed information on this increasingly prevalent attack vector. Starting from June 2009, WOMBAT researchers have been observing and tracking a large number of malicious websites possibly offering this kind of fake security software. This led to the construction of a data set called HARMUR, which is based on an information tracker that collects different features characterizing various aspects related to suspicious web domains. Among those domains, not only those hosting or delivering rogue software are being tracked, but actually any domain that was reported to host malicious content or client-side threats (e.g., browser exploits, trojans, spywares, etc).

In this Chapter, we turn our attention to the detailed analysis of a specific data set made of 5,852 rogue web sites, as observed by HARMUR during a two-month reporting period (July to August, 2009). The primary goal of this analysis is to identify the server-side components of rogue AV *campaigns*, which refer to the coordinated efforts (in terms of server infrastructure, malware code, and registration strategies) made by criminals to distribute their rogue AV.

One assumption that can reasonably be made is that a campaign is managed by a group of people, who are likely to reuse, at various stages of the campaign, the same techniques, strategies, and tools (for obvious reasons of development cost). Consequently, we have applied the very same multi-criteria clustering method to this specific data set, with the purpose of identifying any emerging patterns in the way rogue domains are created, grouped, and interconnected with each other, based upon common elements (e.g., rogue AV-hosting sites, DNS servers, domain registration) that are likely due to the same root cause, i.e., the same rogue AV campaign.

Finally, in Section 6.5, we provide a more in-depth presentation of some typical networks of rogue domains found experimentally by our attack attribution method. Note that some of these experimental results have been included in the *Symantec Report on Rogue Security Software*, published mid-October 2009 [159], and presented at the 5^{th} European Conference on Computer Network Defense (EC2ND) in November 2009 [36].

6.1.1 Rogue AV ecosystem

To introduce the problem, we start by providing a brief overview of some noteworthy characteristics related to rogue security software and its ecosystem, as observed globally by security companies and researchers.

[1] Worldwide Observatory of Malicious Behaviors and Threats - http://www.wombat-project.eu

A rogue security software program is a type of misleading application that pretends to be a legitimate security software, such as an anti-virus scanner, but which actually provides the user with little or no protection. In some cases, rogue security software (in the following, more compactly written *rogue AV*) actually facilitates the installation of the very malicious code that it purports to protect against ([159]).

Rogue AV makes its way on victim machines in two prevalent ways. First, social engineering techniques, such as Web banner advertisements, pop-up windows and attractive messages on blogs or sent via spams, can be used to convince unexperienced users that a rogue tool is free and legitimate and that its use is necessary to remediate often inexistent threats found on the victim's computer (hence, the other name *scareware* given to those programs [159])). A second, more stealthy technique consists in attracting victims to malicious web sites that exploit vulnerabilities in the client software (typically, the browser or one of its plugins) to download and install the rogue programs, sometimes without any user intervention (i.e., via *drive-by* downloads).

Rogue AV programs are distributed by cyber-criminals to generate a financial profit. In fact, after the initial infection, victims are typically tricked into paying for additional tools or services (e.g., to upgrade to the full version of the program or to subscribe to an update mechanism), which are of course fictitious and completely ineffective. For the victims, the initial monetary loss of these scams ranges from $30 to $100. Some examples of prevalent rogue security applications (as reported by Symantec for the period July 2008 - June 2009 [159]) are known as SpywareGuard 2008, AntiVirus 2008, AntiVirus 2009, Spyware Secure, and XP AntiVirus.

Despite its reliance on relatively unsophisticated techniques, rogue AV has emerged as a major security threat, in terms of the size of the affected population (Symantec's sensors alone reported 43 million installation attempts over a one-year monitoring period [159]), the number of different variants unleashed in-the-wild (over 250 distinct families of rogue tools have been detected by Symantec [159]), and the volume of profits generated by cyber-crooks. Their business model relies on an affiliate-based structure, with per-installation prices for affiliate distributors ([77, 159] reported earnings of as much as $332,000 a month in affiliate commissions alone, as observed on a distribution website called `TrafficConverter.biz`).

The prevalence and effectiveness of this threat has spurred considerable research by the security community [159, 112, 113]. These studies have led to a better understanding of the technical characteristics of this phenomenon (e.g., its advertising and installation techniques) and of the quantitative aspects of the overall threat (e.g., the number and geolocation of the web sites involved in the distribution of rogue programs and of their victims).

However, a number of areas have not been fully explored. Indeed, malware code, the infrastructure used to distribute it, and the victims that encounter it do not exist in isolation, but are different aspects of the coordinated effort made by cyber-criminals to spread rogue AV. We refer to such a coordinated activity as a rogue AV *campaign*. As we show in this Chapter, rather than examining single aspects of this phenomenon, we analyzed the rogue campaign as a whole thanks to our multi-criteria clustering method. In particular, we focus on understanding its server infrastructure, and the way it is created and managed.

6.1.2 HARMUR data set

Our experimental data set comes from HARMUR, the **H**istorical **AR**chive of **M**alicious **UR**ls, which is a recent initiative developed in the context of the WOMBAT Project [188, 189]. HARMUR is an information tracker that aims at collecting and storing detailed information on the nature, structure, and evolution of Web threats, with a special focus on their historical dimension.

For every monitored web site, the HARMUR repository gathers information on the possible hosting of browser exploits and *drive-by downloads*, which occur when a user visits a malicious website (or a legitimate website that has been compromised), and malicious code is downloaded onto the user's computer without the user's interaction or authorization. This type of attacks usually implies the exploitation of vulnerabilities in browsers, or browser plug-ins.

Instead of developing new detection technologies (e.g., based on honeyclients, or special web crawlers), HARMUR integrates multiple information sources and takes advantage of various data feeds that are dedicated to detecting Web threats. By doing so, HARMUR aims at enabling the creation of a "big picture" of the client-side threat landscape and its evolution. We are particularly interested in studying the evolution over time of the threats associated to web sites, to get a unique perspective on the modus operandi of attackers exploiting this class of attack vectors for malware distribution.

A prototype of the HARMUR information tracker has been running since June 2009. Since then, it has collected information on the following site or domain features:

- **Norton Safeweb**. Thanks to Symantec's *Norton Safeweb*[2], a web site reputation service, HARMUR gathers detailed information on known web threats identified on each suspicious web site being tracked.

- **Google Safebrowsing**. To double-check the malicious aspect of web sites, HARMUR relies also on blacklisting information available from the Google Safebrowsing service[3] .

- **DNS mapping**. HARMUR keeps track of DNS mappings between domain names, corresponding authoritative nameservers, and server IP addresses (i.e., addresses of HTTP servers hosting suspicious domains).

- **Whois information**. For each web site, *Whois* registration information for the domain name is gathered and stored in HARMUR.

- **Geolocation and Autonomous System**. Information on the physical location (in terms of countries) and network location (in terms of Autonomous System, or AS) is collected for each nameserver and HTTP server associated to a web site.

- **HTTP server status**. When available, information on the reachability of the web servers, and on their software version (as advertised in the HTTP headers) is stored in the repository.

[2] http://safeweb.norton.com
[3] http://code.google.com/apis/safebrowsing/

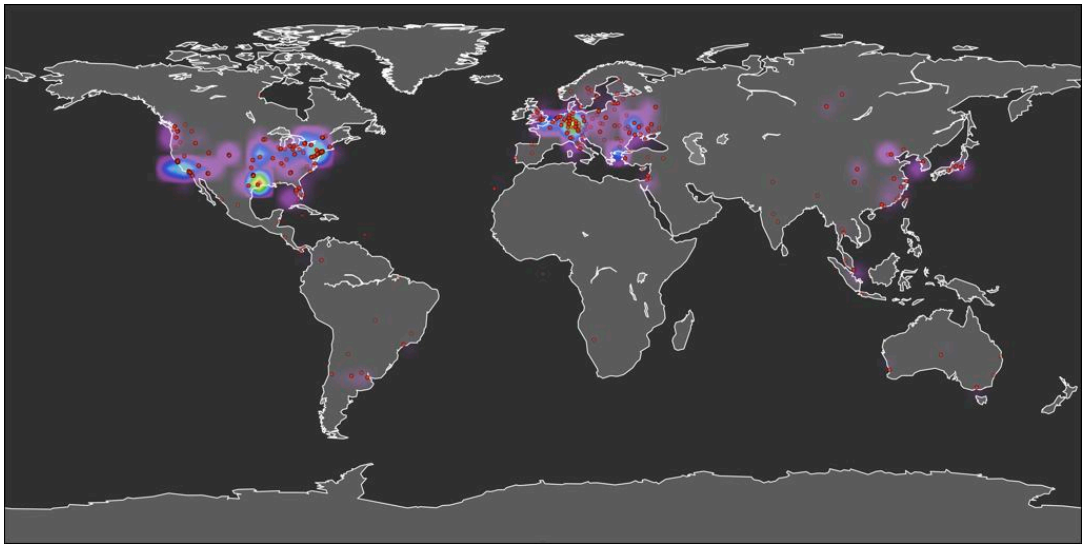


Figure 6.1: Map of rogue AV servers, as monitored by HARMUR over a period of two months, in July and August 2009.

The HARMUR data set is thus centered on the notion of **site**, i.e., a certain web page identified by a URL. The data collection process consists of interleaving a periodic operation of *site feeding*, to identify new interesting URLs to be included and consequently monitored in the dataset, with the process of *site analysis* by means of different analysis feeds, which provide information on the threats associated to HARMUR-monitored sites.

Site analysis is based on a scheduling policy, that determines when a given site needs to be re-submitted to the analysis feeds. The scheduling policy allows to define the frequency by which a site is analyzed, according to its current relevance and interest.

Experimental data set

The HARMUR data set we have considered for this analysis was collected over a period of approximately two months, in July and August 2009. The rogue AV-hosting servers were identified through a variety of means, including automated and manual feeds.

To build our experimental data set, we considered 5,852 DNS entries pointing to 3,581 distinct IP addresses of web servers that were possibly hosting rogue security software. After analysis, the 3,581 Web servers have been broken down into the following categories:

- 2,227 Web servers (as identified by their unique IP addresses) were hosting domains serving only rogue security software,
- 118 servers hosted rogue security software along with domains that served malicious code,
- the remaining IP addresses served malicious code along with innocuous domains.

Regarding the geographical distribution of these servers (Table 6.1), we observed that 53% were in the USA, far more than any other country. The high ranking of the US may be due to the methods used for identifying rogue AV sites, which more easily identified

English-language sites than sites marketing scams in other languages. Germany ranked second in this survey, accounting for 11% of the total servers hosting rogue security software identified. Fig. 6.1 graphically depicts the geographical distribution of rogue AV servers, where each red dot represents a distinct server, while the different gradients on the background underline the areas with highest density of deployed servers.

Table 6.1: Geographical distribution of rogue AV servers, as monitored by HARMUR over a period of two months, in July and August 2009.

Rank	Country	Percentage
1	United States	53%
2	Germany	11%
3	Ukraine	5%
4	Canada	5%
5	United Kingdom	3%
6	China	3%
7	Turkey	3%
8	Netherlands	2%
9	Italy	2%
10	Russia	1%

6.2 Selection of site features

We turn now to the application of our attack attribution method to this data set of 5,852 rogue AV websites. As described previously, we want to identify emerging patterns in the way domains (and servers) are grouped and interconnected with each other, based upon a series of common elements.

As usual, we start by analyzing which elements, or *site features*, can give us relevant viewpoints on those rogue AV phenomena. Some illustrative examples of these features are summarized in Table 6.2 for a subset of rogue domains monitored by HARMUR. We further detail the selection and the meaning of those network-related features hereafter.

6.2.1 Server IP addresses

Every web site (or web domain) has to be hosted on a publicly available Web server, to which an IP address has been assigned. The mapping between a web site and the server IP address is maintained via the Domain Name System (DNS), a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network. Via specific requests to DNS servers, one can know the IP address of the server hosting a given domain name, as well as the *authoritative nameserver* (i.e., the DNS server responsible for keeping the records of the domain name).

As a result, if cyber-crooks want to distribute their rogueware to innocent victims, they have to *i)* find some hosting web server with a publicly available IP address; and *ii)* register their domain names and let them point to those server IP addresses. Due to the fraudulent aspect of their business, we could a priori believe that cyber-criminals

would host rogue AV websites on compromised computers (e.g., on zombie machines that are part of a botnet), as it is often the case with phishing pages, illegal porn and *warez* websites. However, our experimental analysis tends to show that a large majority of those rogue domains are hosted by some popular commercial domain registrars and web hosting companies (e.g., GoDaddy, eNom, Tucows, OnlineNIC, etc).

Assuming that cyber-criminals want to make their efforts profitable, they will probably look for a certain type of hosting infrastructure that is not too expensive, somehow "reliable" for them (i.e., offering some oversight regarding the registration of suspicious names, and slow in taking down fraudulent domains), and possibly allowing to automate certain administrative tasks, such as the bulk registration of new web domains. Furthermore, due to the affiliate structure and the social networking aspect of those criminal organizations, we hypothesize that many affiliates belonging to the same community will most likely reuse the same tools in order to quickly create new domains, distribute rogue AV and thus make some profit.

Consequently, our intuition about a rogue AV campaign is that cyber-crooks of a same organization will, at various stages of the campaign, reuse the very same techniques to create and register their domains. They may also choose for a specific domain registrar or web hosting company, for the very same reasons explained here above. In other words, when two rogue domains are hosted on the same web server at a given point in time, this can *eventually* be an indication of domains involved in the same overall campaign.

However, to reduce the cost of ownership, most web hosting companies offer some shared hosting on *server farms*, which means that two domains registered during the same campaign can perfectly be hosted on two different servers of the same Web company (e.g., GoDaddy), having thus nearby IP addresses (i.e., located within the same IP subnet). For this reason, we observed that it is sometimes useful to group server IP addresses by Class C or Class B-subnet, such that we can compare the IP subnets on which two rogue sites are located, instead of the exact IP addresses (see for example domains 465709 and 465706 in Table 6.2 on page 145).

It is also worth noting that some rogue AV domains were observed as being hosted on more than one server, which may be an attempt to reduce the effectiveness of mitigation measures such as IP blocking or blacklisting servers, by providing a certain level of redundancy with spare IP addresses being reserved for a given domain. That is, when cyber-crooks detect an attempt of blocking certain IP addresses, they just need to activate a spare IP address and let their rogue domain point to this new address (i.e., by changing the 'A' record of the rogue domain in the nameserver).

In conclusion, the considerations here above lead us to define the following *site features*, which can be used to link rogue domains to the same campaign:

- F_{IP}, which represents the IP address(es) of the web server(s) hosting a given rogue domain. The corresponding feature vector is thus a set of server IP addresses;

- $F_{Cl.C}$, which represents the Class C-subnet(s) of the web server(s) hosting a given rogue domain. The corresponding feature vector is thus a set of Class C-subnets;

- $F_{Cl.B}$, which represents the Class B-subnet(s) of the web server(s) hosting a given rogue domain. The corresponding feature vector is thus a set of Class B-subnets;

- F_{NS}, which represents the IP address(es) of the authoritative nameserver(s) for a given rogue domain. The corresponding feature vector is thus a set of nameserver IP addresses.

Finally, it is important to note that none of these network observables *by itself* is considered as sufficient to attribute with high confidence two rogue domains to the same campaign. Indeed, using a specific web hosting provider, or pointing to the same web server, does not *necessarily* mean that the hosted domains are part of the same rogue campaign (and in fact, many legitimate web sites are being hosted on the very same servers as those of rogue sites). Additional features are thus needed to bring a stronger evidence of two domains having a common root cause.

6.2.2 Whois information

Whois is a query/response protocol that is widely used for querying databases in order to determine the registrant or assignee of Internet resources, such as a domain name, an IP address block, or an autonomous system number[4]. By performing periodically Whois lookups, HARMUR can retrieve and store some valuable information about each web site being monitored, such as the *registrant* name (usually, an email address), the domain *registrar*, the geolocation of the hosting web server, and the creation date of the domain.

For the very same reasons as those stated here above, we hypothesize that two domains created by the same criminal organization, for the purpose of running a rogue campaign, will have commonalities in one or several *Whois* features (i.e., same registrant address, or same registrar and creation date for domains that are create in bulk using automated tools).

This leads us to define the following site features related to the available Whois information:

- F_{Reg}, which refers to the name or email address of the registrant;
- F_{Rar}, which refers to the name of the Registrar;
- F_{Geo}, which refers to the geolocation of the web server hosting a given domain (i.e., a country);
- F_{Crea}, which refers to the creation date of the domain, as given in the registration data.

As with the previous ones, these features can be used to link rogue domains to the same campaign, but none of them is sufficient by itself to identify a rogue campaign in a reliable fashion.

6.2.3 Domain names

When cyber-crooks decide to create dedicated web sites for hosting rogue AV software, we observed that, in many cases, they tend to choose some appealing names that can lure users into believing that their web sites are genuine and legitimate. Furthermore, the domain names can also be chosen so as to be consistent with the "brand name" given to

[4] http://en.wikipedia.org/wiki/WHOIS

Site Id	F_{Dom}	F_{IP}	$F_{Cl.C}$	$F_{Cl.B}$	F_{NS}	F_{Reg}	F_{Rar}	F_{Geo}	F_{Crea}
271665	windowsantivirus2008.com	74.54.82.219, 209.62.20.233	74.54.82 209.62.20	74.54 209.62	74.54.82.119	domadmin@privateregistrations.ws	DIRECTI	US	2008-06-04
271621	Xp-2008-Antivirus.com	208.73.210.27, 208.73.210.121	208.73.210	208.73	204.13.161.55, 204.13.160.55	-	-	US	-
272656	malwaredefender2009.com	67.43.237.75, 211.95.73.189	67.43.237 211.95.73	67.43 211.95	208.76.62.100, 75.102.60.66	jsfsl2341@googlemail.com	Regtime Ltd.	CN	2009-03-04
211552	anti-malware-2010.com	74.205.8.7	74.205.8	74.205	216.69.185.2, 208.109.255.2	ANTI-MALWARE-2010.COM@domainsbyproxy.com	GODADDY.COM	US	2009-05-31
122287	antivirus360remover.com	174.132.250.194	174.132.250	174.132	207.218.223.162, 207.218.247.135	ANTIVIRUS360REMOVER.COM@domainsbyproxy.com	GODADDY.COM	US	2009-02-22
272539	norton-antivirus-2010.com	74.208.42.60, 74.208.156.41, 82.165.245.27	74.208.42, 74.208.156, 82.165.245	74.208, 82.165	74.208.3.8, 74.208.2.9	proxy1994891@1and1-private-registration.com	GODADDY.COM	US	2007-07-08
272540	nortonantivirus2010.com	69.64.145.229, 209.249.222.18, 208.116.34.163	69.64.145 209.249.222 208.116.34	69.64 209.249 208.116	209.249.221.130, 72.34.41.47, 74.81.64.51	support@NameCheap.com	ENOM	US	2006-08-13
334096	home-antivirus2010.com	72.52.210.132	72.52.210	72.52	76.73.35.154, 72.52.210.132	blair@8081.ru	ONLINENIC	US	2009-07-14
334091	homeanti-virus-2010.com	72.52.210.130	72.52.210	72.52	76.73.35.155, 72.52.210.130	blair@8081.ru	ONLINENIC	US	2009-07-14
389838	homeav-2010.com	72.52.210.133	72.52.210	72.52	76.73.35.158, 72.52.210.133	tours@infotorrent.ru	ONLINENIC	US	2009-07-14
465709	pc-anti-spyware-2010	174.139.5.50, 209.31.180.235	174.139.5, 209.31.180	174.139, 209.31	174.139.5.50, 209.31.180.235	argue@8081.ru	ONLINENIC	US	2009-07-29
465706	pc-anti-spy2010.com	174.139.243.45 209.31.180.233	174.139.243 209.31.180	174.139 209.31	174.139.243.45, 209.31.180.233	pixie@ml3.ru	ONLINENIC	US	2009-07-29
465710	p-c-anti-spyware-2010.com	174.139.243.46, 209.31.180.234	174.139.243 209.31.180	174.139 209.31	174.139.243.46, 209.31.180.234	kites@e2mail.ru	ONLINENIC	US	2009-07-29

Table 6.2: Network observables used as *site features*, as collected by HARMUR for a set of rogue domains and the associated web servers.

their rogue software, for example windowsantivirus2008.com, xp-2008-antivirus.com, malwaredefender2009.com, or pcregistrycleaner.com.

Consequently, a good reason to look at domain names patterns is that rogueware distributors and their affiliates can rely on the same software tools to quickly create new domain names in an automated fashion, e.g. by combining (randomly or with a given logic) some tokens that are commonly used in the names of legitimitate anti-malware products (e.g., XP, anti, virus, spyware, malware, registry, home, cleaner, defender, etc).

Identifying patterns and commonalities among domain names may thus give us some indication on the tools or techniques that are being reused by criminals during the same campaign when creating new domains. We denote this site feature as F_{Dom}.

6.2.4 Other possible features

Some other network observables, or features, could be useful in the identification process of rogue AV campaigns. While we haven't directly used those features in the multi-criteria clustering process, we believe that they can bring additional interesting viewpoints on groups of domains attributed to a common phenomenon.

Software version

A first additional feature we may consider is the *software version* of the HTTP server hosting the rogue domain. This feature can be obtained either from the HTTP header sent by the server, or sometimes also from Whois servers. Some possible values of server version are, for example, "Apache/2.2.3 (Red Hat)", "Microsoft-IIS/6.0", but also "nginx/0.6.35", "lighttpd/1.5.0", "LiteSpeed", or eventually some other not-so-commonly-used software like "Oversee Turing" and "gws".

As such, two domains hosted on different web servers running the same HTTP software may not indicate anything useful if we consider this feature alone. However, looking at the distribution of software versions for a group of domains that we suspect of being part of the same phenomenon may help us to confirm the results. More precisely, we see two main reasons for this:

- if cyber-criminals are hosting their rogue web site on compromised servers (as they don't want to pay a commercial provider for hosting hundreds of domains), chances are high that they will reuse the very same technique (or exploit) to take advantage of a given vulnerability they discovered in a *specific version* of a server software (e.g., an exploit only targeting Apache/2.2.3 running on Windows platforms).

- cyber-criminals can also decide to hire zombie machines that are part of a botnet to host their server infrastructure. In this case, the available server software will depend on the bot software used to control the compromised machines, and in most cases, the web server is then based on some custom, lightweight server software that can be easily downloaded and installed, in a stealthy manner, on zombie machines with high-speed connections and public IP addresses. Since the bot herder has usually full access to the underlying operating system of the bots, he can even configure the HTTP server software in such a way that standard banners are replaced by a stealthier one (of his own choice).

Threat type

Another useful feature that we could include in the analysis of rogue AV campaigns is the *type of threats* found on each rogue web site, such as possible browser exploits, trojan downloads, obfuscated (malicious) javascripts, or fake codecs installation. This kind of information is available via web site reputation services, such as *Norton SafeWeb*, which is used as analysis feed by HARMUR. Those reputation services usually rely on high-interaction honeypots to effectively detect these client threats.

In our measurements, only a small fraction of the monitored rogue domains also involved the hosting of additional threats, such as browser exploits, malware downloads, etc. However, we hypothesize that using this feature in combination with the others may be helpful to identify and characterize malicious domains that are controlled by the same group of criminals, as they will quite likely reuse the same techniques to distribute their malware, at least as long as those techniques are effective for infecting new victims.

6.3 Graph-based clustering

Based on the feature analysis and our domain knowledge, we have selected the following 6 features for the iterative application of our *graph-based clustering* component: F_{Dom}, F_{IP}, $F_{Cl.C}$, $F_{Cl.B}$, F_{Reg} and F_{NS}.

The other features were not selected mainly for two reasons: *(i)* for certain aspects, the collected data may be, at this stage of the HARMUR project, either too generic or incomplete to be used as clustering feature (like threat types or Whois information, also because some web sites may not be active any more); and *(ii)* features like F_{Geo} and F_{Rar} are somehow redundant with other selected features, such as the information provided by IP subnets. Indeed, each registrar is managing specific IP blocks allocated to him, and there is a direct mapping between IP addresses and the geographical place.

One could argue that $F_{Cl.C}$ and $F_{Cl.B}$ are also redundant with F_{IP}. However, as explained in previous Section, those features are less specific than F_{IP} but still more precise than F_{Rar}, and can better grasp the fact that rogue domains created during the same campaign can point to different server IP's with nearby addresses (which eventually belong to the same web provider). Moreover, we note that domains registered through the same Registrar on very close dates are apparently hosted on servers with nearby IP addresses (as it is the case with domains 465709, 465706 and 465710 in Table 6.2 on page 145), which further justifies the choice of $F_{Cl.C}$ and $F_{Cl.B}$ as clustering features.

As usual, prior to running the clustering algorithm, we need to define a dissimilarity metric for each site feature.

6.3.1 Distance metrics for measuring pattern proximities

IP addresses

Since feature vectors defined for F_{IP}, $F_{Cl.C}$, $F_{Cl.B}$ and F_{NS} are simply *sets* of IP addresses (or sets of IP subnets), it is relatively easy to calculate a similarity between two sets by using the *Jaccard coefficient* (given by equation 5.1).

For example, looking at domains in Table 6.2 (page 145):

- for domains 465709 and 465706, comparing IP addresses (F_{IP}) with the Jaccard coefficient gives a similarity equal to 0, whereas the same operation for $F_{Cl.C}$ and $F_{Cl.B}$ yields 0.5 and 1 respectively.
- idem for domains 334096 and 334091, where the similarity for F_{IP} is zero, but the IP subnets (Class C and B) are identical.

Registrant names

The most straightforward way of measuring a similarity between two registrant names is simply to check the equality between the two. However, this does not allow us to grasp commonalities in the way that registrant names (i.e., email addresses) have been created, like for example:

- people can use automated tools to create new email accounts, usually offered by specific email providers (such as Gmail, Hotmail, Yahoo, etc), and use them afterwards to automatically register rogue domains;
- we have observed some recurrent patterns in the email addresses of certain registrants, like the use of specific keywords, or string tokens (often borrowed from the domain name being registered), probably also created in an automated fashion;
- a substantial amount of rogue domains have been created using third-party companies offering *domain privacy protection* services (e.g., `domainsbyproxy.com`, `whoisguard.com`, `privateregistrations.ws`, or eNom's "Whois Privacy Protection Service"). Consequently, comparing which specific *email domain* is used by registrants can reflect a certain strategy used by cyber-crooks (in this case, protecting their identity).

These considerations have led us to define a heuristic distance for comparing email addresses used by registrants. More precisely, for two given rogue domains:

(1) we start obviously by checking the equality between the two registrants addresses;

(2) when the value given in (1) is zero, then we further compare the three following sub-features: *email domain*, *username*, *presence of AV-keywords*. The final similarity value is then given by a weighted mean, defined by the following empirical weighting values: [0.2, 0.2, 0.5].

The latest sub-feature refers to the screening of about 30 commonly-used AV-keywords within the email addresses, such as $\{anti, virus, cleaner, remove, malware, spyware, \ldots\}$. When at least 2 different tokens are found, this sub-feature is equal to 1. This simple heuristic distance proved to be effective in grouping registrants addresses that looked semantically related.

Let us consider a few examples from Table 6.2 (page 145):

- for domains 334096 and 334091, the similarity for F_{Reg} is equal to 1:
- for domains 334096 and 465709, the similarity is equal to 0.2 (same domain only);
- for domains 211552 and 122287, the similarity is equal to 0.7 (same domain and presence of at least two AV-related keywords);

Domain names

Regarding F_{Dom}, we need to catch commonalities between rogue domain names having similar patterns, or common sequences of the very same tokens, which can indicate that the very same tool has been used to dynamically create new names based on a given set of keywords. A commonly used technique for measuring the amount of difference between two sequences (or two strings) is the *Levenshtein distance*, also called the *edit distance*, which is used in various domains such as spell checking, information retrieval, text and web mining, or DNS sequence analysis.

The Levenshtein distance is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. It can be considered as a generalization of the Hamming distance, which is used for strings of the same length and only considers substitution edits.

The Levenshtein distance is zero if and only if the strings are identical, and the upper bound is given by the length of the longer string. Let us consider a few examples of distances between rogue domain names we have observed:

- the distance between `scan4lite.com` and `scan4life.com` is only 1, whereas the distance between `scan4lite.com` and `scan4safe.com` is equal to 3;
- the distance between `gofilescan.com` and `gofullscan.com` (resp. `fast4scan.com`) is 2 (resp. 6);
- in Table 6.2, the distance between `home-antivirus2010.com` and `homeanti-virus-2010.com` (resp. `homeav-2010.com`) is 3 (resp. 8);
- in Table 6.2, the distance between `home-antivirus2010.com` and `pc-anti-spy2010.com` (resp. `anti-malware-2010.com`) is 9 (resp. 12).

Since Levenshtein gives a distance (and not a similarity) metric, we still need to tranform those values into similarities, for example by using equation 5.2. Similarly to the calibration procedure performed in Chapter 5 (on page 111), we need to determine a constant σ that reflects the decreasing rate of the similarity as an exponential function of the distance.

By considering a large number of domain names in our data set, we observed that an edit distance of 5 or 6 was in most cases reflecting an *average similarity* between two rogue domains names, with a sufficient number of common tokens justifying a similarity value around 0.5, whereas a Levenshtein distance above 12 was clearly indicating that the two domains had almost nothing in common in their name schemes (or at least, no significant pattern in common). Based on those observations, we have derived an empirical value for $\sigma = 7$. The similarity values obtained from transforming some Levenshtein distances are showed in Table 6.3, which correctly translates our requirements regarding the measurement of similarities between rogue domain names.

Note that some other types of string or text distances could be used to better grasp commonalities among domain names. For example, we could try define a "token-based distance", which computes the number of common tokens between two domain names (based on a predefined list of commonly-used keywords), and normalizes this value by using the Jaccard coefficient (equation 5.1).

Table 6.3: Transforming Levenshtein distances into similarity values using $\sigma = 7$ in equation 5.2.

Levenshtein	0	1	2	3	4	5	6	7	8	9	10	11	12
Similarity	1	0.98	0.92	0.83	0.72	0.60	0.48	0.37	0.27	0.19	0.13	0.08	0.05

Another possiblity would be to rely on more complex distances, such as *semantic matching*, or certain metrics defined in *Latent Semantic Analysis* (LSA), to analyze the possible meaning of each domain name and consequently, to determine whether two domain names are semantically related (i.e., similarly to what a human expert tries to do). However, we leave the study of these options as future work, as the Levenshtein distance has performed very well on this data set, and meaningful clusters have been obtained with this simpler metric.

6.3.2 Cluster analysis of Rogue domains

Based on the previously defined features and distance metrics, we have then applied the graph-based clustering to each individual feature, using the dominant set framework introduced in Chapter 3.

An overview of the clustering results is given in Table 6.4, where we can compare the global performance of site features individually. As usual, the consistency of the clusters can be assessed through the average *compactness* value of all clusters, given by the column $\overline{C_p}$. We have also calculated an average compactness for the first 20 clusters, represented by $C_{p,20}$, as these are usually the most meaningful clusters found by the dominant sets algorithm. The column $\overline{size}$ refers to the average size of the clusters.

First, we can observe that the features related to IP addresses (F_{IP}) or to IP subnets ($F_{Cl.C}$ and $F_{Cl.B}$) provide apparently very compact clusters, i.e., very high C_p (even close to 1). This seems to indicate that rogue AV sites are located in a limited number of IP subnets (between 61 and 73% of all rogue sites could be clustered in only 110 to 192 clusters), and they tend to form very tight clusters. This can be observed in more details in Fig. 6.2 showing the individual compactness values for the first 20 clusters. Furthermore, a large majority of the sites is included in the first 20 clusters, as the evolution of the cluster size seems to show (curve in magenta in Fig. 6.2 (b), (c) and (d)). As we could expect, features related to IP subnets (Class C and B) give fewer clusters than F_{IP}, and they are also larger and more compact.

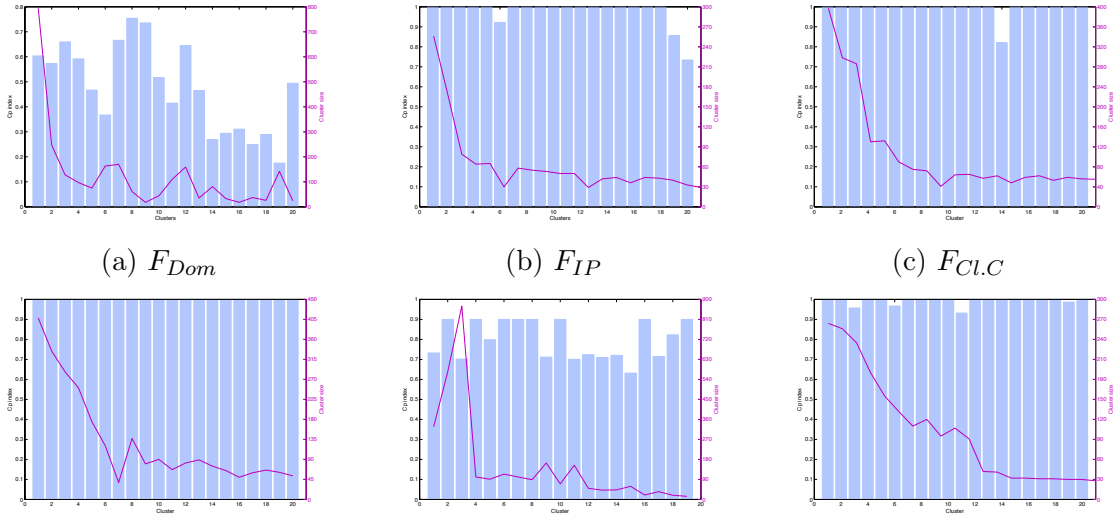
Table 6.4: Overview of the graph-based clustering results for the rogue AV data set.

Feature	**Nr clusters**	**Nr sites**	$\overline{\textbf{size}}$	$\overline{C_p}$	$\overline{C_{p,20}}$
F_{Dom}	132	4,117 (70%)	31.2	0.46	0.53
F_{IP}	192	3,559 (61%)	18.5	0.82	0.98
$F_{Cl.C}$	110	3,667 (63%)	33.3	0.97	0.99
$F_{Cl.B}$	122	4,250 (73%)	34.8	0.98	1
F_{Reg}	19	2,937 (50%)	172.2	0.78	0.78
F_{NS}	40	2,529 (43%)	63.2	0.95	0.99

Regarding *domain names* (F_{Dom}), it is quite surprising to see that about 70% of this data set has been clustered based on commonalities in domain name patterns. The average compactness of the clusters is, admitteddly, a bit lower than for IP addresses, but still acceptable (see also Fig. 6.2 (a)). Still, the largest cluster found for F_{Dom} contains about 800 domain names. These results, in addition to some illustrative patterns of domain name clusters given hereafter, seem to confirm the intuition that miscreants are likely using very effective, automated tools to create large amounts of domain names with common patterns, in a bulk fashion.

Even more surprisingly, only 19 clusters of *registrant names* were found, accounting for 50% of the data set. The mean cluster size for this feature is also the largest one, with (on average) about 170 rogue sites associated with the same registrant(s) ! The individual C_p values of the clusters, showed in Fig. 6.2 (e), indicate that all registant clusters are quite consistent, i.e., there are apparently very good reasons to group those rogue domains based on the registrant names. Here too, we hypothesize that cyber-crooks are able to create new email accounts using highly effective, automated tools, and those new email addresses are then used to create and register new domain names quite anonymously. An in-depth analysis of the cluster patterns further confirmed this assumption, as illustrated hereafter with some detailed results.

Finally, looking at clustering results obtained w.r.t. *nameservers* (F_{NS}), we note that rogue domains are apparently not as much correlated than by the other IP-related features. The number of clusters found with this viewpoint is significantly lower (only 40 groups comprising totally 43% of the sites). However, those clusters are still very compact, but also larger than clusters of IP addresses or IP subnets.



(a) F_{Dom} (b) F_{IP} (c) $F_{Cl.C}$

(d) $F_{Cl.B}$ (e) F_{Reg} (f) F_{NS}

Figure 6.2: Compactness values of the clusters (in blue) and their respective sizes (in magenta) for the first 20 clusters found in the Rogue AV data set (5,852 rogue sites).

Some detailed results

To visualize the clusters, we have created 2-dimensional maps for different features using the same dimensionality reduction technique as used before in Chapter 3 and 5, i.e., t-distributed Stochastic Neighbor Embedding, or *t-SNE* [175].

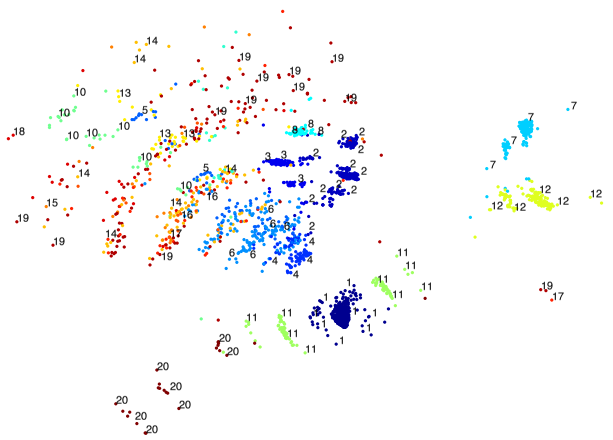
For the sake of illustration, we have considered all rogue sites involved in the largest clusters only, which are the most representative of the kind of results we have obtained. Also, we illustrate some detailed results only for the four features F_{Dom}, F_{IP}, $F_{Cl.B}$ and F_{Reg}. However, very similar visualizations can be obtained for the other two features (F_{NS}, $F_{Cl.C}$), with similar interpretations regarding cluster patterns.

Let us consider Fig. 6.3 (a) (on page 153), on which we can visualize the 20 largest clusters obtained w.r.t. **rogue domain names** (F_{Dom}), and comprising 3,151 rogue sites. On this map, each pixel represents the domain name of a given site, and the pixel color refers to which cluster the site belongs to (notice the numbers around certain points to indicate the cluster id's). The relative interpoint proximities have been mapped to the inter-domain similarities, as calculated with the chosen distance metric (i.e., Levenshtein in this case). As we can observe, the overall structure of the map seems to indicate that there are basically two regions: *(i)* a region with well-separated clusters of domain names (like clusters 1, 11, 20, 7, 12); and *(ii)* a quite large and "messy" zone, where data points are somehow mixed and clusters overlap with each other (like clusters 2, 4, 6, 3, 5, 16, 19, etc).

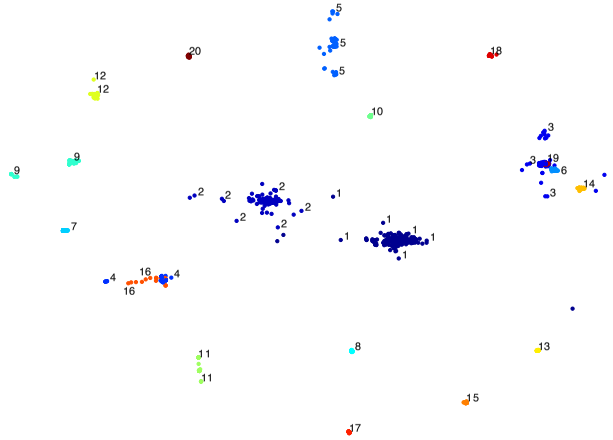
This aspect can be easily explained by looking at the cluster patterns, like those explicitely given in Fig. 6.4 (a). In this table, some domain name patterns are represented using *regular expressions* for the variable parts of the names, whereas fixed string tokens (common to all domains of the indicated cluster) are highlighted in **bold**. This can help the analyst to understand, very quickly and via a single global picture, the inter-relationships among those domain name patterns. For example, cluster **1** (which contains about 794 sites) is made of domain names that are built with exactly 5 randomly-chosen alphanumeric characters, whereas cluster **11** (containing 110 sites) represents domain names made of 7 to 8 alphanumeric characters (most of them also randomly chosen). This does not mean, *per se*, that the two clusters are due to the same root cause (or the same tool); however, it already explains their mutual proximity on the map.

The same reasoning holds for clusters **7** and **12**, where some common string tokens (e.g., **scan** and **.info**) tend to tie those clusters close to each other (idem with clusters **2** and **8**). Regarding the fuzzy area in which we find clusters **4, 6** and many others as we move towards the top of the map, we observed that those overlapping clusters represent domain names with commonly-used words, made of 4 to 10 alphanumeric characters, and involving many *keywords* usually related to anti-malware or anti-spyware software, which explains why those domains are somehow inter-related with many others. In other words, the variability of those patterns combined with the numerous commonalities among some of them explains why the t-SNE algorithm had difficulties to clearly separate those domain names on a reduced map with only 2 dimensions.

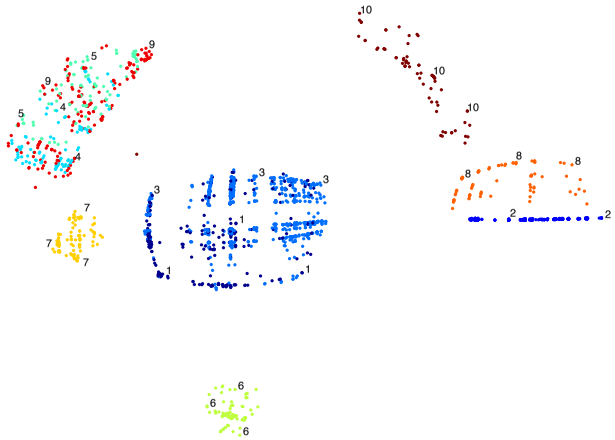
As we could expect from the global clustering results, the two-dimensional maps obtained for the viewpoints related to IP addresses (Fig. 6.3 (b)) and IP subnets (Fig. 6.3 (d)) reflect the strong correlating aspect of those features. In both maps, we have considered the 20 largest clusters, which comprise about 1,269 and 2,589 rogue AV sites for F_{IP} and



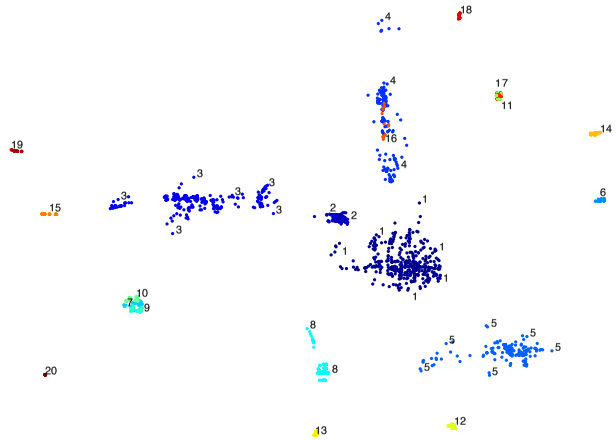
(a) Map of rogue domains clusters (F_{Dom})



(b) Map of rogue AV Server IP's (F_{IP})



(c) Map of Registrant clusters (F_{Reg})



(d) Map of IP subnets (Class B) clusters ($F_{Cl.B}$)

Figure 6.3: Visualizing clustering results for the rogue AV data set.

Id.	Cluster pattern
1	[a-z0-9] {5}**.cn**
11	[a-z0-9] {7, 8}**.cn**
7	**scan**[4|6] {lite, live, home, user ...}**.info**
12	{lite, live, home, user ...} [4|6]**scan.info**
20	{assist, beers, cds, cigars, sofas, ...}**online.cn**
2	{any, av, best, easy, fast, go, lite, ...}**scan.com**
4,6	[a-z0-9] {4, 10}**.com**
10	**adware**{2009, clean, alert, bot, pro, ...}**.com**
8	**goscan**{-pro, data, elite, file, gate, lite, ...}**.com**

(a) F_{Dom}

Id.	Cluster pattern
1	84.16.247.12, 89.149.236.145
2	209.44.126.102
7	209.44.126.241
11	69.64.155.119
4	64.191.92.197
16	64.191.92.197, 91.206.231.146
3	195.95.151.174
19	195.95.151.174, 195.95.151.138, 91.212.41.114

(b) F_{IP}

Id.	Cluster pattern
1,3	[a-z0-9]*__@gmail.com__
2,8	cn@id-private.com, cn@space.kz
4,5,9	[a-z0-9]*__@yahoo.com__
6	contact@privacyprotect.org
7	admin@mas2009.com
10	support@NameCheap.com
17	{ *AV-keywords* }__@domainsbyproxy.com__

(c) F_{Reg}

Id.	Cluster pattern
1	84.16, 89.149
3	195.95
5	63.146
7	209.44
9	209.44
10	209.44, 69.64
4	64.191
16	64.191, 91.206
20	210.51, 220.196, 222.73, 91.212

(d) $F_{Cl.B}$

Figure 6.4: Some cluster patterns found for each feature of the rogue AV data set. For clusters containing multiple string patterns (F_{Dom}, F_{Reg}), variable parts are represented with a syntax based on regular expressions, whereas fixed string tokens are highlighted in bold. For F_{Reg}, *AV-keywords* refers to a set of about 30 AV-related keywords, such as *adware*, *anti*, *malware*, *spyware*, *av360*, *error*, *repair*, *tool*, *virus*, *bot*, *clean*, *registry*, *fix*, *remove*, ...

$F_{Cl.B}$ respectively. Most of the clusters are apparently well-separated. However, we can observe a few overlapping clusters, for example clusters **4** and **16** (in both maps), or clusters **7**, **9**, **10** in the $F_{Cl.B}$ map, which can again be easily explained by the corresponding patterns given in Fig. 6.4 (b),(d). We note also that clusters found w.r.t. $F_{Cl.B}$ are quite obviously much larger than those found with F_{IP}.

Even though IP-related clusters tend to form very tight clusters, we argue that this feature *alone* is probably not sufficient in many cases to identify a rogue AV campaign. In our opinion, all web sites hosted on the same IP subnet (or even on the same web server) are not necessarily created by the very same group of cyber-crooks. There are, apparently, several popular Web providers among those communities (probably for good reasons), and thus the same web server can perfectly host (malicious) domains created by different groups or communities.

Finally, we have represented in Fig. 6.3 (c) the 10 largest clusters obtained with F_{Reg}. Those clusters contain 2,501 rogue sites, and each data point on the map represents here the domain registrant. The patterns corresponding to those clusters are given in Fig. 6.4 (c), where variable parts of the registrants are again represented by regular expressions, and fixed tokens are emphasized in **bold** style. These registrant patterns can explain why we find a very large mixed cluster (1,197 points) in the center of the map, due to two

inter-related clusters (**1** and **3**) that are composed exclusively of email addresses from the gmail.com domain. Idem with clusters **4**, **5** and **9**, but this time within the email domain yahoo.com. As such, observing two domains whose registrants use an email address of the same domain does not mean anything special. However, other clusters are well-separated, showing more interesting patterns like clusters **2** and **8** (cn@id-private.com and cn@space.kz), or cluster **6** and **17** for which we can see that domain owner(s) also protect their privacy by using different *whois domain protection* services.

In conclusion, we note that most of the clusters obtained w.r.t. each site feature can reveal interesting and meaningful patterns revealing how those rogue sites have been created and managed. In fact, each feature can be seen as a viewpoint giving a certain *perspective* on the underlying phenomenon (or root cause), which in turn can also highlight interesting aspects of the modus operandi of the miscreants.

However, it becomes difficult (and time-consuming) to combine those viewpoints *manually* when the number of features increases, even when we rely on graphical visualization techniques such as those presented here. Furthermore, the fact that rogue sites have been clustered w.r.t. *a given aspect*, or even two clusters that are lying in the same neighborhood on a given 2D map, does not mean that the rogue sites under scrutiny are *necessarily* due to the same root cause. As explained previously, only one common feature can be merely due to a coincidence, to a commonly-seen pattern, or to a commonly-used technique.

To identify the underlying phenomena in a more systematic and reliable manner, certain clusters are likely to be merged whereas some others may have to be split. To aid the analyst to make such a decision, our multi-criteria aggregation method can be used to effectively combine all these viewpoints, such that the final result models the expectations of an expert regarding the (combination of) features that must be satisfied in order to attribute two rogue sites to a common root cause.

6.4 Aggregation of all features

6.4.1 Defining parameters

We are now in a position to combine all site features using an *aggregation function*, with the purpose of identifying rogue AV campaigns whose rogue domains are automatically grouped together based upon common elements likely due to the same root cause.

As a first exploratory approach, we have used two different Ordered Weighted Averaging functions (OWA and Weighted OWA) as aggregation means, which have been extensively described in Chapter 4. However, nothing forbids us from performing the very same analysis using more complex aggregation methods, such as a Choquet integral, assuming that we are able to model the behavior of rogue AV phenomena in a more accurate way though. We leave the study of this option as future work.

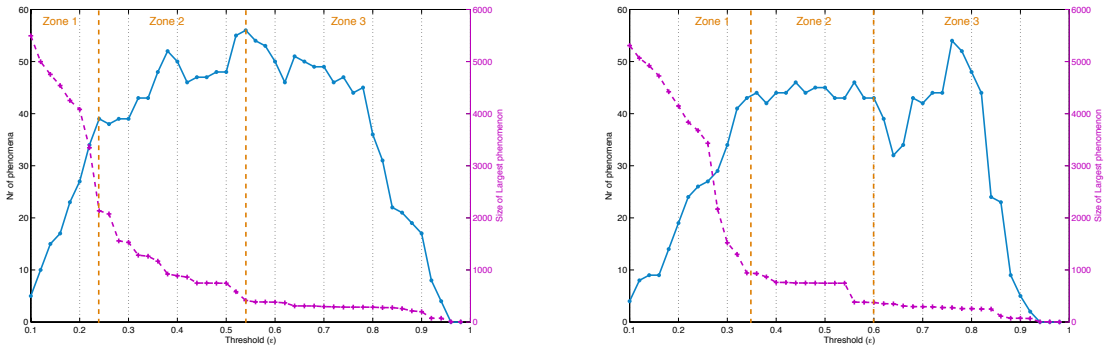
Based on our site feature analysis performed in Section 6.2, and considering our intuition on rogue AV campaigns, we have defined the following weighting vectors to be used in the (W)OWA aggregation:

$$\begin{cases} \mathbf{w} &= [\,0.10,\ 0.10,\ 0.20,\ 0.30,\ 0.20,\ 0.10\,] \\ \mathbf{p} &= [\,0.20,\ 0.20,\ 0.15,\ 0.10,\ 0.25,\ 0.10\,] \end{cases}$$

By defining the weighting vector **w**, we give more importance to criteria starting from the third highest position, which means that the two highest scores will have lower weights (0.10) and thus *at least three* strong correlations will be needed in order to have a global score above 0.3 or 0.4.

Regarding the weighting vector **p**, we give a little more confidence to the features F_{Dom}, F_{IP} and F_{Reg}. Intuitively, we can reasonably assume that a combination of those specific features will yield a high probability that correlated rogue sites are likely due to the very same campaign. On the other hand, we give a little less confidence to $F_{Cl.C}$, $F_{Cl.B}$ and F_{NS}, as these features are obviously less specific, and even somehow redundant with some of the previous features.

It is worth reminding that a strong advantage of these agregation techniques is that the analyst does not need to determine beforehand *which* set of features are the most relevant ones in the aggregation.



(a) OWA (b) WOWA

Figure 6.5: Sensitivity analysis of (a) OWA (b) WOWA aggregation techniques; to determine appropriate ranges of values for the threshold ε. The regions 1, 2 and 3 indicated on the graphs correspond to the 3 phases described in Fig. 5.11. The axis on the right (in magenta) shows the size of the largest phenomenon (or rogue AV campaign) for each threshold value.

The last important parameter to be defined is the **decision threshold** ε, which is used to eliminate unwanted links in the combined graph, and thus also to identify connected components from it. As usual, we can determine an appropriate range of values for ε by performing a *sensitivity analysis*, i.e., we let ε increase from a very low to high value, and we observe the number of largest components (or phenomena) that we obtain in the resulting graph, as well as the size of the largest one. This is illustrated in Fig. 6.5 where we can observe the impact of the threshold on (a) the OWA aggregation method, and (b) on the Weighted OWA. We have indicated on the graphs the three regions of interest in the determination of ε, as described previously in Chapter 5 (on page 120).

In the second region, we can observe a sort of plateau starting around the values $\varepsilon = 0.25$ for OWA, and $\varepsilon = 0.35$ for WOWA, which seems to indicate some reasonable values for our decision threshold (as the number of large phenomena becomes stable). Even the size of the largest phenomenon becomes fairly stable at those threshold values (certainly with the WOWA operator). It should be noted that increasing ε up to an excessive value leads to a significant loss of rogue sites, and thus also of semantics that can be derived

from the phenomena.

6.4.2 Results overview

In Table 6.5, we briefly compare the performance of each aggregation method. Overall, we note that the two sets of phenomena found by the two techniques are consistent with each other, both in terms of size as well as with respect to the composition of each phenomenon. As it could be expected, the WOWA technique performed a little better than OWA, which is due to the use of a second weighting vector $\mathbf{p}$ that models the reliability of site features.

Table 6.5: Comparison of OWA and WOWA aggregation methods for the rogue data set.

Characteristic	OWA	WOWA
Threshold ε	0.30	0.35
$\lvert\mathcal{P}\rvert$	161	173
$\lvert\mathcal{P}\rvert$, with $\lvert P_i\rvert \geq 10$	39	44
Nr of sites	4,049 (69%)	3,586 (61%)
Average C_p	0.46	0.51

To further evaluate the consistency of the results, we have represented in Fig. 6.6 the global graph compactness C_p for the largest phenomena, as calculated individually for each feature. This interesting view can be used to determine which features tend to group rogue sites together within each phenomenon P_i. First, we note that most of the phenomena have globally high compactness values, except for a few ones (such as P_1 and P_3 found with OWA). A deeper inspection of those phenomena reveals that these are also very large connected components, which explains why they are less compact since they are usually made of several loosely connected subgraphs.

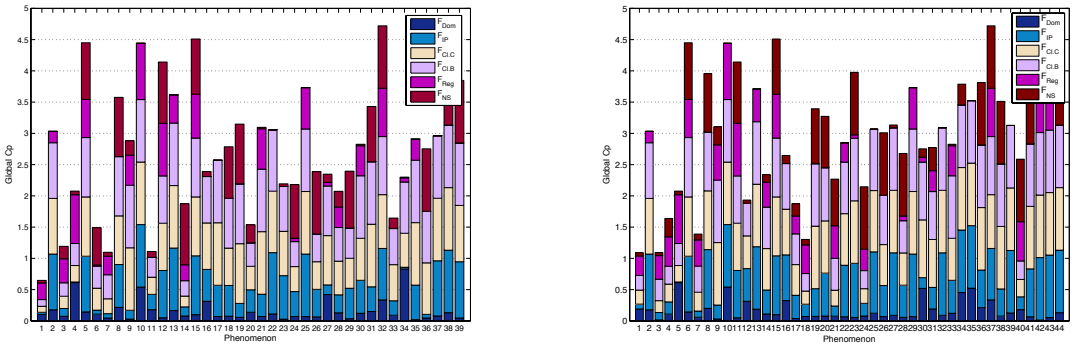
Quite naturally, we see also on Fig. 6.6 that IP-related features contribute the most to the correlation of rogue sites. In many cases, F_{Reg} seems to complete or reinforce those correlations. It is also interesting to see that some phenomena are correlated by $F_{Cl.C}$ and $F_{Cl.B}$, but not by F_{IP} (such as P_{27} found with OWA), which justifies the selection of those features.

F_{Dom} has in general lower C_p values, but there are still 3 or 4 phenomena in which domain name correlation plays a significant role (like for P_4, P_{10} and P_{34} found by OWA). Finally, we observe that each P_i can be characterized by varying degrees of correlation regarding each feature, but overall there are always at least three different features having a high correlation (except for P_1, the largest phenomenon comprising more than one thousand sites).

6.5 Behavioral analysis of Rogue AV campaigns

6.5.1 Introduction

In this final Section, we present a more in-depth analysis of some illustrative case studies, in order to show the kind of insights we can get into the behavior of so-called *Rogue AV*



(a) OWA (b) WOWA

Figure 6.6: Evaluation of the results consistency using the global compactness (C_p) of the largest phenomena found using (a) the OWA aggregation; and (b) the WOWA aggregation. Each color refers to the C_p index of each site feature individually.

Table 6.6: High-level characteristics of some typical Rogue AV campaigns (RC's).

RC	# sites	# Reg.	# Registrar	# IP's	# Class B	Timespan	Server countries
3	438	115	6	50	15	03 Oct 2008 - 03 Jun 2009	UA, CN, KY, SG
4	752	4	1	135	7	22 Jun 2008 - 27 Feb 2009	US, DE, BY
5	310	17	1	13	5	17-20 Oct 2008	CN, DE
27	15	3	1	8	4	25 Jun - 14 Jul 2009	US
34	14	3	1	19	2	29 Jul 2009	US

campaigns (shortly written RC's in the following), which were identified, in a systematic and automated manner, by our multi-criteria clustering technique.

The different RC's that are studied in this Section are summarized in Table 6.6. It is worth mentioning that these experimental results are based on case studies that have been presented in [159, 36].

6.5.2 Two similar campaigns related to Anti-{virus, spyware}-2010

As a first illustrative case study, we present two relatively simple but interesting phenomena identified by our method, namely RC 27 (Fig. 6.7) and 34 (Fig. 6.8). In those figures, the domain names are shown in light blue, the web server /24 subnets in yellow, nameservers in purple, and the email address of the Registrant in red. Double-edged purple boxes indicate servers with co-located DNS and web servers.

Both RC's are composed of a small number of rogue sites, and these are mostly correlated by server IP addresses and by common patterns in the domain names (notice that this is consistent with the assessment of graph compactness in Fig. 6.6). Note also that all domain names are clearly referring to anti-virus or anti-spyware software "products".

Although the two RC clusters initially appear to be distinct, they present a number of similarities:

- Both clusters use the exact same domain naming scheme (except that one uses "spyware" while the other uses "virus");

- All of the domains in each cluster use the same registrar (OnlineNIC) and are serviced by the same two ISPs;
- The email addresses of all domain registrants are in ".ru" domains;
- The servers were on consecutive IP addresses;

Perhaps even more conclusively, we found that the content of each site was identical, with the exception of one differing image (recall that the site content was not a feature of our clustering system).

In fact, cyber-crooks used for both RC's a single registrar (OnlineNIC[5]) which apparently offers the free use of their registration API/template system. The similarities described here above strongly suggest that the task of registering, creating, and hosting these rogue security software domains was automated and that the same entity may be responsible for both clusters.

Also worth noting is that both clusters are split between two different ISPs, suggesting an attempt to provide some level of redundancy in case a cluster is taken offline by the ISP. Finally, we observed that all of the hosting web servers were located in the US.

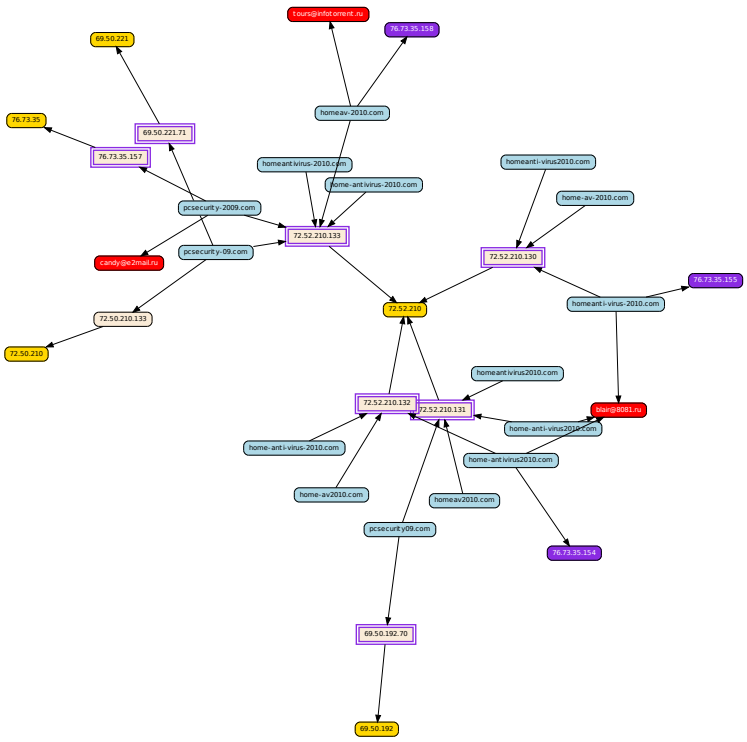


Figure 6.7: RC27: a rogue campaign related to Anti-virus2010.

[5] http://www.onlinenic.com

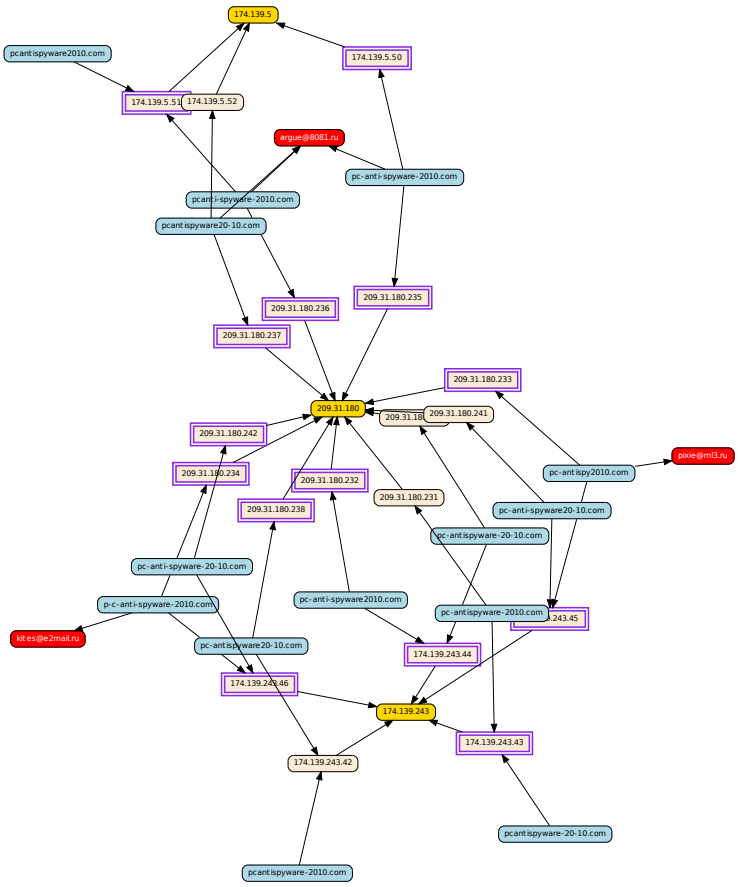


Figure 6.8: RC34: a rogue campaign related to Anti-spyware2010.

6.5.3 Two different campaigns within the .cn TLD

Our attribution method has also identified some other clusters that represent more sophisticated campaigns. Two such examples are RC 4 and RC 5, which are two different campaigns observed within the **.cn** top-level domain (TLD). Regarding cluster RC 5, about 310 .cn domain names were registered in only three days, as represented in Fig. 6.9 on page 163. The domain names (in blue) point to 13 IP addresses residing in five subnets (yellow) and were registered by a number of Web-based email addresses (red) in three days (purple). The prevalent use of popular Web-based email accounts (e.g., `yahoo.com`, `gmail.com` and `hotmail.com`) to register these domains is assumed to be because these email services are easily anonymized.

Interestingly, all of the domain names in RC 5 are referring to various popular web categories, such as games, fun, e-cards, casino and even porn sites, but apparently not a single domain name seems to relate to AV software. However, since they have been included in our rogue data set, they were still somehow related to rogue AV. One could think that some of those web sites are possibly "legitimate" ones that have been compromised, so that they could serve rogue AV software. Note also that we found many of these sites were also hosting malware (e.g., trojans, fake codecs). Considering the abnormal pattern showing the

registration of all those domains "in bulk", a more likely explanation is that *(i)* cyber-crooks try to diversify their activities, by hosting different types of threats, and *(ii)* they want to optimize the monetization of their web sites by attracting as many users as possible with popular web categories. Finally, all of the domains have been registered at a single Chinese registrar (Era of the Internet Technology), and 97% of the web servers are located in China.

In the next cluster, RC 4, about 750 rogue domains have been registered also in the **.cn** TLD (resolving to 135 IP addresses in 14 subnets), on eight specific dates over a span of eight months (Fig. 6.10 on page 164). However, differently from RC 5, the majority of the IP addresses of the hosting servers (pointed to by these domains) were hosted in the United States, Germany, and Belarus. In fact, no server could be identified as being located in China[6].

Like in RC 5, the same Chinese registrar (Era of the Internet Technology) was used by cyber-crooks for the registration of all domain names. However, differently from RC 5, all of the domain names are composed of exactly 5 alphanumeric characters, apparently chosen in a random fashion, which again indicates the use of automated tools to create those domains. Finally, a noteworthy characteristic of this RC is that the registrant responsible for 76% of the domains makes use of a *whois domain privacy protection* service (`cn@id-private.com`), which is also a commonly observed characteristic in certain rogue campaigns.

6.5.4 An affiliate-based rogue network

Our multi-criteria method has identified RC 3, a rogue network showing an even more complex structure, as represented in Fig. 6.11 on page 165. In this cluster, more than 430 rogue sites (in blue) are forming some sort of "bubbles" that are interconnected by common registrants (in red) and common hosting servers or IP subnets (in yellow). We hypothesize that this weird and complex network of rogue AV websites is likely to reflect the affiliate-based structure used by cyber-crooks to propagate rogue security software. Different reasons support this assumption:

- there is a large number of registrants, and most of them are responsible for a single domain;
- the domains are registered at 6 different registrars, which are quite popular for hosting malicious web sites;
- besides rogue AV names, many other domain names are also associated to other web categories that are often used for hosting dubious or suspicious websites, e.g.: online pharmacy (like `pharmacyeasy.cn`), casino (like `smilecasino.cn`), porn sites (like `hot-girl-sex-tube.com`), and there are even a few examples of typo-squatting web sites (like `disenyworld.com` or `rapidhsare.com`). This seems to indicate a *diversification of activities* as performed by different affiliates;

[6]It should be noted that the .cn top-level domain (i.e., the domain designation for China) has no registration restrictions, and non-Chinese based operators can easily register .cn domain names for a very cheap price.

- many other types of web threats have been found on a significant number of those sites (almost 50% of them), which can again reflect that affiliates attempt to monetize their domains by serving other malicious purposes as well;

- in this cluster, the numerous commonalities found among subsets of rogue sites indicate that several groups of affiliates are probably reusing the very same techniques,e.g.: creating and registering new domains at the same registrar or ISP (which does not really care about suspicious domain names), reusing the same browser exploits for hosting drive-by downloads, or serving the same malware or trojan (only 8 unique MD5 were found among all malicious binairies hosted on these web sites).

Also worth noting is that this cluster RC 3 has been observed in a span of time of 8 months. However, we observe once again that most of these sites are being registered in large groups during three phases, each one having a timespan of only a few days, which obviously requires a high level of coordination.

Finally, the geographical location of the web servers has also an interesting pattern, with 42% of the servers in Ukraine (UA), and the rest of the servers is spread in China but also in Cayman Islands (KY), Singapore (SG), and a few of them in Latvia (LV). We note again the prevalent use of Web-based email domains for registrants (more than 40% of gmail.com and about 26% of yahoo.com).

6.6 Summary

In this Chapter, we analyzed different rogue AV campaigns by leveraging our multi-criteria clustering technique. By attributing rogue web sites to a common root cause based upon common elements, we showed how this technique can offer to analysts some unique perspective on how those rogue campaigns and their server-side components are effectively organized, created and managed. By using various server features, we could identify about 40 campaigns out of 5,852 involved rogue sites. An in-depth analysis of some of them was presented to demonstrate the kind of insights we get into the behavior of those miscreants, in a systematic and automated manner.

These results can be leveraged in several ways. First, they give a more explanatory description of the rogue AV threat, in which, for example, individual, disconnected sites are substituted by sets of related sites in which time relationships (e.g., creation dates) are more explicit. Second, campaign-level information reveals the *modus operandi* of the criminals orchestrating the campaign, i.e., how they register the domains, what are their hosting partners, the duration of their efforts, the sophistication of the tools available to them (e.g., to automate the registration of domain names), and the countermeasures they employ against take-down efforts. Finally, the patterns discovered by our multi-criteria clustering analysis could yield means for identifying additional rogue AV sites proactively or reactively.

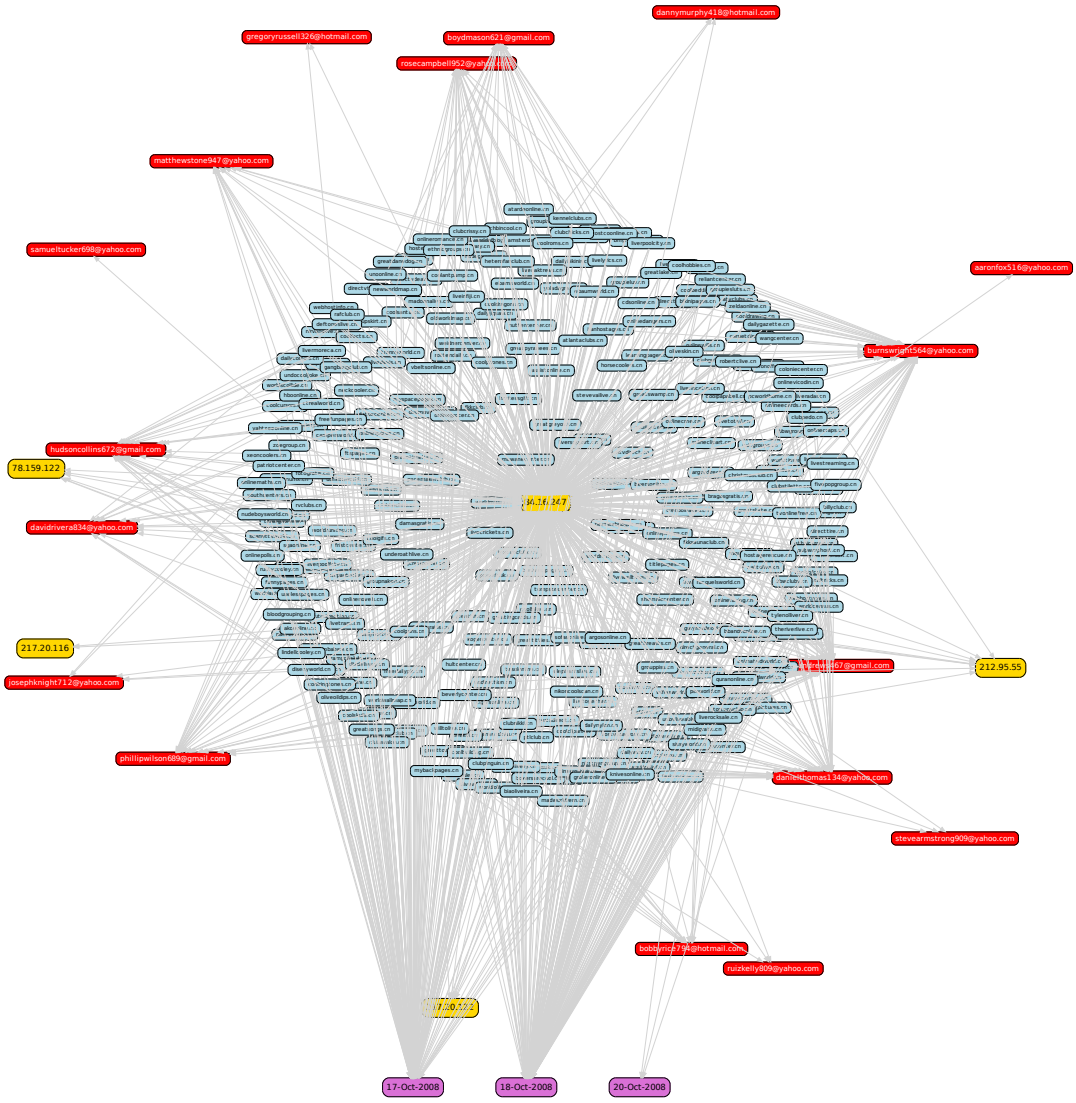


Figure 6.9: RC5: A rogue campaign within the .cn TLD.

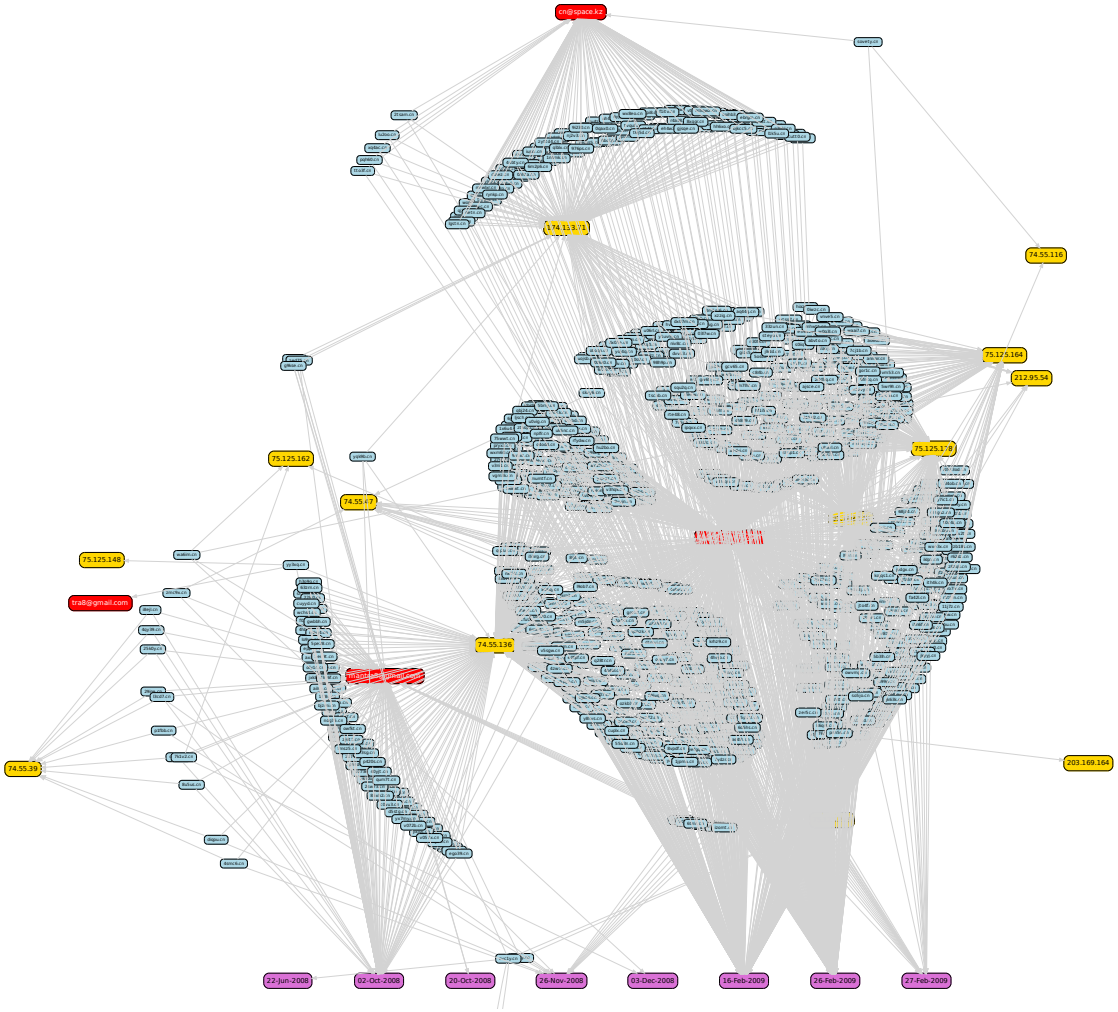


Figure 6.10: RC4: Another rogue campaign within the .cn TLD (different from RC5).

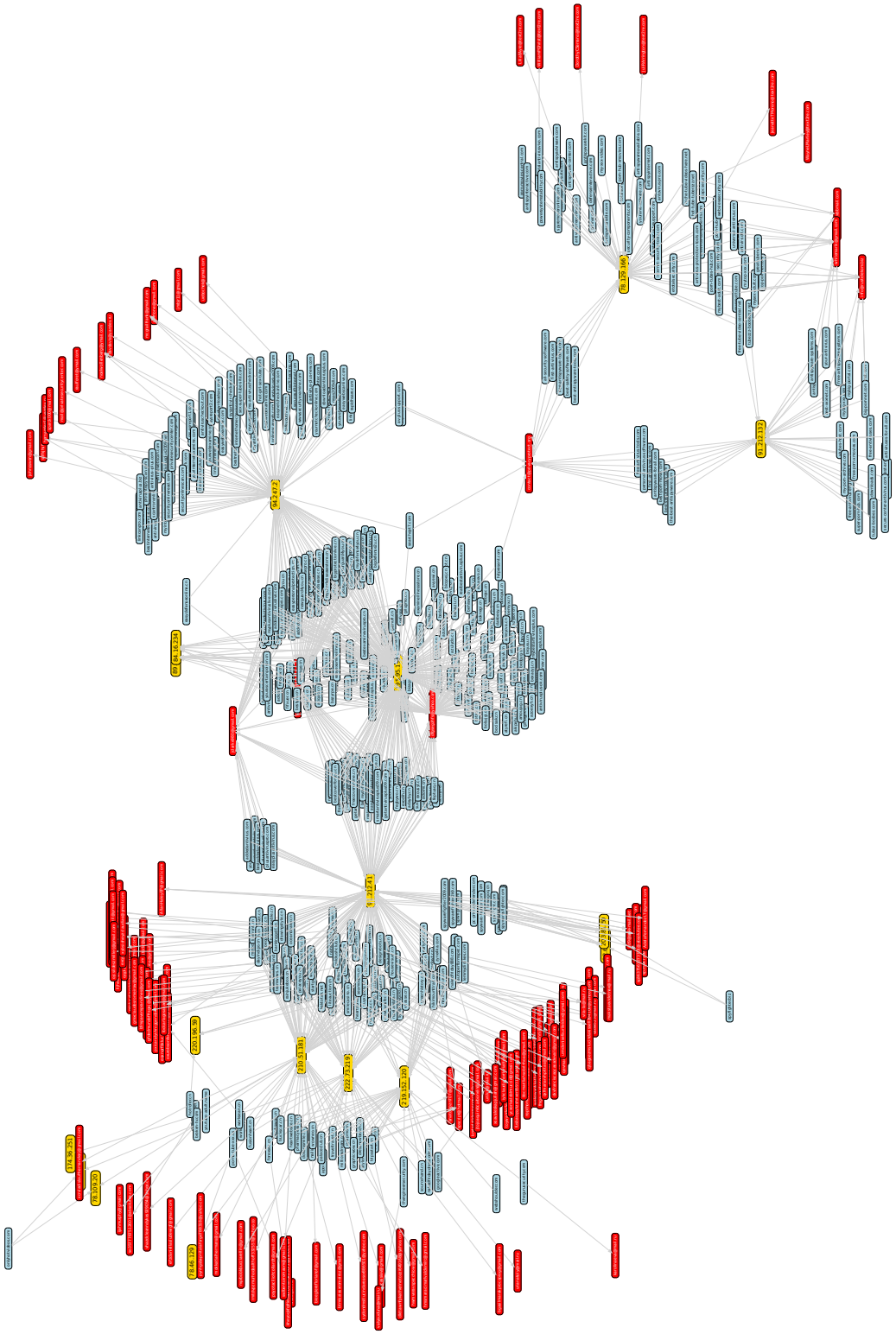


Figure 6.11: RC3: An affiliate-based rogue network.

Chapter 7

Conclusion and Perspectives

"The cause is hidden; the effect is visible to all."
– Ovid

The investigations performed in this research thesis have led to many interesting observations and new insights, which are summarized in this Chapter. We provide also answers to the research questions and to the general problem statement of the thesis. Finally, we conclude this work by presenting some interesting directions for future research.

7.1 Research contributions

This work started by underlining the importance of developing a systematic, analytical approach to address the problem of attack attribution. In the light of the developments and experimental results presented in the previous chapters, we are now able to answer the research questions formulated in the introduction of the thesis.

7.1.1 Answers to the research questions

Research question 1 (RQ1): *how can we analyze attack phenomena from separate viewpoints, revealing different correlation patterns?*

This problem was successfully addressed by a novel graph-based clustering approach that allows an analyst to extract hidden correlation patterns with respect to any attack feature. The particularity of investigating attack phenomena, and their root causes, is the fact that we usually have no "ground truth". This aspect has motivated the choice of unsupervised classification techniques, such as the one we have presented, which is based on the *dominant sets* framework.

We have also underlined the importance of selecting *salient features*, which can potentially reveal interesting viewpoints on unknown attack phenomena. Closely related to this feature selection process, we have showed how critical it is to define appropriate distance

metrics for comparing observations, such that this comparison truly reflects the *semantics* of the underlying phenomenon.

Conclusion 1: we have clearly demonstrated that our graph-based clustering approach could answer **RQ1**. In particular, this approach is appropriate to automate the discovery of new relevant knowledge from security data sets. Furthermore, the modularity of this iterative clustering approach facilitates the integration of new features in the framework, as a function of the analyst's requirements.

Research question 2 (RQ2): *how can we systematically combine all viewpoints such that the behavioral properties of attack phenomena are appropriately modeled in the aggregation process?*

We have presented a formal and elegant solution to this problem, by showing how an approach based on *multi-criteria decision analysis* (MCDA) can effectively help to combine multiple attack viewpoints. The reasons of this success are threefold.

First, *aggregation functions* used in MCDA are not bound to a rigid decision-making scheme, i.e., they are not based on binary relations between pairs of events. On the contrary, the inputs (e.g., cluster membership or similarity degree) can be naturally expressed by fuzzy variables. This can effectively address the problem of *fuzziness* and *uncertainty* that are intrinsically bound to real-world phenomena and imperfect measurements.

Secondly, attack features are, generally speaking, not independent. As a result, it is necessary to model *interactions* among subsets of inter-related features, such as a redundancy or a synergy between two or more features. Aggregation functions provide such a flexibility, and are particularly well-suited to this kind of modeling.

Thirdly, the nature of real-world phenomena is essentially *evolutive*. Consequently, it is difficult to predict which set of features is the most relevant to link an attack event to a given phenomenon, as it may have evolved with respect to previously observed events. Here too, we have demonstrated that aggregation functions allow analysts to get rid of the need to define the importance (or the relevance) of attack features in a rigid manner.

Conclusion 2: based on the observations here above, we can conclude that our approach based on multi-criteria decision analysis can successfully answer **RQ2**.

7.1.2 Answer to the problem statement

On the basis of the answers to the research questions, we are now in the position of answering the problem statement.

> *Can we effectively address the problem of attack attribution in the cyber-space by means of a systematic, analytical approach?*

As showed throughout this thesis, attack attribution in the cyberspace is a complex problem, mainly because of the large number of dimensions involved in the investigation process, and the dynamicity of real-world attack phenomena. However, we have demonstrated that this problem can be successfully addressed by means of an analytical method that leverages multiple attack viewpoints in a systematic (and possibly automated) way.

Through this multi-criteria analytical approach, the dynamic behavior of *a priori* unknown attack phenomena can be effectively modeled, and the *modus operandi* of the attackers are also better emphasized.

We believe that the results of our experimental validation on two different security data sets have clearly demonstrated the applicability and the effectiveness of this attack attribution method, but also the kind of new insights it can offer to security analysts with respect to open questions in the security domain. In conclusion, we believe that the proposed framework is a successful answer to the problem statement that has motivated this work. We are looking forward in having other opportunities to apply this method to other security datasets that future partners would be willing to share with us.

7.2 Future research

The development of this attack attribution framework has also opened a number of interesting questions and challenges that could be investigated in future research.

First, by acquiring more experience in threat monitoring, we can identify other innovative features that could represent some critical aspects of attack phenomena. The integration of these attack features into our framework can obviously offer additional viewpoints to security analysts, which could further improve our insights into emerging cyber-criminal activities.

Secondly, other clustering techniques could be considered in order to improve the scalability and the efficiency of the framework. We envision also the development of an *incremental*, DB-oriented version of this framework, such that larger data sets can be processed more easily, and can also be updated in a very short time. This would alleviate (at least partially) the scalability issue related to the exponential complexity of the memory requirements imposed by pairwise correlation techniques. At this point, we are indeed limited by the amount of memory that is needed to store the n dissimilarity matrices made of $m \times (m-1)/2$ elements (where n is the number of attack features, and m is the number of security events in the data set). As of now, the memory complexity of the framework is thus $\mathcal{O}(nm^2)$. Yet, in practice we could easily process data sets comprising up to 10,000 events, using 8 attack features concurrently. The processing time for such a data set is about a few hours.

Thirdly, we believe that the application of the very same method to many different (but still inter-related) security data sets, followed by the combination of all results, could provide a rather unique *multi-perspective viewpoint* on cyber-criminal activities. Examples of unexplored data sets that we may consider to include in such analysis are malware repositories, and rich attack data sets collected by high-interaction honeypots. Regarding malware data sets, there are many interesting features that we could leverage with our multi-criteria analysis method, such as behavioral features related to the dynamic execution of malware, or static features related to their coding style. An example of rich data set collected by high-interaction honeypots is the SGNET infrastructure, which enables to collect detailed information on server-based code injection attacks using a distributed and

highly scalable solution.

Fourth, we have extensively studied in this work two main classes of agregation functions, namely Ordered Weighted Averaging functions and Choquet integrals. However, many other families of aggregation functions exist and have been successfully used in other domains in which complex decision-making schemes have to be modeled. We hope this work will inspire other researchers to explore the utility of other multi-criteria methods for addressing the problem of *attack attribution*.

Fifth, we are looking forward to developing a software abstraction layer around our framework, which could hide the complexity and the implementation details of the multi-criteria clustering components. This would enable other security analysts to take advantage of our attack attribution method in a flexible way, so that they could potentially obtain new insights into their own data set.

Finally, we have showed along this thesis how different visualization techniques, such as graph-based visualizations and dimensionality reduction (t-SNE), could help us to achieve a better *situational awareness* in network and information security. We believe that this work has opened a very interesting and promising research avenue, and suggests also that more research should be carried on regarding the application of novel **visual analytics** technologies to security data sets, perhaps in combination with the multi-criteria analysis techniques presented in this thesis. This would enable us to perform a joint reasoning on security events and attack attribution in a systematic, but also visual and *interactive* way.

Improving all those different aspects should facilitate the development of an automated system that could be used as the basis for the construction of predictive threat models, and consequently, the development of an effective *Early Warning System*.

"The important thing is not to stop questioning.
Curiosity has its own reason for existing."
– Albert Einstein

Synthèse en français

Vers un regroupement multicritères comme outil d'aide à l'attribution d'attaque dans le cyber-espace

Olivier Thonnard, Mars 2010

1. Introduction

Une meilleure compréhension des menaces informatiques existantes ou émergeantes sur Internet devrait nous aider à mieux protéger notre infrastructure et l'économie qui en dépent. Même si cela peut sembler assez évident aux yeux de tout un chacun, atteindre un tel objectif est moins aisé qu'il n'y parait, et la plupart des experts en sécurité s'accordent tous au moins sur un fait: combattre la cybercriminalité devient aujourd'hui de plus en plus difficile [137, 33, 127]. Ceci semble être lié à la consolidation des organisations responsables des phénomènes d'attaque à grande échelle sur Internet [157, 158, 72, 94].

La motivation des cyber-criminels est bien entendu liée au profit financier considérable qu'ils peuvent réaliser grâce à la "monétisation" d'activités malveillantes, telles que la vente d'exploits (type *0-day*) et de malware, la location de machines compromises (machines dites *zombies*), le spam et le phishing, la vente d'informations personnelles volées, etc [54, 158]. Dans la plupart des cas, ces activités semblent être facilitées par la mise en oeuvre de réseaux de machines infectées, appelés *botnets*, qui peuvent compter des milliers, voire même des millions de machines zombies appartenant souvent à des utilisateurs résidentiels (ignorant d'ailleurs leur participation à ces réseaux malveillants) [11, 34, 126, 10]. Le problème mondial de l'envoi de spam sur Internet serait d'ailleurs aussi étroitement lié à ces botnets qui sont contrôlés par des organisations criminelles. Selon des rapports publiés récemment par SecureWorks [151] et MessageLabs [94], les principaux botnets observés en 2009 seraient capables d'envoyer quotidiennement plus de 107 milliards de spam. De plus, le taux moyen de messages de type spam interceptés par la société MessageLabs s'élevait à plus de 87% de l'ensemble des messages analysés.

Un autre phénomène, tout aussi inquiétant, est celui lié aux présumés "cyber-conflits" qui auraient eu lieu entre différents pays, et dont l'Estonie et la Géorgie ont été les victimes en 2007 et 2008 respectivement. Selon les analyses effectuées par différents experts [3, 37, 41], il semblerait que des botnets puissent être utilisés en tant que "armée digitale" pour attaquer les ressources informatiques d'un pays entier, simplement en lançant des attaques

de type déni de service distribué (DDoS) contre des systèmes critiques (tels que serveurs DNS, routeurs, sites web gouvernementaux et bancaires, etc). Vu les pertes économiques que cela peut engendrer, il est vital de mieux appréhender le comportement et l'évolution à long terme de ces nouvelles armées de machines [168].

Plus récemment, nous avons assisté à un autre phénomène de monétisation d'activité illégale par le biais de la distribution de logiciels anti-virus factices (appelés "rogue antivirus") [159, 112, 36]. En combinant des techniques d'ingénierie sociale avec des méthodes d'attaque plus sophistiquées (comme l'exploitation de failles dans les navigateurs web ou le piratage de sites web mal protégés), des cyber-fraudeurs sont capables de distribuer de faux logiciels anti-virus à des utilisateurs peu méfiants, ce qui leur offre un profit financier direct assez important étant donné le prix de vente moyen de ces faux logiciels[1].

1.1 Le problème de l'attribution d'attaque

Depuis 2003, il semblerait que l'on assiste donc à un changement assez radical dans la nature des menaces sur Internet. Tous les phénomènes décrits précédemment ont un problème en commun, à savoir celui de l'*attribution d'attaque*. Car même s'il existe des indices probables indiquant les origines, les causes et les conséquences de ces nouvelles activités malveillantes, assez peu d'affirmations peuvent être réellement soutenues ou démontrées par des preuves scientifiques. En particulier, pas mal de questions subsistent concernant l'attribution des attaques et l'organisation de la cybercriminalité: combien de communautés organisées sont responsables de ces phénomènes, quelles sont leurs origines (e.g., géographiques), quels fournisseurs d'accès Internet en particulier semblent offrir un hébergement protégé à ces activités (comme par exemple, RBN[2], Atrivo aussi connu sous le nom de Intercage, McColo, 3FN, etc)? Peut-être encore plus important, comment ces réseaux plutôt douteux offrant un "bullet-proof hosting" évoluent-ils dans le temps? Les botnets hébergés sur ces réseaux sont-ils capables de coordonner leurs efforts? Et finalement, parmi les milliers d'échantillons de malware reçus chaque jour par des sociétés telles que Symantec et VirusTotal, comment déterminer ceux qui pourraient provenir d'une même organisation criminelle?

1.2 Objectifs et contributions

La contribution principale de cette thèse consiste à développer une méthode analytique permettant d'aborder de manière systématique le problème de l'*attribution d'attaque* dans le cyber-espace, c'est à dire: comment attribuer des éléments d'attaque apparaissant peut-être comme différents à première vue, à une même cause ou une même origine, et ceci en combinant tous les indices disponibles. Par "une même cause", on entend par là l'identification de groupes d'événements, ou des communautés de machines responsables pour *différentes* attaques, mais qui sont vraisemblablement liés à un même phénomène. L'objectif est de faciliter l'identification du *type* de phénomène (e.g., ver, botnet, etc), mais un aspect important d'une telle méthode est avant tout sa capacité à mettre en évidence

[1] Des gains mensuels allant jusqu'à $332.000 ont été observés sur le site web de distribution `TrafficConverter.biz` utilisé par ces cyber-fraudeurs [77, 159]

[2] Le *Russian Business Network* (RBN) est une organisation cybercriminelle à multiples facettes, qui est surtout réputée pour l'hébergement d'activités commerciales fort douteuses, voire illégales telles que le phishing, le spam, la pornographie enfantine et la distribution de malware (voir `http://www.bizeul.org/files/RBN_study.pdf`).

les *modes opératoires* des attaquants. Ceci doit permettre une analyse stratégique à moyen et long terme des méthodes utilisées par les cyber-criminels dans la mise en oeuvre de leurs attaques. Notons que le but ultime de ce travail n'est pas de fournir des noms de personnes à des autorités ou des forces de police, mais bien d'avoir une méthode d'analyse systématique qui fournit des *modèles* comportementaux des entités ou des organisations responsables des attaques observées. Par la généralisation de ces modèles, cela nous permet de mieux comprendre les menaces réelles que chaque individu, ou chaque organisation connectée à Internet, doit apparemment affronter.

Enfin, la méthode d'attribution d'attaque recherchée doit aussi rester la plus générique possible. Les phénomènes d'attaque étant extrêmement variés, la même approche doit pouvoir être appliquée à des ensembles de données différents sans nécessiter de changement fondamental.

1.3 Enoncé du problème

Ayant décrit les problèmes auxquels les chercheurs en sécurité sont confrontés par rapport à l'attribution d'attaque sur Internet, cela nous permet à présent de définir la thèse que nous voulons démontrer dans ce travail:

> *Est-il possible d'aborder de manière systématique le problème de l'attribution d'attaque sur Internet par une méthode analytique?*

Ce problème peut se traduire par un processus de fouille de données dans des traces d'attaques provenant de l'observation de phénomènes à l'aide de divers senseurs. Par conséquent, nous avons abordé le problème en cherchant comment appliquer différentes techniques de regroupement et de classification à des *événements d'attaques*, eux-mêmes enrichis d'informations contextuelles. En regroupant de manière adéquate tous les indices disponibles ainsi que les caractéristiques observées, nous espérons pouvoir dériver des modèles riches en sémantique, mais également associer de nouveaux événements à des phénomènes observés précédemment. Les caractéristiques ou données contextuelles qui peuvent servir à un tel regroupement peuvent être, entre autres, les origines des attaques, leur succession dans le temps, la méthode de propagation, un biais dans leur implémentation (par exemple, au niveau codage), diverses caractéristiques comportementales, etc. Dans la suite de ce document, nous utiliserons le terme général de *caractéristique d'attaque* pour nous référer à ces différents aspects observables des phénomènes.

Notons que, comme hypothèse de départ, nous supposons que des ensembles de données *cohérents* et *représentatifs* soient disponibles pour l'analyse de phénomènes d'attaque dans le cyber-espace (e.g., données honeypot ou IDS, dépôts de malware, données collectées par des *web crawlers*). Par ailleurs, nous supposons aussi que le degré de coordination inhérent aux organisations cyber-criminelles devrait se traduire également par divers motifs de corrélation entre des groupes d'événements observés, ce qui justifie l'utilisation d'algorithmes de fouille de données et de découverte d'information.

A partir de l'énoncé général du problème et des hypothèses de travail ci-dessus, nous dérivons alors deux points plus spécifiques que nous tenterons de développer:

- **Problème de recherche 1 (PR1)**: *comment pouvons-nous analyser des phénomènes d'attaque à partir de différents points de vue, de sorte qu'ils nous fassent découvrir des motifs de corrélation intéressants?*

- **Problème de recherche 2 (PR2)**: *comment pouvons-nous combiner systématiquement différents points de vue d'attaque par un processus d'agrégation de données, de sorte que les propriétés comportementales des phénomènes observés soient modélisés de manière adéquate?*

Plus précisément, le **PR1** a trait au processus de découverte d'information dans des données de sécurité, dont le but est de créer des points de vue par rapport à différentes caractéristiques d'attaque, tandis que **PR2** concerne la *fusion* de tous ces points de vue par la combinaison appropriée des corrélations trouvées précédemment. Nous verrons que ce processus de fusion de données impliqué par **PR2** n'est pas trivial puisque deux sous-problèmes y sont intimement liés, à savoir: (1) l'aspect dynamique (ou *évolutif*) des phénomènes d'attaque; et (2) le problème d'*incertitude* inhérent à tout phénomène réel que l'on tente d'observer par différentes mesures.

1.4 Positionnement par rapport à l'état de l'art

Une étude approfondie de l'état de l'art a révélé les insuffisances des méthodes d'analyse actuellement utilisées dans le domaine *INFOSEC*[3] pour aborder le problème de l'attribution d'attaque. Tel que démontré dans la revue de la littérature au Chap. 2, beaucoup d'efforts ont été investis dans le développement d'infrastructures de collecte de données permettant le monitoring d'activités malveillantes sur Internet. En particulier, les techniques les plus utilisées sont:

- les honeypots/honeynets [110, 183, 5, 128, 84, 86, 83, 85, 120, 108, 73, 178, 26, 176],
- les *darknets*, tels que CAIDA [23], IMS [6], ShadowServer [163] et Team Cymru [164],
- les partages de logs IDS et pare-feu (D-Shield [45, 195], EmergingThreats [47]),
- les projets de collecte de malware (SGNET [83], Mwcollect Alliance [104], Offensive Computing [111] et Shadowserver Foundation [163] et VirusTotal [177]), ainsi que les *sandboxes* (ou "bacs à sable"), tels que Anubis [2], Norman Sandbox [109], Sunbelt CWSandbox [154], ou encore Argos [119],
- la détection et le tracking de botnets (BotHunter [65], BotMiner [64], [126, 10, 34, 79]).

Toutes ces méthodes de monitoring ou de détection sont aujourd'hui relativement matures d'un point de vue technique. Par contre, les méthodes pour analyser l'énorme quantité de données collectées restent relativement peu développées. Par exemple, la plupart des méthodes d'analyse de trafic malveillant sur Internet (e.g., honeynet, darknet) s'appuient principalement sur des techniques statistiques assez simples, telles que le nombre d'attaques groupées par port, par jour/heure, la répartition des attaques par pays ou par réseau d'origine (AS ou ISP), etc.

Une autre lacune importante des méthodes d'analyse traditionnelles est le manque de flexibilité pour inclure des informations *contextuelles*, voire de l'intelligence collective, à propos des attaques observées. Ceci nous permettrait de mieux appréhender certains

[3] INFOSEC: sécurité de l'information (En: information security).

phénomènes en découvrant des liens ou des rapprochements non évidents entre des événements d'attaque qui peuvent sembler à priori différents. Un des objectifs de ce travail est donc de combler cette lacune en développant une méthode permettant de combiner systématiquement un grand nombre de points de vue différents, grâce à des techniques de fouille de données et d'analyse décisionnelle multicritères.

Notre approche s'inscrit aussi dans une perspective visant à améliorer l'aspect *Situational awareness* dans le cyber-espace [196], c'est à dire, permettre à un analyste sécurité d'évaluer rapidement un phénomène d'attaque à l'aide d'informations de plus haut niveau, telles que:

- la cause d'une attaque (e.g., un nouveau ver, un botnet, ou un problème de configuration), même si les événements observés n'ont pas de signature connue,
- est-ce que l'attaque était ciblée ou non,
- est-ce que les nouveaux événements peuvent être attribués à des phénomènes déjà observés dans le passé.

Bien que certains progrès ont été effectués dans cette direction [145], l'état de l'art nous montre que ce domaine de recherche est encore fort peu exploré.

Ensuite, nos recherches s'inscrivent aussi dans un domaine appelé "fouille de données investigatrice", dans lequel des techniques de *data mining* ont été spécifiquement adaptées aux besoins liés à des tâches d'investigation (éventuellement criminelle). Concernant l'aspect *INFOSEC*, tous les efforts de ces dernières années se sont concentrés sur l'amélioration des systèmes IDS[4] (i.e., améliorer leur base de signatures d'attaque) par l'usage de techniques d'apprentissage automatique (e.g., règles d'association, règles d'épisode fréquent, algorithmes de classification) [9, 80, 81, 48, 44, 75, 8, 21]. Notre recherche est fort différente à plusieurs niveaux, puisque notre objectif ne consiste pas à générer de nouvelles signatures d'attaque pour des systèmes IDS, mais plutôt de comprendre les causes fondamentales des phénomènes observés ainsi que leur comportement. De plus, les méthodes que nous développons s'appliquent principalement à des données déjà identifées comme étant de nature malicieuse, et donc le but est d'en tirer parti pour mieux comprendre les *modes opératoires* des attaquants.

La fouille de données investigatrice est aussi appliquée dans le domaine de l'*intelligence*, ou celui des enquêtes criminelles [92, 93, 184]. Les efforts effectués dans ce domaine concernent principalement l'application de techniques de fouille assez traditionnelles ("off te shelf"), telles que l'analyse de relations entre événements (*link analysis*), la fouille de texte, les réseaux de neurones, et l'analyse de réseaux sociaux à l'aide de graphes (*Social Network Analysis*, ou SNA). Bien que notre approche présente des similarités avec ces techniques, il y a aussi des différences notables: (i) notre regroupement basé sur les graphes peut s'appuyer sur des métriques de distance plus élaborées que celles utilisées en général en *link analysis* ou SNA; et (ii) notre approche permet de combiner plusieurs graphes de similarité à l'aide de techniques d'analyse décisionnelle multicritères, permettant ainsi de modéliser des relations plus complexes. Ce genre de traitement n'est pas encore disponible dans les techniques SNA traditionnelles.

[4]IDS: système de détection d'intrusions (Intrusion Detection System).

Finalement, nous avons formalisé la dernière étape du processus d'attribution d'attaque comme un problème d'analyse décisionnelle multicritères (MCDA). Notons au passage que les techniques MCDA ont été retenues par certains spécialistes parmi les méthodes d'analyse les plus utiles et les plus efficaces dans le domaine *intelligence analysis* [185].

Dans une approche MCDA, il est nécessaire de définir une *fonction d'agrégation* qui modélise le comportement des phénomènes (i.e., *domain knowledge*), afin d'attribuer de nouvelles observations à des phénomènes de manière appropriée. Il existe plusieurs méthodes bien connues dans le domaine des applications MCDA (également appelé MAUT, pour *Multi-Attribute Utility Theory*) qui réalisent cette fonction d'agrégation multicritères de différentes façons. Dans la plupart de cas, la fonction d'agrégation est basée sur une simple fonction de moyenne arithmétique (éventuellement pondérée), tel que Simple Additive Weighting, Weighted Product Method, ou la méthode AHP (Analytical Hierarchy Process) [197]. Les méthodes de recherche opérationnelle telles que ELECTRE, TOPSIS et PROMETHEE [51] sont également trois autres méthodes relationnelles bien connues dans l'aide à la décision, qui permettent de classer des alternatives en les évaluant de façon pair-à-pair par rapport à plusieurs critères d'intérêt.

Dans notre approche, les fonctions d'agrégation utilisées pour combiner les corrélations sont plus élaborées. En effet, nous démontrons qu'il est possible de modéliser le processus d'attribution d'attaque à l'aide de moyennes ordonnées (Ordered Weighted Averaging [192]), et dans des cas plus complexes, à l'aide de l'intégrale de Choquet [63, 62, 17, 173] appliquée sur une mesure floue qui modélise les interactions entre les différentes caractéristiques d'attaque. A notre connaissance, une telle méthode d'attribution n'existe pas encore dans le domaine lié à l'analyse des menaces sur Internet.

1.5 Structure du document

Le reste de ce document est structuré de la manière suivante. La Section 2 introduit la méthode d'attribution d'attaque telle que développée dans cette thèse. Dans cette même Section, nous tentons de répondre aux deux questions de recherches **PR1** et **PR2** définies précédemment. Ensuite, la Section 3 présente la validation expérimentale de la méthode effectuée sur deux ensembles de données différents: (i) l'application aux traces d'attaque réseau collectées par un honeynet distribué mondialement pendant plus de deux ans, et (ii) l'application de la méthode à un ensemble de données collectées par différentes sources sur Internet (entre autres, des crawlers web et des honeypots client), qui caractérisent des sites web hébergeant ou offrant des logiciels antivirus factices (*rogue antivirus*). La Section 4 conclut le document et suggère des perspectives intéressantes pouvant faire l'objet de recherche future.

2. Méthode d'attribution d'attaque

En fouille de données investigatrice, l'analyste doit en général synthétiser différents éléments ou indices permettant d'identifier les causes de phénomènes. L'objectif final est de déterminer comment "connecter les points", i.e., comment découvrir des motifs de corrélation entre différentes observations, et comment les combiner ensuite de manière pertinente, de sorte qu'on obtienne finalement une vue d'ensemble (the "big picture") sur les phénomènes recherchés [184]. Toutefois, les quantités de données collectées par les systèmes

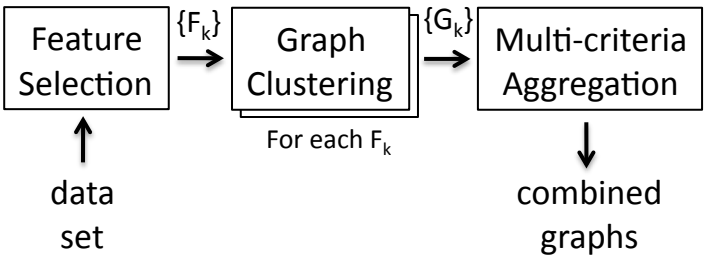


Figure 1.1: Aperçu de la méthode de regroupement multicritères. F_k indique l'ensemble des caractéristiques sélectionnées à partir des données de sécurité. G_k représente l'ensemble de graphes de relations obtenus par l'algorithme de regroupement. Quant au composant multicritères, il se charge de combiner tous les graphes G_k pour produire un graphe combinant toutes les caractéristiques d'attaque.

actuels excèdent de loin notre capacité à les analyser manuellement. C'est pourquoi il est nécessaire de développer une méthode de regroupement systématique capable d'extraire et de combiner des motifs de corrélation à priori inconnus, en tenant compte de multiples caractéristiques potentiellement intéressantes.

Tel qu'illustré à la Fig. 1.1, notre approche s'appuie sur trois composants:

1. **Sélection de caractéristique**: nous déterminons quelles caractéristiques (désignées par F_k) nous désirons inclure dans l'analyse globale, et nous générons pour chacune d'entre elles les vecteurs caractéristiques correspondant (*feature vectors*) pour l'ensemble des objets ou événements constituant l'ensemble de données;

2. **Regroupement par graphes**: un graphe de similarité est créé pour chaque caractéristique d'attaque F_k, en utilisant une métrique de distance appropriée à chaque vecteur caractéristique. Les sous-groupes fortement connectés au sein de chaque graphe peuvent alors être identifiés, afin de révéler les corrélations fortes existant parmi des groupes d'objets;

3. **Agrégation multicritères**: les différents graphes de similarité sont alors combinés en utilisant une fonction d'agrégation qui modélise le comportement attendu des phénomènes sous-jacents que l'on veut identifier.

Notons que l'approche est principalement non supervisée, i.e., elle ne requiert aucune phase d'apprentissage afin de relier les objets (ou les événements) aux phénomènes sous-jacents qui en sont probablement la cause.

2.1 Sélection de caractéristiques d'attaque

Une analyse par regroupement (*cluster analysis*) commence la plupart du temps par une étape qui consiste à sélectionner des caractéristiques pertinentes, i.e., pouvant révéler des *corrélations* intéressantes et instructives [70]. Cette sélection de caractéristiques peut éventuellement être complétée par une ou plusieurs transformations des données initiales,

afin de produire des données dont le format est mieux adapté au traitement ultérieur (par exemple, un processus de normalisation). Enfin, ces caractéristiques doivent aussi être représentées sous une forme adéquate en définissant un certain nombre de classes, de catégories, ou de variables, lesquelles seront traitées par l'algorithme de regroupement.

De manière plus formelle, nous avons donc un ensemble de données $\mathcal{D}$ composé de m "objects", qui sont en général, dans notre domaine d'intérêt, des *événements* de sécurité (e.g., l'observation d'une ou plusieurs attaques par des honeypots, des alertes d'intrusion données par un IDS, etc). Nous définissons ensuite un ensemble de n caractéristiques $F = \{F_k\}, k = 1, \ldots, n$. Le but de cette première étape est de créer des vecteurs caractéristiques pour chaque événement e_i contenu dans $\mathcal{D}$. Nous utilisons la notation $\mathbf{x}_i^{(k)}$ pour représenter le vecteur extrait pour l'événement e_i par rapport à la caractéristique F_k. En fait, $\mathbf{x}_i^{(k)} \in \mathbb{R}^d$ est un vecteur de valeurs réelles composé de d dimensions, i.e.:

$$\mathbf{x}_i^{(k)} = \{x_{i,1}^{(k)}, \ldots, x_{i,d}^{(k)}\}$$

où d est fonction de la caractéristique F_k.

Finalement, nous pouvons grouper tous les vecteurs définis par rapport à un caractéristique donnée dans un ensemble $\mathbf{X}_k = \{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_m^{(k)}\}$. En data mining, il est assez courant d'utiliser une notation sous forme de matrice pour représenter cet ensemble de vecteurs caractéristiques $\mathbf{X}_k$, i.e.:

$$\mathbf{X}_k = \begin{bmatrix} x_{1,1}^{(k)} & x_{1,2}^{(k)} & \cdots & x_{1,d}^{(k)} \\ x_{2,1}^{(k)} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{m,1}^{(k)} & \cdots & \cdots & x_{m,d}^{(k)} \end{bmatrix}$$

où la i^{eme} ligne représente le vecteur caractéristique $\mathbf{x}_i^{(k)}$ extrait pour l'événement $e_i \in \mathcal{D}$, et obtenu pour la k^{eme} caractéristique F_k.

En résumé, la dimensionnalité de notre problème est composée comme suit: m est le nombre d'événements de sécurité, n est le nombre de caractéristiques d'attaque, et d est la dimension du vecteur (cette dernière étant fonction de chaque F_k).

Un exemple concret de vecteur caractéristique pourrait être la distribution géographique des attaquants pour un événement composé de plusieurs traces d'attaque observées par un honeypot durant un intervalle de temps déterminé. Dans ce cas, le vecteur serait, par exemple, composé de 229 variables représentant chacune un pays d'origine, et la valeur de chaque variable serait le nombre d'attaquants provenant de ce pays (i.e., la fréquence absolue). Une autre manière de représenter cette information serait d'utiliser des fréquences relatives, comme par exemple: US(35%),CN(7%),DE(5%),CA(5%), autres pays (47%).

2.2 Regroupement par graphes et découverte d'information

La méthode de regroupement que nous avons choisi dans le 2ème composant s'appuie sur un technique de regroupement pair-à-pair. Nous avons formulé le problème en utilisant un approche basée sur les graphes qui est inspirée par l'approche développée par Pouget dans [120]. Nous avons ensuite étendu et amélioré cette approche, en intégrant entre autres des métriques plus élaborées (e.g., distances statistiques).

Pour chaque caractéristique d'attaque F_k, nous construisons un graphe non dirigé, pondéré, et sans boucle, que l'on note G_k, dans lequel les noeuds correspondent aux vecteurs $\mathbf{x}_i^{(k)}$, et les poids des arêtes (ou liens) reflètent le degré de similarité entre les objets ou événements par rapport à la caractéristique F_k considérée. Nous représentons un tel graphe par les notations suivantes:

$$G_k = (V_k, E_k, \omega_k)$$

$$\text{où } \begin{cases} V_k = \{\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}, \ldots, \mathbf{x}_m^{(k)}\} & \text{est l'ensemble des noeuds} \\ E_k \subseteq V_k \times V_k & \text{est l'ensemble d'arêtes (i.e., les relations entre noeuds)} \\ \omega_k : E_k \rightarrow \Re^+ & \text{est une fonction positive représentant les poids} \end{cases}$$

En pratique, nous représentons chaque graphe G_k à l'aide de sa *matrice d'adjacence* (aussi appelée matrice de dissimilarité), qui est une matrice $m \times m$ symétrique définie par:

$$A_k(i,j) = \begin{cases} \omega_k(i,j), & \forall (i,j) \in E_k \\ 0, & \text{autrement.} \end{cases}$$

Métrique de similarité

Il est évident que la fonction de poids $\omega_k(i,j)$ doit être définie à l'aide d'une métrique de distance appropriée à la forme des vecteurs $\mathbf{x}_i^{(k)}$. Ceci est valable en fait pour n'importe quel algorithme de regroupement, en particulier ceux qui s'appuient sur une approche pair-à-pair (par ex., le clustering hiérarchique est confronté au même problème). Le choix de cette métrique est fondamental, puisqu'elle a un impact direct sur les propriétés des *clusters* obtenus, telles que leur taille et leurs homogénéités interne et externe.

La distance la plus communément utilisée est probablement la distance euclidienne, qui n'est en fait rien d'autre qu'un cas particulier de la métrique de Minkowski (pour $p = 2$):

$$\begin{aligned} d_p(\mathbf{x}_i, \mathbf{x}_j) &= \left(\sum_{k=1}^{d} |x_{i,k} - x_{k,j}|^p\right)^{\frac{1}{p}} \\ &= \|\mathbf{x}_i - \mathbf{x}_j\|_p \end{aligned}$$

Les distances euclidiennes souffrent non seulement du problème d'échelle (*scaling*) entre les vecteurs à comparer, mais surtout ces distances sont complètement inappropriées quand on traite des données n-dimensionnelles (avec n assez élevé). Ceci est principalement dû au problème surnommé la "malédiction de la dimensionnalité"[5], et qui est causé par une augmentation exponentielle en volume quand le nombre de dimensions augmente. Par conséquent, les concepts de "proximité", "distance", ou de voisin le plus proche, n'ont plus du tout la même signification quand on utilise des distance orthonormées (du type normes L_k) dans ce genre d'espace, ce qui peut fausser les résultats des algorithmes de data mining [1]).

Une autre métrique fréquemment utilisée pour comparer des séquences de valeurs réelles est le coefficient de corrélation, ou coefficient de *Pearson*, défini par:

$$d_{corr}(\mathbf{x}_i, \mathbf{x}_j) = \frac{(\mathbf{x}_i - \overline{\mathbf{x}})^T (\mathbf{x}_j - \overline{\mathbf{x}})}{\sqrt{(\mathbf{x}_i - \overline{\mathbf{x}})^T (\mathbf{x}_i - \overline{\mathbf{x}})} \sqrt{(\mathbf{x}_j - \overline{\mathbf{x}})^T (\mathbf{x}_j - \overline{\mathbf{x}})}} \tag{1.1}$$

[5] *The curse of dimensionality*, un terme introduit pour la première fois par Richard Bellman en 1957 [18].

où $\overline{\mathbf{x}}$ represents $(\mathbf{x}_i + \mathbf{x}_j)/2$.

Le coefficient de corrélation reflète le degré de dépendance linaire entre deux vecteurs (i.e., le degré de similarité entre leurs "formes").

Finalement, quand on doit comparer des vecteurs représentant des distributions empiriques (i.e., des histogrammes de fréquences), des distances (ou éventuellement divergences) statistiques, telles que Kullback-Leibler, sont plus appropriées. Par exemple, si $\mathbf{x}_i$ et $\mathbf{x}_j$ sont deux distributions de probabilité, alors la divergence de Kullback-Leibler de $\mathbf{x}_j$ vers $\mathbf{x}_i$ est définie comme étant:

$$D_{KL}(\mathbf{x}_i||\mathbf{x}_j) = \sum_{k=1}^{d} \mathbf{x}_i(k) \log \frac{\mathbf{x}_i(k)}{\mathbf{x}_j(k)}$$

qui est aussi appelée la divergence d'information (ou entropie relative). Puisque D_{KL} n'est pas considérée comme une vraie métrique (car pas symétrique), il est en général conseillé d'utiliser la distance de Jensen-Shannon ([87]), définie par:

$$D_{JS}(\mathbf{x}_i, \mathbf{x}_j) = \frac{D_{KL}(\mathbf{x}_i||\bar{\mathbf{x}}) + D_{KL}(\mathbf{x}_j||\bar{\mathbf{x}})}{2} \tag{1.2}$$

avec $\bar{\mathbf{x}} = (\mathbf{x}_i + \mathbf{x}_j)/2$.

Une alternative à Jensen-Shannon pour mesurer la similarité entre deux distributions est la distance de *Bhattacharyya* ([19]), qui est surtout utilisée en traitement de signal.

Regroupement par graphes

En théorie des graphes, la plupart des algorithmes de regroupement (*clustering*) consistent à chercher des structures combinatoires dans un graphe de similarité. Quelques exemples bien connus sont l'algorithme de l'arbre recouvrant minimum (*minimum spanning tree* [199]), ou celui de la coupe minimale (*minimum cut* [144, 190]). Une autre approche classique est celle qui consiste à rechercher des sous-graphes complets, telle que l'algorithme de "couplage complet" (*complete linkage*).Un sous-graphe complet maximal, également appelé une clique maximale, est en effet considéré comme la définition la plus stricte d'un cluster dans [4] et [125].

Le concept de clique maximale était à l'origine défini uniquement sur des graphes non pondérés, mais cela a été généralisé récemment au cas des graphes pondérés par Pavan et al. [116] qui ont proposé une nouvelle approche de clustering basée sur les ensembles dominants (*dominant sets*). Leur algorithme consiste à extraire itérativement des ensembles dominants dans un graphe pondéré, en enlevant à chaque étape les noeuds appartenant à cet ensemble jusqu'à ce que tous les noeuds soient regroupés (partitionnement complet), ou dès qu'un critère d'arrêt soit satisfait, ce qui peut éventuellement mener à une partition incomplète. Quelques exemples de contraintes qu'on peut utiliser comme critères d'arrêt sont: (i) un seuil minimum pour le nombre de noeuds ou d'arêtes restant dans le graphe; (ii) un seuil inférieur sur la somme des poids des arêtes restantes (par exemple, la procédure s'arrête si cette somme est inférieure à 0.01 de la quantité initiale). L'algorithme de clustering des ensembles dominants est décrit par le pseudo-code donné par l'algorithme 1.1. Tel qu'on peut voir, le fondement de cet algorithme est la procédure DOMINANT_SET, qui doit encore être définie.

Dans [116], il a été démontré qu'il existe une correspondance directe entre le problème d'identification d'ensembles dominants et le problème de trouver des extrema d'une fonction quadratique continue dans un simplex standard. Ceci signifie que l'on peut trouver des ensembles dominants (i.e., des clusters) en utilisant des techniques d'optimisation continues telles que des équations de réplication, qui sont des systèmes dynamiques utilisés en théorie évolutive du jeu. De tels systèmes sont également attrayants car ils peuvent être codés en quelques lignes seulement dans n'importe quel langage de programmation de haut niveau.

En conséquence, nous pouvons trouver des ensembles dominants en faisant simplement converger une expression temporelle particulière, exprimée par le système dynamique suivant:

$$x_i(t+1) = x_i(t) \cdot \frac{(A_k \mathbf{x}(t))_i}{\mathbf{x}(t)^T A_k \mathbf{x}(t)}$$

avec A_k la matrice d'adjacence du graphe G_k, et $i = 1, \ldots, N$. Partant d'un état initial arbitraire, ce système dynamique sera attiré par le point asymptotiquement stable le plus proche, ce qui correspondra à un ensemble dominant, et donc à une clique de poids maximal. Ensuite, l'algorithme enlève du graphe les noeuds de l'ensemble dominant trouvé, et recommence avec les noeuds restants.

Algorithm 1.1 Dominant sets Clustering

Input: un graph de similarité $G = (V, E, \omega)$
Output: une partition $\mathcal{P}$ (éventuellement incomplète)

```
P = ∅
while STOPPING_CRITERION(G) do
    S ← DOMINANT_SET(G)
    P ← P ∪ {S}
    V ← V \ S
return P
```

2.3 Attribution d'attaque par l'analyse décisionnelle multicritères

Le processus de découverte d'information basé sur le regroupement par graphes nous permet de créer différents points de vue par rapport à des caractéristiques d'attaque pertinentes. L'étape suivante consiste à combiner tous ces points de vue de manière intelligente, c'est à dire en utilisant une méthode d'agrégation qui puisse modéliser les comportements des phénomènes d'attaque sous-jacents.

Les *fonctions d'agrégation* sont utilisées dans diverses situations où nous devons évaluer différentes options par rapport à des critères d'intérêt. L'objectif consiste à calculer un score global combiné pour chaque option, qui peut alors servir de base au processus de prise de décision. Les fonctions d'agrégation sont, par exemple, largement utilisées en analyse décisionnelle multicritères (MCDA), où des alternatives doivent être évaluées sur base de critères qui sont même parfois conflictuels. Ces critères sont exprimés en général à l'aide de valeurs numériques exprimant un degré de préférence, ou un degré d'appartenance.

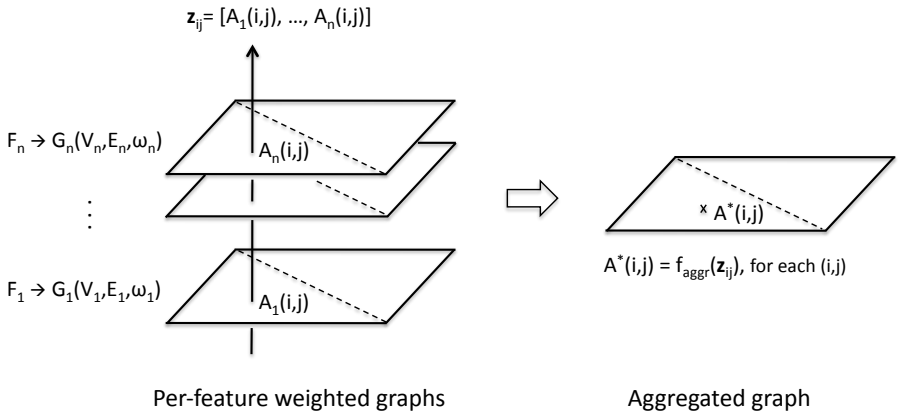


Figure 1.2: Illustration du processus d'agrégation effectué sur n graphes de similarité.

Definition 1.1. (Fonction d'agrégation [17]) *Une fonction d'agrégation est une fonction à n arguments ($n > 1$) qui projette le cube unitaire n-dimensionnel sur l'intervalle unitaire: $f_{aggr} : [0,1]^n \longrightarrow [0,1]$, en respectant les propriétés suivantes:*

(i) $f_{aggr}(\underbrace{0,0,\ldots,0}_{n\text{-times}}) = 0 \quad and \quad f_{aggr}(\underbrace{1,1,\ldots,1}_{n\text{-times}}) = 1$

(ii) $x_i \leq y_i$ *for all* $i \in \{1,\ldots,n\}$ *implies* $f_{aggr}(x_1,\ldots,x_n) \leq f_{aggr}(y_1,\ldots,y_n)$

Tous les intervalles unitaires $[0,1]$ *sont considérés ici comme étant continus.*

Dans notre méthode d'attribution multicritères, nous avons n caractéristiques d'attaques F_k, dont les indices font partie de l'ensemble $\mathcal{N} = \{1,2,\ldots,n\}$. Pour chaque F_k, nous avons construit un graphe $G_k = (V_k, E_k, \omega_k)$, représenté par sa matrice de similarité $A_k(i,j) = \omega_k(i,j)$, avec ω_k défini selon une métrique de distance appropriée.

Donc, pour chaque paire d'événements (i,j) provenant des données $\mathcal{D}$, un *vecteur de critères* $\mathbf{z}_{ij} \in [0,1]^n$ peut être construit à partir des matrices de similarités, de sorte que:

$$\mathbf{z}_{ij} = [A_1(i,j), A_2(i,j), \ldots, A_n(i,j)]$$

De manière informelle, notre approche consiste donc à combiner les n valeurs de chaque vecteur $\mathbf{z}_{ij}$ qui reflète chacun l'ensemble des relations existant entre une paire d'événements. Le résultat de cette opération (illustrée à la Fig. 1.2) est alors un graphe *combiné* $G^* = \sum G_k$.

En fait, une approche relativement simpliste pourrait se réduire à effectuer une simple moyenne arithmétique, éventuellement pondérée par des coefficients d'importance (i.e., une moyenne pondérée). Toutefois, ce genre d'agrégation ne nous permet pas de modéliser des relations plus complexes, comme par exemple: "la plupart des critères" ou bien "au moins x" critères doivent être satisfaits dans le schéma d'attribution. De plus, l'analyste

ne peut pas toujours exprimer à l'avance quel groupe de critères sera relevant pour chaque paire d'événements. Nous avons donc besoin d'une fonction d'agrégation dans laquelle les combinaisons de critères (et les coefficients associés à ceux-ci) ne sont pas prédéfinies de manière statique.

Ordered Weighted Averaging

Dans [192], Yager a introduit un nouvel opérateur appelé *Ordered Weighted Averaging* (OWA). Cet opérateur permet d'inclure certaines relations entre critères grâce au fait que les coefficients de pondération ne sont pas attribués de manière fixe aux critères évalués, mais plutôt sur leurs scores préalablement ordonnées. Ceci permet d'exprimer par exemple que "la plupart des critères", ou bien, "au moins deux critères" doivent être satisfaits afin de décider de relier ensemble deux événements. L'opérateur OWA peut donc influencer le résultat final en mettant l'accent soit sur les x valeurs les plus grandes (ou plus petites), ou encore sur les valeurs centrales.

Definition 1.2. (OWA) [192, 17] *Etant donné un vecteur de pondération* $\mathbf{w}$*,* $w_i \geq 0$*,* $\sum w_i = 1$*, la fonction d'agrégation* OWA *est définie par:*

$$OWA_{\mathbf{w}}(\mathbf{z}) = \sum_{i=1}^{n} w_i z_{(i)} =< \mathbf{w}, \mathbf{z}_{\searrow} > \tag{1.3}$$

où nous utilisons la notation $\mathbf{z}_{\searrow}$ *pour représenter le vecteur obtenu à partir de* $\mathbf{z}$ *en ordonnant ses composants en ordre décroissant:* $z_{(1)} \geq z_{(2)} \geq \ldots \geq z_{(n)}$*.*

Il est facile de voir que, pour n'importe quel vecteur $\mathbf{w}$, le résultat de l'OWA est situé entre les opérateurs classiques **and** et **or**, qui sont en fait les deux cas extrêmes quand $\mathbf{w} = (0, 0, \ldots, 1)$ (OWA est alors la fonction MIN) et quand $\mathbf{z} = (1, 0, \ldots, 0)$ (fonction MAX). Un autre cas particulier est celui pour lequel tous les coefficients $w_i = \frac{1}{n}$, ce qui revient à effectuer une moyenne arithmétique classique.

Nous observons que l'opérateur OWA a donné naissance à toute une série de variantes en tant que fonctions d'agrégation, telles que Maximum Entropy OWA (MEOWA), Minimum Variance OWA (MVOWA), ou encore Neat OWA, Generalized OWA, etc [17].

Une variante qui nous semble intéressante dans le cadre de l'attribution d'attaque est la fonction *Weighted OWA* (WOWA), introduite par Torra [171], et qui combine les avantages de la moyenne pondérée avec ceux de l'opérateur OWA. Cela permet à l'analyste de définir non seulement des coefficients de pondération sur les valeurs ordonnées des critères de similarité (comme pour OWA), mais également de quantifier la *fiabilité* des sources d'information (i.e., les caractéristiques d'attaque) en définissant un second vecteur de coefficients $\mathbf{p}$ qui attribue un poids à chaque critère évalué, indépendamment de sa valeur (comme dans une moyenne pondérée).

Dans le texte en anglais, nous avons effectué une étude de cas sur base de cet opérateur Weighted OWA, et nous avons comparé les résultats avec ceux obtenus avec une simple moyenne pondérée ainsi qu'avec l'opérateur OWA. Les résultats démontrent que cet opéra-

teur offre une meilleure flexibilité dans la modélisation des préférences d'agrégation, et donc aussi dans le processus d'attribution d'attaque.

Graphe combiné G^*

Ayant défini l'opérateur OWA, il est à présent aisé de l'appliquer aux vecteurs de critères $\mathbf{z}_{ij}$, i.e.:

$$A^*(i,j) = OWA_{\mathbf{w}}(\mathbf{z}_{ij}),\ \forall (i,j) \in \mathcal{D} \tag{1.4}$$

avec comme résultat, A^* la matrice de similarité du graphe combiné G^*.

Nous avons réalisé l'opérateur OWA à l'aide d'une fonction MATLAB® en exploitant l'approche vectorisée de l'environnement. Par conséquent, si nous avons m événements dans les données de départ et n caractéristiques à évaluer, cette approche vectorisée permet d'appliquer l'opérateur aux m^2 éléments des n matrices de similarité en une seule opération. La seule limitation est bien entendu la quantité de mémoire qui est nécessaire sur un seul ordinateur pour stocker les n matrices. Ceci dit, nous avons pu réalisé des expérimentations sur des données contenant jusqu'à 10.000 événements, en combinant 8 caractéristiques d'attaque. Notons également que ces opérations sont facilement parallélisables, si toutefois on devait appliquer cette approche sur des ensembles de données de taille nettement supérieure.

Evidemment, le problème de la définition des coefficients w_i à utiliser subsiste. Yager suggère deux approches possibles: (i) soit utiliser un mécanisme d'apprentissage, à l'aide de données d'apprentissage et d'un modèle de régression (e.g., trouver les coefficients qui s'adaptent aux données en minimisant l'erreur résiduelle selon les moindres carrés), ou (ii) donner une sémantique, c.à.d. une signification aux différents coefficients w_i en demandant à un expert du domaine (ou au décideur) de fournir directement ces valeurs, sur base de la connaissance du domaine. Dans la plupart des cas en attribution d'attaque, il est fort probable que l'on doive choisir la 2ème solution, puisque le processus global est de nature non-supervisée. Il est en fait difficile de générer ou d'obtenir des données d'apprentissage représentatives des phénomènes inconnus à identifier. Toutefois, nous renvoyons le lecteur intéressé au texte anglais de cette dissertation dans lequel nous discutons quelques méthodes qui peuvent aider à déterminer ces coefficients de manière plus rigoureuse et justifiée.

Finalement, à partir du graphe combiné G^*, nous pouvons alors aisément extraire les sous-graphes connectés, c'est à dire les ensembles d'événements qui sont interconnectés entre eux dans ce graphe:

$$\begin{aligned} \mathcal{P} &= components(A^*) \\ &= \{SG_1, SG_2, \ldots, SG_m\} \end{aligned}$$

Ceci nous donne enfin un ensemble de sous-graphes $\mathcal{P}$, où $SG_x \subseteq G^*$, et $\forall (i,j) \in SG_x : OWA_w(\mathbf{z}_{ij}) \geq t$, avec $t \in]0,1]$. En analysant et en visualisant chaque sous-graphe, l'analyste obtient à présent une bien meilleure vue des phénomènes sous-jacents ayant causé les observations. Tel qu'illustré par nos résultats expérimentaux, nous obtenons une image globale de toutes les relations importantes qui relient les groupes d'événements attribués à un même phénomène, ce qui facilite l'identification de la cause fondamentale et donne aussi une meilleure compréhension du son comportement.

Notons pour finir qu'il est utile d'appliquer une fonction de seuil sur la matrice A^* avant

d'identifier les sous-graphes connectés. Cela élimine ainsi les liens faibles entre événements non corrélés, mais qui proviennent de l'agrégation d'un certain nombre de corrélations fortuites non pertinentes. Nous invitons le lecteur intéressé à parcourir le texte en anglais dans lequel il est expliqué comment déterminer cette valeur de seuil à l'aide d'une étude de sensibilité.

Intégrale de Choquet

Dans le cas où les phénomènes étudiés semblent présenter des comportements plus complexes, il peut être nécessaire d'utiliser des fonctions d'agrégation autorisant une modélisation plus souple et plus flexible. Cela peut se présenter dans le cas où l'analyste doit modéliser des *interactions* entre critères qui ne sont donc pas complètement indépendants les un des autres. Par exemple, certaines caractéristiques d'attaque peuvent présenter une *synergie* (i.e., une interaction positive, ou complémentarité) par laquelle leur effet combiné est plus important que la somme de leurs effets individuels; ou au contraire, certains critères peuvent aussi montrer une certaine *redondance* (i.e., une interaction négative, ou substituabilité) par laquelle l'effet de l'un entraîne automatiquement la présence de l'autre critère.

Aucune fonction de moyenne (même ordonnée) ne permet de modéliser ce genre de comportement [58]. Il faut alors passer à des méthodes d'agrégation plus complexes, telles que l'intégrale de *Choquet* qui généralise en quelque sorte les autres fonctions d'agrégation comme les moyennes classiques. L'intégrale de Choquet est aussi appelée intégrale floue, car elle est définie par rapport à un ensemble de coefficients appelé *mesure floue* ou encore *capacité*.

Definition 1.3. (Mesure floue ou Capacité) *Soit $\mathcal{N} = \{1, 2, \ldots, n\}$ un ensemble de n critères. Une* capacité [31] *aussi appelée* mesure floue [153] *est une fonction d'ensemble $v : 2^{\mathcal{N}} \rightarrow [0, 1]$ qui est monotonique (i.e., $v(\mathcal{A}) \leq v(\mathcal{B})$ quand $\mathcal{A} \subset \mathcal{B}$) et qui satisfait $v(\emptyset) = 0$. La mesure est normalisée si, en plus, $v(\mathcal{N}) = 1$.*

Donc, on voit clairement qu'une mesure floue, telle qu'utilisée en analyse décisionnelle multicritères, est un ensemble de 2^n coefficients réels où chaque valeur peut être considérée comme une mesure du degré d'importance d'une *combinaison* particulière de critères (également appelée une *coalition* de critères en théorie du jeu). Notons que la condition de monotonicité impose que l'importance d'une coalition ne peut pas diminuer si on lui ajoute de nouveaux éléments.

Etant donné la définition d'une mesure floue, nous pouvons à présent définir l'intégrale de Choquet.

Definition 1.4. (Intégrale de Choquet) [31] *L'intégrale (discrète) de Choquet appliquée sur un vecteur de critères $\mathbf{z}$ et définie par rapport à une mesure floue v est donnée par*

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} z_{(i)} \left[v(\{j | z_j \geq z_{(i)}\}) - v(\{j | z_j \geq z_{(i+1)}\}) \right] \tag{1.5}$$

où $z_{(1)} \leq z_{(2)} \leq \ldots \leq z_{(n)}$, i.e., $z_{(i)}$ est le i^{eme} plus grand élément du vecteur d'entrée $\mathbf{z}$.

Par exemple, si $n = 3$ et $z_2 \leq z_1 \leq z_3$. Alors, en utilisant l'équation ci-dessus, on obtient:

$$C_v(z_1, z_2, z_3) = z_2 \left[v(\{2,1,3\}) - v(\{1,3\})\right] + z_1 \left[v(\{1,3\}) - v(\{3\})\right] + z_3 v(\{3\})$$

Ceci montre clairement que le score global sera influencé par les coefficients quantifiant les interactions entre les différents critères, en particulier pour les scores les moins élevés (z_2 et z_1 dans cet exemple).

Il est important de noter que l'intégrale de *Choquet* généralise toutes les fonctions de moyenne discutées précédemment. Celles-ci ne sont en fait que des cas particulier de C_v par rapport à des mesures floues dégénérées. La description d'autres propriétés formelles de cette méthode d'agrégation, ainsi qu'une étude de cas pratique et une comparaison avec (W)OWA, sont disponibles dans le texte en anglais, vers lequel nous renvoyons le lecteur intéressé.

3. Validation expérimentale

Cette Section donne un résumé des expérimentations effectuées sur deux ensembles de données réelles collectées sur Internet. La première application démontre l'utilité de notre méthode d'attribution dans l'analyse de traces d'attaque réseau collectées pendant plus de deux ans par des honeypots distribués mondialement. Dans la seconde application, nous démontrons l'aspect générique de notre méthode en l'appliquant à des données différentes, qui caractérisent des sites web malveillants, ou hébergeant des logiciels antivirus factices (appelés *rogue antivirus*) dans le but d'escroquer des utilisateurs peu méfiants. Quelques résultats expérimentaux illustrent le genre d'informations assez riches que notre méthode permet d'obtenir, en particulier concernant les causes et les propriétés de ces phénomènes à grande échelle, ainsi que l'analyse des *modes opératoires* des attaquants.

3.1 Application aux traces d'attaque réseau

Pour cette 1ère validation expérimentale, nous avons utilisé un ensemble de traces d'attaque réseau collectées sur Internet par des senseurs appelés honeypots, et déployés dans le contexte du Projet *Leurré.com* [86, 85, 122]. Pour rappel, un *honeypot* est une ressource informatique dont l'intérêt réside dans le fait d'être sondée, attaquée ou exploitée. Vu qu'un honeypot n'a aucune valeur de production, il ne devrait observer aucune activité, et donc toute tentative de connexion envers celui-ci est considérée comme suspecte, voire mal-intentionnée. Par extension, un réseau distribué de honeypots est appelé un *honeynet*, tel que celui maintenu par *Leurré.com*.

Données collectées par le honeynet

Dans le projet *Leurré.com*, chaque source IP observée par un honeypot est assignée à un certain cluster d'attaque (ou *attack cluster* [121, 85]) en fonction des caractéristiques réseau laissée par celle-ci sur le senseur, i.e.: nombre d'adresses IP contactées sur le senseur, nombre de paquets et d'octets échangés, durée de l'attaque, durée moyenne entre l'arrivée de deux paquets, séquence de ports visés, et charge utile des paquets envoyés. En d'autres mots, toutes les sources IP malveillantes appartenant au même cluster d'attaque ont laissé

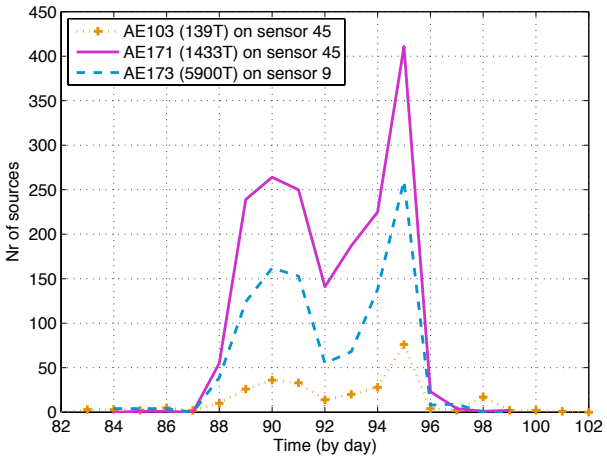


Figure 1.3: Illustration d'un $\mathcal{M}$-event composé de 3 μ-events qui sont corrélés sur 2 capteurs différents, et visant 3 ports différents (observés en avril 2008).

la même empreinte sur un honeypot, et ont donc le même profil d'attaque. Ceci nous mène à la notion d'événement d'attaque (ou *attack event*):

Definition 1.5. ***(μ-event).*** *Un* micro attack event *(ou μ-event) est un groupe de sources IP ayant le même profil d'attaque, et dont une activité coordonnée est observée par un certain senseur pendant une durée déterminée.*

La Fig. 1.3 illustre cette notion en représentant les séries temporelles de trois μ-events (i.e., nombre de sources par jour), tels qu'observés par deux senseurs différents pendant le même intervalle de temps, et visant trois séquences de port différentes (dans ce cas-ci, un seul port à chaque fois). Par extension, un *macro-event* (ou $\mathcal{M}$-event) est défini comme l'ensemble des μ-events observés sur un même intervalle de temps, et dont les activités sont fortement corrélées (tel que ceux de la Fig. 1.3). Le processus d'identification de tels événements d'attaque dans le trafic collecté par les honeypots a été expliqué en détail dans [117]. Pour cette validation expérimentale, nos données contiennent 2.454 μ-events collectés par 40 plateformes situées dans 22 pays différents, sur une période allant de Sep 2006 à Nov 2008. Ces données représentent globalement l'activité de 2.538.922 sources IP malveillantes, qui ont été assignées à 320 profils d'attaque distincts.

Le but d'appliquer notre méthode d'attribution sur ces événements d'attaque consiste à identifier des phénomènes globaux auxquels une série d'événements peut être attribuée, en s'appuyant sur différentes caractéristiques d'attaque que toutes ces sources malveillantes ont en commun. Ces phénomènes d'attaque forment une sorte d'essaim ou de "nuée malveillante", que l'on a baptisée *Misbehaving Cloud* (MC), et qui se propage sur Internet à la recherche de nouvelles machines vulnérables. Notre méthode permet non seulement de les identifier, mais également d'en étudier les propriétés ainsi que les *modes opératoires* des sources qui les composent.

Application de la méthode d'attribution multicritères

Nous avons tout d'abord sélectionné un certain nombre de caractéristiques d'attaque qui nous semblent pertinentes dans l'application de la méthode d'attribution décrite précédemment. Les deux premières caractéristiques retenues ont trait aux distributions spatiales des sources malveillantes impliquées dans les μ-events, c'est à dire les pays d'origine où se trouvent ces sources (en faisant correspondre les adresses IP aux pays correspondants), que l'on dénotera par F_{geo}, et les réseaux d'origine des sources (dénoté par F_{sub}). Ce choix est motivé, entre autres, par l'existence de réseaux dits "infestés" [32], i.e., des réseaux qui ont tendance à regrouper un grand nombre de machines infectées (machines dites *zombies*) pendant des périodes de temps assez longues.

Ensuite, nous avons choisi une caractéristique (dénoté par F_{time}) qui est liée à la manière dont les sources malveillantes ciblent les plateformes. Pour cela, nous profitons de la technique d'identification des événements d'attaque, qui nous donne la corrélation existant entre certains événements et qui peut provenir de la coordination inhérente aux botnets (dû à l'exécution coordonné de commandes données par le botmaster aux machines zombies).

A côté des origines et de la corrélation temporelle, nous avons aussi sélectionné le *type d'activité* effectuée par les machines attaquantes. En fait, les logiciels malveillants (e.g., vers, bots, etc) sont souvent conçus de telle sorte à intégrer un certain nombre d'exploits visant différentes vulnérabilités logicielles afin d'optimiser leur propagation. Il semble donc raisonnable d'intégrer, pour chaque μ-event, la séquence de ports visés par les sources comme caractéristique d'attaque, que l'on dénotera par F_{ps}.

Finalement, nous avons aussi décidé d'intégrer comme caractéristique le ratio d'adresses IP communes entre deux μ-events. Il va de soi que cette caractéristique est évolutive, c.à.d., les machines infectées peuvent être soit nettoyées, ou recevoir une adresse IP différente de jour en jour, ou encore de nouvelles machines peuvent être compromises et joindre à leur tour le botnet dont les origines peuvent donc évoluer d'événement en événement. Toutefois, compte tenu de la taille relativement grande de l'espace IP, il est raisonnable de considérer que deux μ-events partageant un nombre considérable d'adresses IP en commun puissent probablement être liés au même phénomène.

En résumé, nous considérons l'ensemble de caractéristiques d'attaque suivante pour l'application de la méthode:

$$\mathcal{F} = \{F_{geo}, F_{sub}, F_{time}, F_{ps}, F_{cip}\}$$

où

$$\begin{cases} F_{geo} = \text{geolocation, as a result of mapping IP addresses to countries;} \\ F_{sub} = \text{distribution of sources IP addresses (grouped by Class A-subnet);} \\ F_{time} = \text{degree of temporal coordination on the targeted platforms;} \\ F_{ps} = \text{port sequences probed or exploited by malicious sources;} \\ F_{cip} = \text{feature representing the ratio of common IP addresses among sources;} \end{cases}$$

Dans la deuxième étape de la méthode, un graphe non-orienté pondéré est créé pour chacune de ces caractéristiques séparément, et l'algorithme de regroupement des ensembles dominants se charge d'en extraire des cliques de poids maximal. Pour ce faire, une distance

appropriée à chaque caractéristique doit être définie. Pour mesurer les similarités entre les distributions créées par rapport à F_{geo} et F_{sub}, nous avons choisi la distance statistique de Jensen-Shannon, telle que définie par l'équation 1.2. Concernant les vecteurs caractéristiques créés pour F_{targ}, F_{ps}, et F_{cip}, il s'agit de simples ensembles de valeurs, donc nous avons utilisé le coefficient de Jaccard comme mesure de similarité. Si s_1 et s_2 sont deux ensembles de valeurs (par exemple, deux ensembles de ports visés par les sources de deux μ-events), alors le coefficient de Jaccard est défini par:

$$JC(i,j) = \frac{|s_1 \bigcap s_2|}{|s_1 \bigcup s_2|} \tag{1.6}$$

Afin d'évaluer les résultats de clustering obtenus par la méthode des *dominant sets*, il peut être utile de les visualiser sur un graphique. Nous avons donc créé pour chaque caractéristique d'attaque un graphe de dispersion en utilisant une technique de réduction de dimensionnalité appelée *t-distributed Stochastic Neighbor Embedding*, ou *t-SNE* [175]. La Fig. 5.9 (a), à la page 117, représente les clusters (ou groupes) obtenus par rapport à la dimension F_{geo}. Sur ce graphe, chaque point représente la distribution géographique d'un μ-event donné, et sa couleur indique son appartenance à un certain cluster. Pour illustrer le type de motifs de corrélation trouvés, les centroïdes de certains clusters sont indiqués sur le même graphe.

Ce genre de visualisation donne immédiatement un aperçu global des relations existants entre événements d'attaque par rapport à un point de vue donné (géographique dans ce cas-ci). Par exemple, nous pouvons facilement observer les similitudes entre groupes de points proches les uns des autres (impliquant en général des pays assez grands ou assez populaires au point de vue origine des attaques), tandis que des groupes éloignés ne partagent quasiment rien en commun. Des conclusions similaires peuvent être visualisées par rapport aux autres dimensions, i.e., les réseaux IP d'origine à la Fig. 5.9 (b), les corrélations temporelles à la Fig. 5.9 (c), et les séquences de ports à la Fig. 5.9 (d).

Ces résultats de clustering apportent des points de vue intéressants sur les phénomènes d'attaque observés par les honeypots. Intuitivement, nous pouvons facilement imaginer qu'une combinaison de tous ces points de vue puisse encore mieux mettre en évidence ces phénomènes. Toutefois, la façon de les combiner reste encore à définir: comment combiner tous ces clusters de manière intelligente, c'est à dire, quelles caractéristiques devons-nous combiner afin de pouvoir observer un phénomène X ou Y? Par exemple, les caractéristiques d'un phénomène de type *botnet* évoluent souvent dans le temps: ses origines peuvent changer (dû à un changement dans la composition des *bots*), et ses activités peuvent aussi changer (à cause de différents ordres donnés par le *botmaster*). Par ailleurs, il est parfois difficile de séparer avec précision les motifs de corrélation par un clustering, à cause de l'aspect flou des phénomènes réels. Ceci peut entraîner un certain recouvrement entre différents clusters (e.g., certains pays sont plus populaires que d'autres, certains ports Windows sont plus visés que d'autres, etc).

Il faut donc fusionner les caractéristiques d'attaque de manière cohérente, en faisant appel à des techniques d'analyse décisionnelle multicritères (MCDA). Pour cette application, nous avons choisi d'appliquer l'opérateur OWA_w tel que défini par la formule 1.4. Nous avons utilisé notre expérience en analyse de menaces sur Internet afin de définir les coefficients du vecteur $\mathbf{w}$, de façon à modéliser les phénomènes de manière appropriée. Nous avons émis l'hypothèse que deux μ-events provenant d'un même phénomène (i.e.,

le même *Misbehaving Cloud*) doivent être corrélés par au moins deux ou trois caractéristiques différentes (parmi les cinq F_k considérées). Notons que ce ne sont pas forcément les deux mêmes caractéristiques qui doivent corréler chaque paire d'événements d'un même phénomène.

Pour modéliser ce comportement, nous avons donc défini un vecteur de coefficients

$$\mathbf{w} = (0.1, 0.35, 0.35, 0.1, 0.1)$$

qui peut être interprété comme: au moins trois caractéristiques doivent être fortement corrélées, mais la 1^{ere} est de moindre importance dans le score final (car une seule caractéristique corrélée peut être due à une simple coïncidence). Ces coefficients doivent être choisis avec précaution afin d'éviter un couplage fortuit entre μ-events qui n'ont pas été causés par le même phénomène, comme par exemple des événements ayant des origines similaires (e.g., des pays populaires) et visant les mêmes ports quasiment dans le même intervalle de temps. En considérant différents cas extrêmes, on peut arriver à minimiser la valeur de décision finale pour ces cas indésirables.

Il est également intéressant de noter que certaines caractéristiques d'attaque peuvent présenter une interaction (positive ou négative), comme par exemple F_{geo} et F_{sub} qui sont en quelque sorte redondants (i.e., ces caractéristiques sont toutes deux liées aux origines des phénomènes). Pour modéliser ces interactions, nous pouvons faire appel à l'intégrale de *Choquet*, tel qu'introduit précédemment. Nous renvoyons le lecteur intéressé au texte anglais pour plus d'information concernant les expérimentations effectuées à l'aide de cette fonction de décision offrant une modélisation plus flexible (mais toutefois plus complexe).

Résultats expérimentaux

Partant des 2.454 μ-events, la méthode a pu identifier 83 "nuées malveillantes" (notées MC, pour *Misbehaving Clouds*), qui regroupent 1.607 μ-events comprenant un total de 506.835 sources malveillantes. La plupart des MC contiennent assez peu d'événements et de sources, mais environ 10% des MC contiennent tout de même plus de 20.000 sources *observables*[6] Concernant leur durée de vie, 67% des MC existent pendant moins de 50 jours, mais environ 22% de celles-ci ont réussi à survivre pendant plus de 200 jours. Dans certains cas extrêmes, nous avons pu observer certaines nuées malveillantes actives pendant près de 700 jours. Enfin, il est intéressant de noter que dans 94% des cas, ces MC ne sont observées que par 10 plateformes maximum.

Ces diverses caractéristiques suggèrent donc que les causes fondamentales sous-jacentes de ces nuées malveillantes sont des phénomènes relativement stables et localisés. En d'autres mots, on observe différents phénomènes d'attaque en fonction de l'endroit sur Internet, mais leurs modes opératoires restent stables sur de longues périodes d'activité. Ceci suggère également que nous ne sommes pas vraiment en mesure de stopper ces phénomènes dans des délais raisonnables.

Concernant les ports et services visés par les sources malveillantes, ils impliquent quasiment tous les services couramment attaqués, tels que NetBios (ports 139/TCP, 445/TCP), Windows DCOM Service (port 135/TCP), Virtual Network Computing (port 5900/TCP),

[6]Il est important de signaler que les tailles de ces phénomènes ne reflètent que la partie visible ou observable par les senseurs. Les tailles réelles de ces nuées malveillantes sont fort probablement beaucoup plus grandes.

Microsoft SQL Server (port 1433/TCP), Windows Messenger Service (ports 1025- 1028/UDP), Symantec Agent (port 2967/TCP), etc. Ceci n'est pas vraiment surprenant, puisque la plupart de ces services sont également réputés pour être largement exploités par plusieurs familles de logiciels de type bot, comme SDBot, Spybot, Agobot, GT Bot et RBot [136, 79].

Les origines spatiales des phénomènes identifiés sont assez diverses, mais nous pouvons tout de même observer quelques groupes de réseaux IP et de pays qui semblent être fortement infestés de machines zombies. Pour illustrer ce point, nous avons représenté à la Fig. 1.4les distributions cumulatives (CDF) des adresses IP impliquées dans cinq des plus grands phénomènes (où l'axe des abscisses représente le premier octet des adresses IP). Il apparait assez clairement sur cette figure que les sources de ces nuées malveillantes sont fortement concentrées au sein de certains blocs d'adresses IP. Ceci est d'ailleurs cohérent avec d'autres mesures expérimentales effectuées globalement sur le "bruit de fond Internet" (*Internet background radiation* [30, 114, 195]). Toutefois, un avantage important de notre méthode est qu'elle permet de distinguer ces différents phénomènes et d'en étudier les propriétés dynamiques, malgré le fait que les sous-réseaux d'origine de ces phénomènes soient souvent assez proches les uns des autres.

Des études de cas plus détaillées de nuées malveillantes ont été présentées dans plusieurs publications, telles que dans [167, 168, 40]. A des fins d'illustration, nous donnons ci-dessous un seul exemple de phénomène de ce type, à savoir un cas de *botnet coordonné*. Nous renvoyons le lecteur intéressé vers les publications mentionnées ci-dessus pour d'autres exemples de phénomènes (tels que des nuées de vers Allaple, des spammers UDP Windows Messenger, ou encore un phénomène d'aberrations P2P).

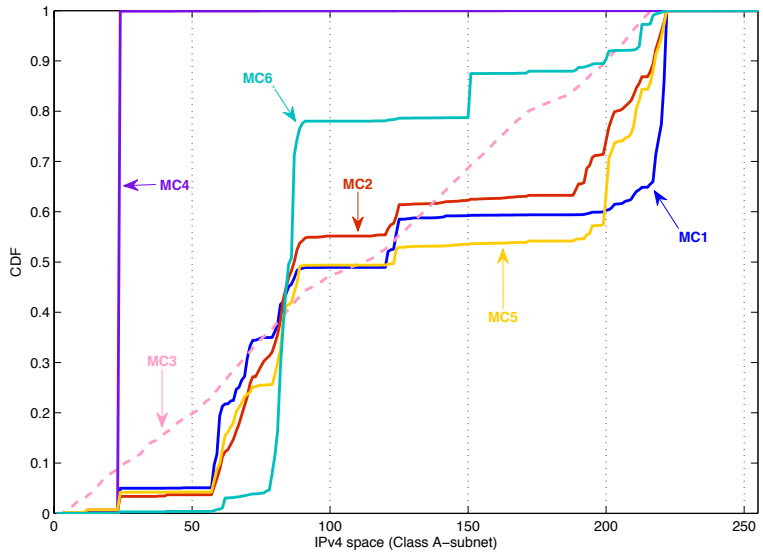


Figure 1.4: Distributions Cumulatives (CDF's) des réseaux IP d'origine de quelques phénomènes de type MC assez importants.

Botnet coordonné

Parmi les nuées malveillantes détectées à l'aide de la méthode d'attribution, $MC1$ est un cas intéressant qui semble impliquer un *botnet coordonné*. Les machines zombie de celui-ci ont visé principalement des ports Windows de partage de fichiers (445T and 139T). En analysant en détail la synchronisation et la coordination temporelle des sources attaquantes parmi les 143 μ-events composant ce phénomène, nous en avons déduit qu'il s'agissait fort probablement d'un botnet.

A la Fig. 1.5, qui représente les séries temporelles des événements appartenant à $MC1$, on peut remarquer que ce phénomène est marqué par quatre vagues d'activité principale pendant lesquelles le botnet scanne aléatoirement cinq sous-réseaux Internet différents, à la recherche de nouvelle victimes. En analysant les distributions d'adresses IP d'origine, on peut clairement observer une certaine évolution, probablement due au nettoyage de machines infectées et au recrutement de nouveaux zombies. Au total, plus de 64.000 sources IP ont pu être atribuées à cette armée de machines qui a survécu sur Internet pendant au minimum 320 jours.

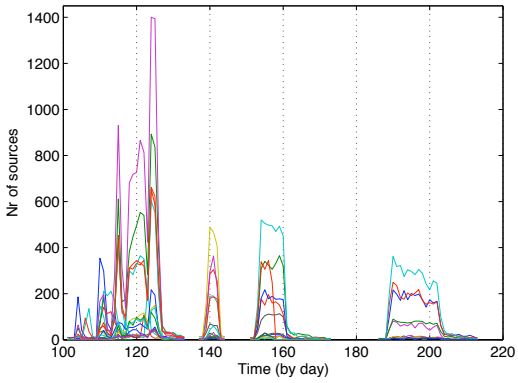


Figure 1.5: Séries temporelles coordonnées des μ-events appartenant à $MC1$.

Enfin, l'aspect sans doute le plus intéressant de ce phénomène mis en évidence par la méthode d'attribution, est qu'il semble coordonner ses zombies selon deux communautés distinctes de machines: une communauté assez vaste au sein de $MC1$ qui ne fait apparemment que scanner les machines présentes dans les sous-réseaux visés (à l'aide de paquets ICMP), tandis qu'une autre communauté de machines (nettement plus petite) se charge d'attaquer les machines actives découvertes précédemment par les scanners. Les origines spatiales de ces deux communautés sont complètement différentes, et pourtant les machines de la communauté attaquante semblent bien connaitre les adresses IP des machines actives sur lesquelles les ports 139 et 445/TCP sont ouverts. En effet, les machines de type "scanners" visent de manière égale les trois honeypots de chaque plateforme du honeynet (33% de requêtes sur chaque IP ciblée), tandis que les attaquants ne visent uniquement les deux premières IP qui émulent une machine Windows 2000 (et où les ports Windows sont effectivement ouverts). La troisième IP de chaque plateforme émule, quant à elle, une machine Linux. Ceci montre qu'une coordination *interne au botnet* est nécessaire afin d'optimiser ses ressources. Dans ce cas de figure, il est fort probable que les scanners utilisent une technique de prise d'empreinte d'OS, telle que *P0f* ou *Xprobe*), et transmettent ensuite

cette information vers un plan de contrôle qui permet de diriger les attaquants vers les "bonnes adresses IP".

A la Fig. 5.17, nous avons produit un autre type de visualisation basée sur des graphes multidimensionnels qui représentent toutes les corrélations liant les μ-events par rapport aux différentes caractéristiques d'attaque considérées. Ceci permet à l'analyste de visualiser un phénomène entier à l'aide d'une seule et même image d'ensemble.

3.2 Application aux sites web hébergeant des logiciels antivirus factices

Une des conditions nécessaires dans la conception de la méthode d'attribution d'attaque est son applicabilité à un large éventail de problèmes ou de phénomènes liés aux menaces globales sur Internet, à l'analyse de renseignements (*intelligence analysis*) dans le cyber-espace, ou plus généralement à tout ensemble de données de sécurité.

Pour démontrer cet aspect, nous donnons ici un aperçu des expérimentations effectuées par rapport à un autre problème émergeant sur Internet, à savoir celui des logiciels antivirus factices (appelés aussi *rogue antivirus* [159]). Un tel logiciel prétend être une application tout à fait légitime permettant de se protéger contre les virus et autres menaces, mais en réalité le logiciel ne fournit en général aucune protection. Dans certains cas, il peut même installer (de manière intentionnelle) du code malveillant, alors qu'il est supposé protéger l'utilisateur contre celui-ci.

Nous décrivons ci-après comment nous avons pu profiter de notre méthode d'attribution d'attaque pour analyser les *campagnes* par lesquelles ce type de malware est distribué, i.e., quelles techniques automatisées, quelle infrastructure de serveurs et quels efforts coordonnés les cyber-criminels semblent utiliser afin de propager et distribuer leur logiciel factice à des utilisateurs non méfiants. Nous avons analysé les données collectées sur 5.852 sites web suspects, tel qu'observé par un tracker appelé HARMUR pendant une période de deux mois (juillet-août 2009). Le but principal de cette analyse est d'identifier l'infrastructure serveur mise en place lors de campagnes de distribution de faux anti-virus, et les efforts coordonnés des criminels responsables pour celle-ci.

Il est en effet relativement raisonnable de penser que de telles campagnes soient organisés et gérées par un nombre limité de personnes qui vont réutiliser, à différentes étapes de chaque campagne, les mêmes outils, techniques ou stratégies (à cause de coûts évidents liés au développement). Par conséquent, nous avons appliqué la méthode de regroupement multicritères aux données spécifiques caractérisant ces sites web malveillants dans le but d'identifier des motifs émergeants dans la manière dont ces domaines sont créés, regroupés et interconnectés les uns aux autres.

Ecosystème des *rogue antivirus*

Il y a deux manières assez répandues par lesquelles un logiciel anti-virus factice arrive à s'installer sur des machines victimes. Premièrement, des techniques d'ingénierie sociale (comme des bannières de publicité, des fenêtres pop-up ou des messages attractifs sur des blogs, ou encore envoyés via des spams) sont utilisées pour convaincre facilement des utilisateurs novices d'installer un outil antivirus gratuit, qui est nécessaire pour remédier à des menaces (inexistantes) soi-disant trouvées sur l'ordinateur de la future victime. Une deuxième tactique, un peu plus sournoise, consiste à attirer des victimes vers des sites web

malveillants qui tentent d'exploiter des vulnérabilités existant dans les navigateurs web (tel qu'Internet Explorer) ou dans l'un de ses plugins. Ceci permet aux cyber-criminels de télécharger leur malware sans qu'aucune intervention de l'utilisateur ne soit nécessaire (i.e., via une technique appelée *drive-by* downloads).

Les faux anti-virus sont distribués dans le but de générer un profit financier assez substantiel. En fait, après l'infection initiale par le logiciel, les victimes sont typiquement incitées à payer pour des services additionnels (e.g., un upgrade vers la version complète du logiciel), qui sont bien sûr à nouveau factices et complètement inefficaces comme protection antivirus. La perte financière initiale pour les victimes est en moyenne entre $30 et $100, perte qui peut éventuellement être aggravée par le vol du numéro de carte bancaire de la victime servant à une fraude ultérieure. Quelques exemples de logiciels anti-virus factices assez répandus (tel que rapporté par Symantec pour la période Juillet 2008 - Juin 2009 [159]) sont SpywareGuard 2008, AntiVirus 2008, AntiVirus 2009, Spyware Secure, and XP AntiVirus.

Malgré des techniques relativement peu sophistiquées, ce phénomène de faux anti-virus a émergé comme une menace sérieuse sur Internet, certainement en termes de population concernée (plus de 43 millions de tentatives d'installation rapportées par Symantec durant la même période [159]), de nombre de variantes distribuées sur Internet, et le volume de profits générés par les cyber-fraudeurs dont le business model s'appuie sur une structure de distributeurs affiliés qui sont rétribués par des commissions par nombre d'installations. Des bénéfices mensuels allant jusqu'à $332.000 ont été rapportés par [77, 159] uniquement pour les commissions d'un affilié, tel qu'observé sur un site web de distribution appelé `TrafficConverter.biz`.

Par conséquent, pas mal d'études et d'analyses ont été effectuées par la communauté sécurité suite à l'emergence de ce phénomène inquiétant [159, 112, 113]. Toutefois, la plupart de ces études se sont penchées uniquement sur certains aspects techniques tels que l'installation de ces malwares, leur technique de marketing, ou certains aspects quantitatifs comme le nombre de sites web distribuant les faux anti-virus, et leur localisation géographique. Par contre, les interconnections entre l'infrastructure serveur, les logiciels anti-virus factices distribués et le processus de création et d'enregistrement des noms de domaine n'ont pas encore été explorées. Un des objectifs de cette validation expérimentale était donc de combler cette lacune. Les résultats de notre analyse multicritères ont ensuite été présentés dans [159, 36].

Données HARMUR

Les données expérimentales de cette analyse proviennent de HARMUR, qui signifie **H**istorical **AR**chive of **M**alicious **UR**ls. Ce projet est une initiative récemment lancée dans le cadre du projet WOMBAT [188, 189], et il a comme but de traquer et collecter des informations détaillées sur la nature et l'évolution des menaces existants sur le Web.

HARMUR surveille donc toute une série de sites web suspects et collecte des informations concernant l'hébergement éventuel d'exploits visant les navigateurs web, ou l'existence de *drive-by downloads* visant à installer du code malveillant à l'insu des utilisateurs en exploitant des vulnérabilités dans le navigateur. Pour cela, HARMUR s'appuie sur plusieurs sources d'information spécialisées, et pour chaque site web surveillé, les informations suivantes sont rassemblées et stockées dans une base de données:

- **Norton Safeweb**. Grâce au service de réputation *Norton Safeweb*[7], HARMUR collecte des données détaillées sur les menaces détectées par ce service en visitant chaque site web surveillés.
- **Google Safebrowsing**. Pour vérifier l'aspect malveillant d'un site, HARMUR s'appuie aussi sur l'information fournie par les listes noires de Google Safebrowsing[8] .
- **Mapping DNS**. HARMUR garde aussi les correspondances entre noms de domaine, serveurs DNS autoritaires et adresses IP correspondants aux serveurs HTTP hébergeant les sites web suspects.
- **Information Whois**. Pour chaque site web, les informations d'enregistrement de nom de domaine *Whois* sont récupérées et stockées dans la DB de HARMUR.
- **Localisation et Système Autonome**. Des informations de localisation géographique (pays d'origine) et réseau (systèmes autonomes, ou AS) sont collectées pour chaque serveur HTTP associé à un site web.
- **Statut et version de serveur HTTP**. Quand c'est possible, des informations quant à la disponibilité actuelle des serveurs web, ainsi que la version de leur logiciel (telle qu'annoncée dans les en-têtes HTTP) sont stockées dans la base de données.

En réitérant ce processus de collecte d'informations à intervalles réguliers, nous avons à notre disposition un ensemble de données assez complet et représentatif de la structure et de la dynamique du paysage des menaces web, en particulier celles associées aux faux logiciels AV.

Application de la méthode d'attribution multicritères

Pour notre analyse, nous avons considéré les données concernant 5.852 noms de domaines collectées sur une période de deux mois (juillet-août 2009). Ces noms de domaine pointaient vers 3.581 adresses IP distinctes (i.e., serveurs web différents), et ils ont été choisis en raison de leur implication possible dans l'hébergement de faux logiciels AV. De plus, une petite partie de ces serveurs web hébergeaient en plus d'autres codes malveillants (e.g., des chevaux de Troie, backdoors, keyloggers, etc).

Environ 45% de ces domaines ont été enregistrés via seulement 29 sociétés d'enregistrement (appelées des *registrars*), parmi les centaines de sociétés existants sur Internet. Ceci pourrait indiquer que les distributeurs de faux logiciels AV choisissent certains *registrars* spécifiques en raison de leur laxisme (intentionnel ou non) dans le contrôle des noms de domaine enregistrés. Concernant la répartition géographique des serveurs web (Fig. 1.6), nous avons observé que 53% des serveurs étaient aux USA, ce qui peut aussi être du à un artefact dans la manière dont les sites *rogue AV* sont identifiés (i.e., il est plus facile d'identifier des escroqueries de ce type en anglais que dans d'autres langues, comme le mandarin).

Parmi les différentes informations surveillées par HARMUR, nous avons donc sélectionné un certain nombre de caractéristiques de sites web qui peuvent nous indiquer comment des phénomènes à grande échelle semblent être organisés sur Internet par un individu ou un groupe spécifique.

- **Adresse email du *registrant***. Lors de la procédure d'enregistrement d'un domaine auprès d'un registrar, l'adresse email du propriétaire du domaine (appelée le

[7] http://safeweb.norton.com
[8] http://code.google.com/apis/safebrowsing/

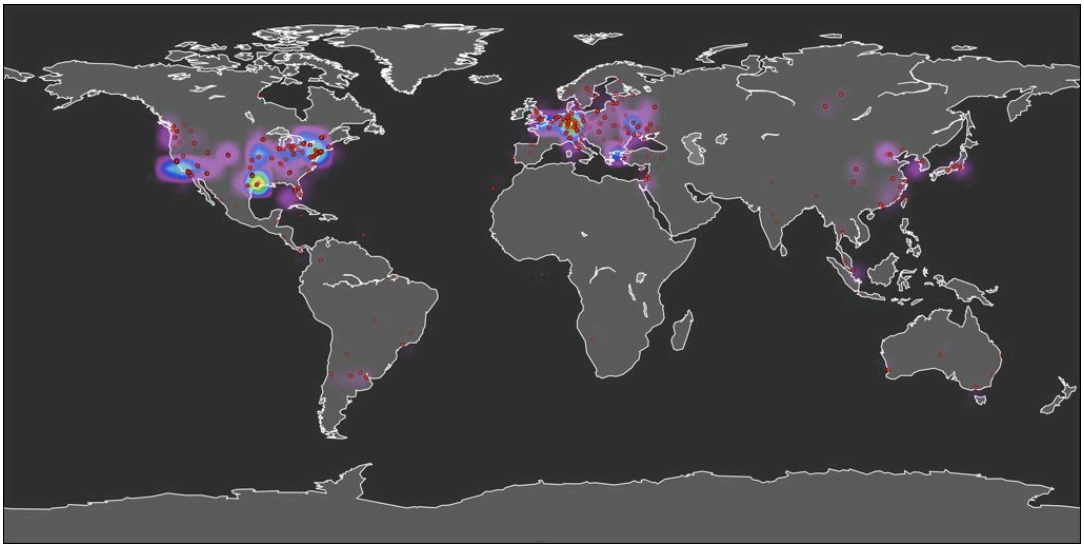


Figure 1.6: Distribution géographique des serveurs hébergeant de faux antivirus (rogue AV) pendant la période Juillet-Aout 2009.

registrant) doit être communiquée.

- **Nom du *registrar***. Le nom complet de l'autorité d'enregistrement (registrar), telle qu'identifiée par les données Whois.
- **Adresses IP / adresses classe C/classe B des serveurs**. Pour permettre l'identification des serveurs appartenant à la même infrastructure, nous avons considéré séparément les adresses IP complètes des serveurs, ainsi que leurs préfixes réseau de classe C (/24) et de classe B (/16).
- **Adresse IP du serveur DNS**. Il s'agit de l'adresse IP du serveur de noms ayant autorité.
- **Nom de domaine enregistré**. Nous avons décidé d'utiliser le nom de domaine lui-même comme caractéristique, afin de détecter des motifs ou des schémas spécifiques dans les dénominations choisies par les fraudeurs.

En résumé, l'ensemble de caractéristiques sélectionnées est défini par:

$$\mathcal{F} = \{F_{Dom}, F_{IP}, F_{Cl.C}, F_{Cl.B}, F_{Reg}, F_{NS}\}$$

ce qui servira à la méthode d'attribution multicritères pour lier des domaines suspects au même phénomène de faux anti-virus. Quelques exemples concrets de ces caractéristiques sont donnés à la Table 6.2 (page 145) pour certains domaines surveillés par HARMUR.

Dans une 2^{eme} phase, nous allons créer un graphe de similarités inter-domaines par rapport à chaque caractéristique F_k, en définissant une métrique de distance appropriée. Vu que les données créées par rapport à F_{IP}, $F_{Cl.C}$, $F_{Cl.B}$ et F_{NS} sont simplement des ensembles d'adresses IP, nous pouvons réutiliser le coefficient de similarité de Jaccard (voir eq. 1.6). Concernant F_{Reg}, nous avons considéré une mesure de similarité un peu plus heuristique, à savoir:

(1) nous vérifions d'abord l'égalité entre les adresses email des *registrants*;
(2) si les adresses sont différentes, nous comparons les sous-caractéristiques suivantes: le

domaine email, le username, et la présence de mots-clés typiques liés aux faux logiciels AV. La valeur finale de similarité est alors calculée via une moyenne pondérée, avec les coefficients suivants: [0.2, 0.2, 0.5].

Finalement, nous avons cherché une mesure de similarité pour F_{Dom} qui puisse représenter certaines similitudes dans la manière dont les noms de domaine ont été créés, par exemple en identifiant certains motifs ou séquences de caractères que des domaines pourraient avoir en commun. Nous avons réalisé cet objectif en utilisant la distance de *Levenshtein*, qui correspond au nombre minimum d'opérations nécessaires pour transformer une certaine séquence de caractères en une autre séquence (où une *opération* peut être soit une insertion, une suppression, ou encore une substitution d'un seul caractère). Vu que Levenshtein donne une distance, nous l'avons ensuite converti en similarité à l'aide de la fonction de transformation suivante [143]:

$$sim_{ij} = exp(\frac{-{d_{ij}}^2}{\sigma^2})$$

où d_{ij} est la distance entre les noms de domaine i et j, et σ est une constante positive qui affecte le taux de décroissance de sim. Dans notre cas, nous avons défini cette constante de manière empirique à une valeur de 7, ce qui nous permet de modéliser les similarités inter-domaines de façon efficace.

Enfin, en nous appuyant sur l'analyse des caractéristiques des sites web et sur notre connaissance du domaine à propos des faux anti-virus, nous avons aussi défini un vecteur de coefficients **w** pour l'agrégation multicritères basée sur l'opérateur OWA:

$$\mathbf{w} = [\ 0.10,\ 0.10,\ 0.20,\ 0.30,\ 0.20,\ 0.10\ ]$$

En d'autres mots, nous donnons plus d'importance aux caractéristiques à partir de la troisième plus haute valeur de similarité. Les deux premiers scores ont en effet des poids moins important (0.10), et donc il est nécessaire d'avoir au moins trois corrélations fortes entre deux sites web afin d'obtenir un score final au-dessus de la valeur 0.3 ou 0.4 (valeur pouvant être utilisée comme seuil de décision). En réalité, une analyse de sensibilité a été effectuée sur cette valeur de seuil afin de déterminer des plages de valeurs adéquates (voir texte anglais pour plus de détails). Un aspect intéressant dans cette approche, c'est qu'elle libère l'analyste de la nécessité de définir *quelles caractéristiques* sont les plus pertinentes, c.à.d., lesquelles doivent être corrélées pour décider de lier deux sites web au même phénomène. Ceci est un avantage non négligeable, puisque les caractéristiques d'une même campagne de faux anti-virus peuvent évoluer dans le temps.

Résultats expérimentaux

Concernant les détails des résultats intermédiaires de clustering pour chaque caractéristique F_k considérée, nous renvoyons le lecteur intéressé au texte anglais. Dans cette Section, nous ne donnons qu'un aperçu des résultats finaux en termes de campagnes.

La méthode d'attribution multicritères a identifié 127 campagnes distinctes regroupant au total 4.549 noms de domaines liés à des faux antivirus. En moyenne, les campagnes comprennent 35.8 domaines, mais avec une grande variance en taille. En fait, 4.049 domaines sont associés à seulement 39 campagnes relativement importantes, la plus grande campagne impliquant 1.529 domaines.

Pour évaluer la cohérence des résultats, nous avons représenté à la Fig. 1.7 les indices de "compacité" (C_p), qui sont utilisés en clustering de graphes pour évaluer la qualité des clusters obtenus par leur densité interne. La figure représente les indices C_p pour les 39 plus grands phénomènes, et calculés pour chaque caractéristique séparément (i.e., chaque couleur indique une certaine caractéristique F_k). La Fig. 1.7 donne une vue globale intéressante sur la cohérence des phénomènes, et peut aussi être utilisée pour déterminer quelles caractéristiques semblent lier les les domaines web au sein d'un même phénomène P_i, c.à.d., au sein d'une même campagne de faux anti-virus. Nous pouvons aussi observer que la plupart des phénomènes ont globalement un indice de compacité assez élevé, à l'exception de P_1 et P_3 qui sont composés de plusieurs grands sous-graphes plus faiblement interconnectés (i.e., des sous -ensembles de domaines formant des sortes de "bulles" isolées, et faiblement reliées entres elles par seulement une ou deux caractéristiques).

Dans l'ensemble, nous pouvons aussi noter que ce sont les caractéristiques liées aux adresses IP qui semblent contribuer le plus à la corrélation entre sites web proposant des faux anti-virus, souvent complétées par des corrélations par rapport au noms des registrants (F_{Reg}). Il est intéressant de remarquer que certains phénomènes sont corrélés par les sous-reseaux d'origine ($F_{Cl.C}$ et $F_{Cl.B}$) des serveurs web, mais pas spécialement par leurs adresses IP (comme par exemple P_{27}).

Par contre, les corrélations liées aux noms de domaines (F_{Dom}) sont en général plus faibles, sauf pour quelques phénomènes où des motifs bien spécifiques semblent lier les domaines d'une même campagne (comme pour P_4, P_{10} et P_{34}). Finalement, remarquons aussi que chaque phénomène identifié par la méthode présente divers degrés de corrélation par rapport à chaque caractéristique individuelle, mais dans l'ensemble il y a toujours au moins trois caractéristiques différentes qui présentent des niveaux de corrélation élevée.

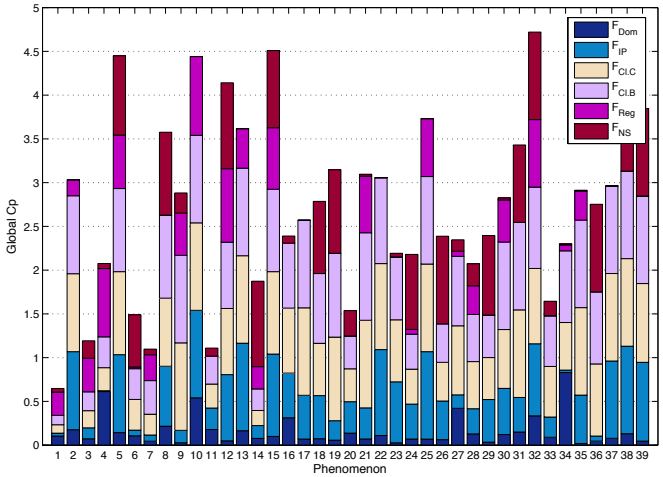
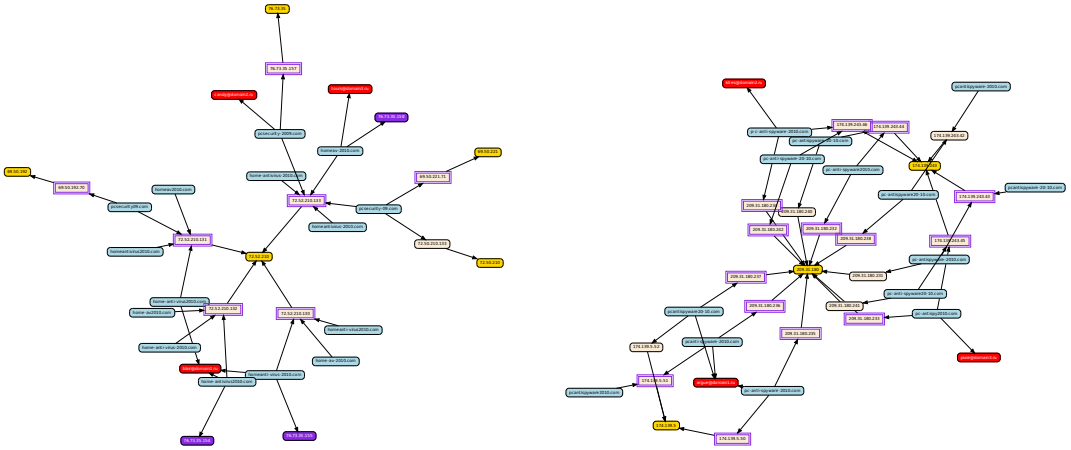


Figure 1.7: Evaluation de la cohérence des résultats à l'aide des coefficients de *compacité* (C_p) des 39 plus grands phénomènes identifiés par l'agrégation OWA. Chaque couleur représente l'indice C_p pour une des caractéristiques F_k.



(a) Campagne PC-Security. (b) Campagne PC-Anti-Spyware.

Figure 1.8: Deux campagnes de faux anti-virus. Les noeuds en bleu représentent des domaines rogue AV, ceux en jaune les sous-réseaux des serveurs web, ceux en rouge les adresses email des *registrants*. Les rectangles à double côté (en mauve) indiquent des serveurs web co-localisés avec les serveurs DNS.

Pour finir, nous nous penchons en particulier sur trois campagnes de faux anti-virus afin d'illustrer le genre de perspective unique que nos techniques MCDA permettent d'obtenir sur la *dynamique* et les modes opératoires de ces phénomènes.

Campagnes *PC-Security* et *PC-Anti-Spyware*.

Les Fig. 1.8(a) et 1.8(b) représentent graphiquement deux campagnes associées à deux clusters obtenus par l'analyse multicritères. Dans les graphes, les domaines associés aux sites web sont représentés en bleu, les sous-réseaux où sont situés les serveurs web sont indiqués en jaune, et les adresses email des *registrants* de ces domaines en rouge. Des rectangles à double côté (en mauve) indiquent des serveurs web co-localisés avec les serveurs de noms (DNS) associés aux domaines.

Bien que ces deux clusters aient été séparés (dû aux caractéristiques utilisées lors de l'agrégation multicritères), nous avons très vite pu les rapprocher grâce aux similitudes assez évidentes indiquant un même modus operandi générique utilisé pour déployer ces campagnes. En particulier, ces deux campagnes sont composées d'un nombre limité de sites web proposant des faux antivirus, et ceux-ci sont fortement corrélés par leurs adresses IP et par des motifs communs dans la composition des noms de domaine choisis. Ces noms de domaine se réfèrent d'ailleurs assez clairement à des "produits" antivirus ou antispyware de type commercial (*e.g., pcsecurity-2009.com*, *homeav-2010.com*, *pc-antispyware2010.com*). Ensuite, nous pouvons observer d'autres similitudes entre les deux campagnes, telles que:

- les deux clusters utilisent exactement le même schéma pour les noms de domaine, qui consiste à insérer des traits d'union entre certains mots fixes (*e.g., pc-anti-spyware-2010.com*, *pc-anti-spyware-20-10.com*, and *pc-antispyware-2010.com*). La seule différence entre les deux campagnes est que le mot *spyware* est remplacé par *virus*.
- tous les domaines dans chaque cluster utilisent le même *registrar* (OnlineNIC) et sont hébergés par les deux mêmes ISPs.
- Les adresses email de tous les *registrants* ont une extension de domaine ".ru".

- Les serveurs web se trouvent sur des adresses IP consécutives.

Enfin, de manière sans doute encore plus concluante, une inspection manuelle des pages web de chaque site a révélé que leur contenu était identique, à l'exception d'une seule image qui différait. Tout ceci nous permet de supposer raisonnablement que le déploiement de ces domaines hébergeant des faux antivirus a été largement automatisé en passant par un seul et unique *registrar*. De plus, le fait d'avoir réparti l'hébergement de ces deux campagnes sur deux ISPs différents indique aussi une tentative de garantir une certaine redondance au cas où un des deux clusters serait mis hors-ligne par l'ISP. Enfin, nous observons que tous les serveurs web d'hébergement étaient situés dans ce cas-ci aux US.

Une campagne à plus grande échelle.

Notre méthode d'attribution a également pu identifier d'autres clusters qui représentent des campagnes plus sophistiquées. Un tel exemple est représenté à la Fig.6.10 à la page 164. Les noeuds en mauve foncé en bas de page donnent les dates d'enregistrement des domaines; les autres noeuds suivent la même convention qu'indiquée ci-dessus (Fig. 1.8).

La Fig. 6.10 regroupe environ 750 domaines qui ont tous été enregistrés dans le domaine TLD[9] **.cn** (associé à la Chine), à des dates bien précises (8 dates seulement). Ces domaines pointent vers 135 adresses IP réparties dans 14 sous-réseaux différents, mais malgré l'extension **.cn**, la grande majorité des adresses IP de ces serveurs web sont situées en réalité aux US, en Allemagne, et en Biélorussie. En fait, aucun de ces serveurs n'a pu être localisé en Chine.

Un autre fait intéressant mis en évidence par la méthode est que le même *registrar* Chinois (*Era of the Internet Technology*) a été utilisé pour enregistrer tous ces noms de domaines, qui sont d'ailleurs tous composés de la même manière, à savoir de 5 caractères alphanumériques choisis apparemment de façon aléatoire (*wxe3x.cn*,*owvmg.cn*,...). Ceci indique fort probablement l'utilisation d'outils automatisés pour la création de ces domaines. Une autre caractéristique importante de cette campagne est que la personne ayant enregistré plus de 76% des domaines (cn@id-private.com) a utilisé un service de protection de vie privée bloquant l'accès aux données WHOIS réelles.

Finalement, une analyse manuelle des domaines représentés à la Fig. 6.10 a révélé un phénomène encore plus complexe qu'il n'y parait. Ces domaines sont en fait reliés à une fausse page de scan du disque dur de la victime potentielle, laquelle est hébergée sur un serveur web appartenant à une autre campagne. Ce genre de découverte souligne l'existence d'interconnexions assez complexes existant dans ce genre d'écosystème de menaces, interconnexions qu'il n'aurait sans doute pas été possible d'identifier sans l'utilisation de techniques de fouille de données capables de réduire un ensemble de milliers de domaines suspects, à quelques campagnes de faux anti-virus qui sont orchestrées apparemment par un groupe assez réduit d'individus.

4. Conclusions et perspectives

Les recherches effectuées dans cette thèse ont mené à une série d'observations et de nouvelles idées qui sont synthétisées dans cette Section.

[9] Top Level Domain

4.1 Contributions de recherche

Au vu des développements et des résultats expérimentaux présentés dans ce document, nous sommes à présent capables de répondre aux deux problèmes de recherche formulés dans l'introduction de ce travail.

Problème de recherche 1 (PR1): *comment pouvons-nous analyser des phénomènes d'attaque à partir de différents points de vue, de sorte qu'ils nous fassent découvrir des motifs de corrélation intéressants?*

Nous avons pu résoudre ce problème en développant une approche de regroupement non supervisée basée sur des graphes, ce qui permet à un analyste de découvrir des motifs de corrélation cachés par rapport à n'importe quelle caractéristique d'attaque. Nous avons aussi souligné l'importance de choisir des caractéristiques pertinentes, i.e., qui peuvent mener à la création de points de vue intéressants, mais aussi de choisir des métriques de distance appropriées pour la comparaison des observations, de sorte que les corrélations obtenues soient réellement représentatives des phénomènes sous-jacents.

Problème de recherche 2 (PR2): *comment pouvons-nous combiner systématiquement tous ces points de vue d'attaque par un processus d'agrégation de données, de sorte que les propriétés comportementales des phénomènes observés soient modélisés de manière adéquate?*

Nous avons présenté une solution formelle et élégante à ce problème en nous appuyant sur une analyse décisionnelle multicritères (MCDA) afin de combiner de multiple points de vue. Nous voyons plusieurs raisons à ce succès.

Premièrement, les fonctions d'agrégation utilisées en MCDA ne sont pas liées à des schémas de décision rigides. Au contraire, le vecteurs de critères peuvent être formés de variables "floues", telles que des degrés de similarité, ou d'appartenance à des clusters, ce qui permet de mieux appréhender l'aspect intrinsèquement flou des phénomènes réels ou l'incertitude liée à des mesures imparfaites.

Deuxièmement, les caractéristiques d'attaque sont rarement complètement indépendantes. Par conséquent, il est nécessaire de modéliser certaines interactions entre critères interdépendants, telles qu'une *synergie* ou bien une *redondance* dans un coalition de critères. Les fonctions d'agrégation sont particulièrement bien adaptées à cette modélisation, en particulier l'intégrale de Choquet.

Dernièrement, nous avons constaté que la nature même des phénomènes réels est en général *évolutive*. Il est donc difficile de prédire avec précision quel sous-ensemble ou combinaison de caractéristiques sera la plus pertinente dans chaque cas. L'attribution de nouveaux événements à des phénomènes connus (ou inconnus) peut donc devenir compliquée pour l'analyste. Ici également, nous avons démontré que certaines fonctions d'agrégation (telles que celles décrites dans cette thèse) permettent de s'affranchir de cette difficulté, en évitant de devoir définir de manière rigide l'importance ou la pertinence de chaque critère d'attaque.

Réponse au problème général de recherche

Sur base des réponses individuelles données aux deux problèmes de recherche ci-dessus, nous pouvons finalement donner une réponse circonstanciée au problème général qui a motivé ce travail.

> *Est-il possible d'aborder de manière systématique le problème de l'attribution d'attaque sur Internet par une méthode analytique?*

Tel que démontré tout au long de cette dissertation, le problème de l'attribution d'attaque sur Internet est clairement un problème assez complexe, d'abord parce qu'il implique un grand nombre de dimensions dans le processus d'investigation, mais également à cause de l'aspect évolutif des phénomènes d'attaque. Toutefois, nous avons démontré que ce problème pouvait être abordé au moyen d'une méthode analytique permettant de combiner de multiples points de vue de manière systématique, voire automatisée. Par cette approche analytique multicritères, nous avons aussi montré que le comportement dynamique de phénomènes d'attaque (à priori inconnus) peuvent être effectivement modélisés, et que les modes opératoires des attaquants sont également mieux mis en évidence.

En conclusion, nous pensons que notre approche de regroupement multicritères offre une contribution tout à fait intéressante au problème de l'attribution d'attaque, et qu'elle peut certainement aider les experts en sécurité à éclaircir des questions encore non élucidées concernant l'organisation d'activités cyber-criminelles, voire même aider à découvrir de nouveaux aspects liés à celles-ci. Nous sommes bien entendu ouverts à toute proposition ou opportunité d'application de cette méthode à d'autres ensembles de données que de futurs partenaires seraient prêts à partager avec nous.

4.2 Perspectives intéressantes

Comme première perspective intéressante, de plus amples recherches en analyse des menaces pourraient mener à l'identification et l'intégration de nouvelles caractéristiques à la méthode d'attribution, ce qui permettrait d'amener des points de vue supplémentaires et donc d'enrichir les informations sur les phénomènes identifiés.

Deuxièmement, nous pourrions considérer d'autres techniques de regroupement (*clustering*) que celle des ensembles dominants afin d'améliorer l'efficacité mais également l'extensibilité de la méthode. Pour pouvoir traiter des ensembles de données de taille supérieure, nous envisageons aussi le développement d'une version incrémentale de cette méthode d'attribution qui s'appuie sur une base de données. Actuellement, nous sommes en effet limités par la quantité de mémoire nécessaire au stockage de n matrices de similarité contenant chacune $m \times (m-1)/2$ éléments, ce qui conduit à une complexité en mémoire de l'ordre de $\mathcal{O}(nm^2)$. En pratique, il est toutefois possible de traiter jusqu'à 10.000 événements en utilisant 8 caractéristiques d'attaque simultanément.

Troisièmement, nous pensons que l'application de cette méthode à des ensembles de données différents concernant des activités cyber-criminelles pourrait fournir une perspective unique et un regard croisé sur ces activités. Quelques exemples de données que l'on pourrait inclure dans ce type d'analyse ont trait aux dépôts de malware (i.e., les caractéristiques statiques et dynamique liées à l'analyse des codes malveillants) et aux données plus riches collectées par des honeypots à *haute interaction* (tels que SGNET [83], qui permet

de collecter des informations détaillées sur des attaques automatisées de type injections de code du côté serveur).

Quatrièmement, nous avons étudié en détail deux classes principales de fonctions d'agrégation, à savoir les moyennes de type *Ordered Weighted Average* (OWA), et l'intégrale de Choquet. Il va de soi qu'il existe encore bien d'autres types de fonctions d'agrégation dans la litérature scientifique, mais dont l'utilité en attribution d'attaque reste à explorer.

Cinquièmement, nous envisageons le développement d'une couche d'abstraction logicielle autour des outils développés dans le cadre de ce travail, dans le but de permettre son utilisation à d'autres chercheurs d'une manière plus aisée et plus flexible.

Enfin, nous avons montré tout au long de cette thèse comment certaines méthodes de visualisation, telles que des graphes de relations et des techniques de réduction de dimensionnalité (telle que *t-SNE*), permettent d'améliorer l'aspect *situational awareness* et l'interprétation de résultats de regroupement, surtout quand de multiples dimensions sont inclues dans l'analyse globale. Dès lors, nous pensons que ce travail a ouvert une voie intéressante, et que cela suggère des recherches supplémentaires dans cette direction afin d'intégrer de nouvelles technologies issues du domaine *visual analytics* à cette méthode d'attribution d'attaque. Ceci nous permettrait d'effectuer un raisonnement combiné sur des événements de sécurité non seulement de manière systématique, mais également visuelle et interactive.

En poursuivant les efforts de recherche dans ces différentes voies, cela devrait nous offrir des perspectives prometteuses quant au développement de modèles prédictifs de menaces sur Internet, et par conséquent, permettre le développement efficace d'un système de type *Early Warning System*.

Appendix

List of Figures best viewed in color print

Reference	Short caption	Page nr
Fig.3.4	Clustering problem illustrated	p.41
Fig.3.7	Dominant sets (DM) clustering results	p.52
Fig.3.10	Clustering results of HC and K-Means	p.59
Fig.4.5	Chaining effect illustrated with connected components and DM	p.71
Fig.5.2	Visualization of malicious sources in the IPv4 space (IP map)	p.102
Fig.5.9	Visualizing clustering results for the honeynet data set	p.117
Fig.5.17	Node-link graph representing MC1	p.126
Fig.5.18	Node-link graph representing MC2	p.128
Fig.5.27	Node-link graph representing MC6	p.135
Fig.6.3	Visualizing clustering results for the rogue AV data set	p.153
Fig.6.7	Node-link graph representing RC27	p.159
Fig.6.8	Node-link graph representing RC34	p.160
Fig.6.9	Node-link graph representing RC5	p.163
Fig.6.10	Node-link graph representing RC4	p.164
Fig.6.11	Node-link graph representing RC3	p.165

Bibliography

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. pages 420–434. 2001. 43, 179

[2] Anubis. Analyzing Unknown Binaries. Available online at `http://anubis.iseclab.org`. 27, 174

[3] Arbor Networks. Estonian DDoS Attacks? A summary to date. Available online at `http://asert.arbornetworks.com/2007/05/estonian-ddos-attacks-a-summary-to-date`, May 2007. 16, 171

[4] J. G. Auguston and J. Minker. An analysis of some graph theoretical clustering techniques. *ACM*, 1970. 47, 180

[5] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. C. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *Proceedings of 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, pages 165–184, Hamburg, Germany, September 2006. 23, 174

[6] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A distributed blackhole monitoring system. In *12th Annual Network and Distributed System Security Symposium (NDSS)*, February 2005. 25, 114, 174

[7] D. Barbará, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: A testbed for exploring the use of data mining in intrusion detection. In *ACM SIGMOD Record, 30(4)*, pages 15–24, 2001. 30

[8] D. Barbará and S. J. (Eds), editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*. Springer, 2002. 30, 175

[9] D. Barbará and S. Jajodia, editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*, chapter Data Mining For Intrusion Detection - A Critical Review (K. Julisch). Springer, 2002. 30, 175

[10] P. Barford and V. Yegneswaran. *An Inside Look at Botnets*. Advances in Information Security. Springer, 2006. 15, 27, 171, 174

[11] D. Barroso. Botnets - The Silent Threat. In *European Network and Information Security Agency (ENISA)*, November 2007. 15, 171

[12] T. Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4):99–105, 2000. 21, 22

[13] M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Process.*, 18(4):349–369, 1989. 45

[14] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes:Theory and Application*. Prentice Hall, 1993. 104

[15] A. Belenky and N. Ansari. On deterministic packet marking. *Comput. Netw.*, 51(10):2677–2700, 2007. 22

[16] G. Beliakov. Shape preserving splines in constructing wowa operators: comment on paper by v. torra in fuzzy sets and systems 113 (2000) 389-396. *Fuzzy Sets Syst.*, 121(3):549–550, 2001. 75

[17] G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Berlin, New York, 2007. 32, 68, 73, 83, 86, 87, 88, 96, 176, 182, 183

[18] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957. 43, 179

[19] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc. 35, 99–109*, 1943. 44, 180

[20] F. Boutin and M. Hascoët. Cluster validity indices for graph partitioning. In *IV'04, 4th International Conference on Information Visualization, 2004,*, 2004. 57, 58, 59

[21] S. T. Brugger. Data Mining Methods for Network Intrusion Detection. In *dissertation proposal, submitted to ACM Computer Surveys (under revision), 2009*, 2009. 30, 175

[22] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *LISA'00: Proceedings of the 14th USENIX conference on System administration*, pages 319–328, Berkeley, CA, USA, 2000. USENIX Association. 22

[23] CAIDA. Home page of the CAIDA project. Available online at `http://www.caida.org`. 25, 114, 174

[24] Capture-HPC. Available online at `http://www.nz-honeynet.org/capture.html`. 24

[25] M. Carbonell, M. Mas, and G. Mayor. On a class of monotonic extended owa operators. *6th IEEE International Conference on Fuzzy Systems*, (III):1695–1700, 1997. 82

[26] CERT POLSKA. Home page of "ARAKIS". Available online at `http://www.arakis.pl`. 24, 174

[27] V. Chatzigiannakis, G. Androulidakis, K. Pelechrinis, S. Papavassiliou, and V. Maglaris. Data fusion algorithms for network anomaly detection: classification and evaluation. In *IEEE International Conference on Networking and Services, ICNS'07, Athens, Greece, June 2007*, June 2007. 32

[28] H. Chen, W. Chung, Y. Qin, M. Chau, J. J. Xu, G. Wang, R. Zheng, and H. Atabakhsh. Crime data mining: an overview and case studies. In *Proceedings of the 2003 annual national conference on Digital government research*, pages 1–5. Digital Government Society of North America, 2003. 31

[29] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *Proceedings of IEEE INFOCOM*, 2003. 105

[30] Z. Chen, C. Ji, and P. Barford. Spatial-temporal characteristics of internet malicious sources. In *Proceedings of INFOCOM*, 2008. 26, 124, 191

[31] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1953. 85, 87, 185

[32] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. D. Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM. 105, 188

[33] Computer Crime Research Center. Cybercrime is an organized and sophisticated business. Available online at `http://www.crime-research.org/news/20.02.2006/1835/`, Feb 2006. 15, 171

[34] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005 Workshop)*, Cambridge, MA, July 2005. 15, 27, 171, 174

[35] S. Corporation. Deepsight early warning services. 27

[36] M. Cova, C. Leita, O. Thonnard, A. D. Keromytis, and M. Dacier. Gone rogue: An analysis of rogue security software campaigns (invited paper). In *Proceedings of the 5th European Conference on Computer Network Defense (EC2ND)*, November 2009. 16, 19, 138, 158, 172, 194

[37] Crime-Research. Cyberwar: Russia vs estonia. Available online at `http://www.crime-research.org/articles/Cyberwar-Russia-vs-Estonia/`, May 2007. 16, 171

[38] Cyber-TA. Cyber-threat analytics (cyber-ta), sri international. 29

[39] M. Dacier, C. Leita, O. Thonnard, V.-H. Pham, and E. Kirda. *Assessing cybercrime through the eyes of the WOMBAT*. Chapter 3 of "Cyber Situational Awareness : Issues and Research", Springer International Series on Advances in Information Security, 2009. ISBN: 98-1-4419-0139-2, 11 2009. 19

[40] M. Dacier, V. Pham, and O. Thonnard. The WOMBAT Attack Attribution method: some results. In *5th International Conference on Information Systems Security (ICISS 2009), 14-18 December 2009, Kolkata, India*, Dec 2009. 19, 123, 191

[41] Darkreading. Botnets behind georgian attacks offer clues. Available online at http://www.darkreading.com/security/app-security/showArticle.jhtml?articleID=211201216, Sep 2008. 16, 171

[42] D. L. Davies and D. W. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227, January 1979. 58

[43] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006. 70

[44] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag. 30, 175

[45] DShield. Dshield distributed intrusion detection system. Available online at http://www.dshield.org. 26, 114, 174

[46] K. El Defrawy, M. Gjoka, and A. Markopoulou. Bottorrent: misusing bittorrent to launch ddos attacks. In *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association. 134

[47] EmergingThreats. Available online at http://www.emergingthreats.net. 26, 174

[48] Ertoz, Eilertson, Lazarevic, Tan, Kumar, Srivastava, and Dokas. MINDS - Minnesota Intrusion Detection System. In *Next Generation Data Mining, MIT Press, 2004*, 2004. 30, 175

[49] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis: Fourth Edition*. Hodder Arnold, 2001. 40, 45, 54

[50] A. Fei, G. Pei, R. Liu, and L. Zhang. Measurements on delay and hop-count of the Internet. In *in IEEE GLOBECOM'98 - Internet Mini-Conference*, 1998. 132

[51] J. Figueira, S. Greco, and M. E. Ehrgott. *Multiple Criteria Decision Analysis:State of the Art Surveys*. Springer, International Series in Operations Research & Management Science , Vol. 78, 2005. 32, 176

[52] D. Filev and R. R. Yager. On the issue of obtaining owa operator weights. *Fuzzy Sets Syst.*, 94(2):157–169, 1998. 83

[53] J. Fodor and M. Roubens. *Fuzzy Preference Modelling and Multicriteria Decision Support*. Kluwer, Dordrecht, 1994. 80

[54] J. Franklin, A. Perrig, V. Paxson, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 375–388, New York, NY, USA, 2007. ACM. 15, 171

[55] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. pages 31–, June-2 July 2004. 44

[56] R. Fullér and P. Majlender. An analytic approach for obtaining maximal entropy owa operator weights. *Fuzzy Sets and Systems*, 124(1):53–57, 2001. 81

[57] R. Fullér and P. Majlender. On obtaining minimal variability owa operator weights. *Fuzzy Sets and Systems*, 136(2):203–215, 2003. 82

[58] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89:445–456, 1996. 89, 185

[59] M. Grabisch. Alternative representations of discrete fuzzy measures for decision making. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 5(5):587–607, 1997. 89

[60] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets Syst.*, 92(2):167–189, 1997. 93

[61] M. Grabisch. The interaction and möbius representations of fuzzy measures on finites spaces, k-additive measures: a survey. *Fuzzy Measures and Integrals. Theory and Applications, M. Grabisch, T. Murofushi, and M. Sugeno (eds)*, 124(1):70–93, 2000. 95

[62] M. Grabisch and C. Labreuche. A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 2009. 32, 176

[63] M. Grabisch, T. Murofushi, M. Sugeno, and J. Kacprzyk. *Fuzzy Measures and Integrals. Theory and Applications*. Physica Verlag, Berlin, 2000. 32, 33, 88, 90, 176

[64] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium*, 2008. 27, 174

[65] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium*, August 2007. 29, 174

[66] J. D. Guyton and M. F. Schwartz. Locating nearby copies of replicated Internet servers. *SIGCOMM Comput. Commun. Rev.*, 25(4):288–298, 1995. 132

[67] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001. 57

[68] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2003. 52

[69] J. Hopfield and D. Tank. Neural computations of decisions in optimization problems. 52:141–152, 1985. 49

[70] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988. 37, 40, 177

[71] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999. 40

[72] M. Jakobsson and Z. Ramzan. *Crimeware: Understanding New Attacks and Defenses*. Addison-Wesley, 2008. 15, 171

[73] X. Jiang and D. Xu. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *Proceedings of the* 13^{th} *USENIX Security Symposium*, Aug. 2004. 24, 174

[74] W. P. Jones and G. W. Furnas. Pictures of relevance: a geometric analysis of similarity measures. *J. Am. Soc. Inf. Sci.*, 38(6):420–442, 1987. 45

[75] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002. 30, 175

[76] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990. 40, 58

[77] B. Krebs. Massive profits fueling rogue antivirus market. Washington Post, published online at `http://voices.washingtonpost.com/securityfix/2009/03/obscene_profits_fuel_rogue_ant.html`, March 2009. 16, 139, 172, 194

[78] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951. 43

[79] Lee, Wang, and Dagon, editors. *Botnet Detection: Countering the Largest Security Threat*, volume 36 of *Advances in Information Security*. Springer, 2008. 27, 124, 174, 191

[80] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 120–132, 1999. 30, 175

[81] W. Lee and S. J. Stolfo. Combining knowledge discovery and knowledge engineering to build IDSs. In *RAID '99: Proceedings of the 3th International Symposium on Recent Advances in Intrusion Detection*, 1999. 30, 175

[82] C. Leita. *SGNET : automated protocol learning for the observation of malicious threats*. PhD thesis, Institut Eurecom, 2008. 23, 24, 109

[83] C. Leita and M. Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008. 24, 27, 100, 109, 174, 202

[84] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005. 24, 100, 174

[85] C. Leita, V. H. Pham, O. Thonnard, E. Ramirez Silva, F. Pouget, E. Kirda, and M. Dacier. The Leurré.com Project: collecting internet threats information using a worldwide distributed honeynet. In *1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands*, Apr 2008. 24, 99, 100, 104, 174, 186

[86] Leurré.com Project. Home page of "Leurré.com Honeypot Project", 2009. 24, 99, 174, 186

[87] J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991. 44, 180

[88] J. Mao and A. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *Neural Networks, IEEE Transactions on*, 7(1):16–29, Jan 1996. 43

[89] J. Marichal. Tolerant or intolerant character of interacting criteria in aggregation by the choquet integral. *European Journal of Operational Research*, 155(3):771–791, 2004. 88, 94

[90] J.-L. Marichal and M. Roubens. Entropy of discrete fuzzy measures. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8(6):625–640, 2000. 88

[91] W. L. Martinez and A. R. Martinez. *Exploratory Data Analysis with MATLAB*. Chapman & Hall/CRC, 2004. 40, 54

[92] C. McCue. *Data Mining and Predictive Analysis: Intelligence Gathering and Crime Analysis*. Butterworth-Heinemann (Elsevier), May 2007, 2007. 31, 175

[93] J. Mena. *Investigative Data Mining for Security and Criminal Detection*. Butterworth-Heinemann (Elsevier,) Avril 2003, 2003. 31, 175

[94] MessageLabs. Messagelabs Intelligence: 2009 Annual Security Report. Available online at `http://www.messagelabs.com/intelligence.aspx`, Dec 2009. 15, 171

[95] Microsoft. TCP/IP and NetBT configuration parameters for Windows XP. Available online at `http://support.microsoft.com/kb/314053`, Nov 2006. 131

[96] Microsoft. TCP/IP and NetBT configuration parameters for Windows 2000 or Windows NT. Available online at `http://support.microsoft.com/kb/120642/en-us`, Oct 2007. 131

[97] P. V. Mieghem. Measurements of the hopcount in Internet. In *Poster session of PAM'01*, 2001. 132

[98] P. Miranda, M. Grabisch, and P. Gil. p-symmetric fuzzy measures. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(supplement):105–123, 2002. 94

[99] R. Mojena. Hierarchical grouping methods and stopping rules: an evaluation. *The Computer Journal*, 20(4):359–363, 1977. 54

[100] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security and Privacy*, 1(4):33–39, July 2003. 26

[101] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, 2006. 26

[102] D. Moore, C. Shannon, and k. claffy. Code-red: a case study on the spread and victims of an internet worm. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 273–284, New York, NY, USA, 2002. ACM. 26

[103] T. Murofushi and S. Soneda. Techniques for reading fuzzy measures (iii): Interaction index. In *Proceedings of the 9th Fuzzy Systems Symposium, Sapporo, Japan*, pages 693–696, 1993. 89

[104] Mwcollect. The Mwcollect Alliance. Available online at `http://alliance.mwcollect.org`. 27, 174

[105] N. Naoumov and K. Ross. Exploiting p2p systems for ddos attacks. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM. 134

[106] Y. Narukawa and V. Torra. Fuzzy measure and probability distributions: Distorted probabilities. *IEEE T. Fuzzy Systems*, 13(5):617–629, 2005. 93

[107] J. Nazario. Phoneyc: A virtual client honeypot. In *Proc. of Large-scale Exploits and Emerging Threats (LEET'09)*, Boston, MA, USA, 2009. 24

[108] NoAH. Home page of "Network of Affined Honeypots (NOAH)". Available online at `http://www.fp6-noah.org`. 24, 174

[109] Norman. Home page of "Norman Sandbox". Available online at `http://sandbox.norman.com`. 27, 174

[110] N.Provos. A virtual honeypot framework. In *Proceedings of the 13th USENIX Security Symposium*, 2004. 23, 174

[111] Offensive Computing. Home page of "Offensive Computing". Available online at `http://www.offensivecomputing.net`. 27, 174

[112] PandaLabs. The Business of Rogueware. Analysis of a new style of online fraud. PandaLabs Reports, available online at `http://www.pandasecurity.com/homeusers/security-info/tools/reports/`, July 2009. 16, 139, 172, 194

[113] PandaLabs. Profitability of rogue antimalware. PandaLabs Bulletins, available online at `http://www.pandasecurity.com/homeusers/security-info/tools/reports/`, 2009. 139, 194

[114] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet Background Radiation. In *Proceedings of the 4th ACM SIGCOMM conference on the Internet Measurement*, 2004. 26, 101, 124, 191

[115] M. Pavan. *A New Graph-Theoretic Approach to Clustering, with Applications to Computer Vision*. PhD thesis, Università di Bologna, 2004. 50

[116] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003. 25, 47, 48, 49, 180, 181

[117] V.-H. Pham. *Honeypot traces forensics by means of attack event identification*. PhD thesis, TELECOM ParisTech, 2009. 23, 76, 104, 106, 127, 187

[118] V.-H. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008. 101, 104

[119] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an Emulator for Fingerprinting Zero-Day Attacks. In *Proc.of the 1st ACM SIGOPS EUROSYS*, Leuven. Belgium, April 2006. 24, 174

[120] F. Pouget. *Distributed System of Honeypots Sensors: Discrimination and Correlative Analysis of Attack Processes*. PhD thesis, Institut Eurecom, 2006. 23, 24, 25, 47, 174, 178

[121] F. Pouget, M. Dacier, and H. Debar. Honeypot-based forensics. In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference 2004*, Brisbane, Australia, May 2004. 100, 186

[122] F. Pouget, M. Dacier, and V. H. Pham. Leurre.com: on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference, 29-30th March 2005, Monaco*, Mar 2005. 100, 186

[123] N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004. 100

[124] A. R. Puniyani, R. M. Lukose, and B. A. Huberman. Intentional walks on scale free small worlds, 2001. 131

[125] V. V. Raghavan and C. T. Yu. A comparison of the stability characteristics of some graph theoretic clustering methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-3(4):393–402, July 1981. 47, 180

[126] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM. 15, 27, 171, 174

[127] V. Reding. Enhanced information security in software and services. What role for government, security providers and users? (speech). European Information Security Awareness Day, available online at `http://ec.europa.eu/commission_barroso/reding/docs/speeches/security_20070227.pdf`, February 2007. 15, 171

[128] J. Riordan, D. Zamboni, and Y. Duponchel. Building and deploying Billy Goat, a worm-detection system. In *Proceedings of the 18th Annual FIRST Conference, 2006*, 2006. 23, 174

[129] J. Rocaspana. Shelia: A client honeypot for client-side attack detection. Available online at `http://www.cs.vu.nl/~herbertb/misc/shelia/`, 2009. 24

[130] G.-C. Rota. On the foundations of combinatorial theory I. Theory of Möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2:340–368, 1964. (MR 30#4688). 86

[131] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987. 58

[132] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 59, Washington, DC, USA, 1998. IEEE Computer Society. 44

[133] L. Rüschendorf. The Wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, March 1985. 44

[134] J. M. F. Salido and S. Murakami. Extending yager's orness concept for the owa aggregators to other mean operators. *Fuzzy Sets and Systems*, 139(3):515–542, 2003. 80

[135] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. In *In Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295–306, 2000. 22

[136] C. Schiller and J. Binkley. *Botnets: The Killer Web Applications*. Syngress Publishing, 2007. 124, 191

[137] B. Schneier. Organized cybercrime. Available online at `http://www.schneier.com/blog/archives/2006/09/organized_cyber.html`, Sep 2006. 15, 171

[138] J. P. Scott. *Social Network Analysis: A Handbook*. Sage Publications Ltd; 2nd edition (March 25, 2000), 2000. 31

[139] C. Seifert, I. Welch, and P. Komisarczuk. HoneyC - The low-interaction client honeypot, 2006. 24

[140] G. Shafer. *A mathematical theory of evidence*. Princeton university press, 1976. 33

[141] C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004. 26

[142] L. Shapley. A value for n-person games. In H. Kuhn and A. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, Princeton, NJ, 1953. 89

[143] R. N. Shepard. Multidimensional scaling, tree fitting, and clustering. *Science*, 210:390–398, 1980. 45, 111, 197

[144] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 47, 180

[145] S.Jajodia, P.Liu, V.Swarup, and C.Wang, editors. *Cyber Situational Awareness: Issues and Research*, volume 46 of *Advances in Information Security*. Springer, Nov 2009. 29, 175

[146] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet ip traceback. *IEEE/ACM Trans. Netw.*, 10(6):721–734, 2002. 22

[147] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings IEEE Infocomm 2001*, 2001. 22

[148] Sophos threat analysis. W32/Allaple-B. Available at http://www.sophos.com/security/analyses/viruses-and-spyware/w32allapleb.html. 133

[149] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley, 2002. 23, 99

[150] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The top speed of flash worms. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 33–42, New York, NY, USA, 2004. ACM. 26

[151] J. Stewart. Top Spam Botnets Exposed. Malware Research, SecureWorks, published online at http://www.secureworks.com/research/threats/topbotnets/, April 2008. 15, 171

[152] B. Stone-Gross, C. Kruegel, K. C. Almeroth, A. Moser, and E. Kirda. FIRE: FInding Rogue nEtworks. In *Twenty-Fifth Annual Computer Security Applications Conference (ACSAC 2009), Honolulu, Hawaii, 7-11 December 2009*, pages 231–240, 2009. 16

[153] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, 1974. 85, 92, 185

[154] Sunbelt. Home page of "CWSandbox". Available online at http://www.sunbeltsoftware.com. 27, 174

[155] Support Intelligence. Home page of "Support-Intelligence". Available online at http://www.support-intelligence.com. 27

[156] Symantec. W32.Rahack.W. Available at http://www.symantec.com/security_response/writeup.jsp?docid=2007-011509-2103-99. 133

[157] Symantec Corporation. Symantec Internet Security Threat Report. Available online at http://www.symantec.com/business/theme.jsp?themeid=threatreport. 15, 138, 171

[158] Symantec Corporation. Symantec Report on the Underground Economy. Available online at http://www.symantec.com/business/theme.jsp?themeid=threatreport, November 2008. 15, 171

[159] Symantec Corporation. Symantec Report on Rogue Security Software. Available online at http://www.symantec.com/business/theme.jsp?themeid=threatreport, October 2009. 16, 19, 138, 139, 158, 172, 193, 194

[160] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005. 40, 43

[161] Technology Research News. Internet stays small world. Available online at http://www.trnmag.com/Stories/2001/091201/Internet_stays_small_world_091201.html, Dec 2001. 131

[162] The Honeynet Project. Home page of the "Honeynet Project". Available online http://www.honeynet.org. 24

[163] The Shadowserver Foundation. Available online at http://www.shadowserver.org. 27, 174

[164] The Team Cymru. Home page of "The Team Cymru darknet" project. Available online at http://www.cymru.com/Darknet. 25, 114, 174

[165] O. Thonnard and M. Dacier. A Framework for Attack Patterns' Discovery in Honeynet Data. *Journal of Digital Investigation*, 5S:S128–S139, 2008. 19

[166] O. Thonnard and M. Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008. 19

[167] O. Thonnard, W. Mees, and M. Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *KDD'09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th - July 1st, 2009, Paris, France*, Dec 2009. 19, 123, 191

[168] O. Thonnard, W. Mees, and M. Dacier. Behavioral Analysis of Zombie Armies. In C. Czossek and K. Geers, editors, *The Virtual Battlefield: Perspectives on Cyber Warfare*, volume 3 of *Cryptology and Information Security Series*, pages 191–210, Amsterdam, The Netherlands, 2009. IOS Press. 16, 19, 123, 172, 191

[169] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the Gap statistic. *Journal of the Royal Statistical Society*, 63:411–423, 2001. 55, 56

[170] V. Torra. Weighted owa operators for synthesis of information. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 966–971 vol.2, Sep 1996. 74

[171] V. Torra. The weighted OWA operator. *Int. Journal of Intelligent Systems*, 12(2):153–166, 1997. 74, 75, 183

[172] V. Torra. The WOWA operator and the interpolation function W*: Chen and otto's interpolation method revisited. *Fuzzy Sets Syst.*, 113(3):389–396, 2000. 75

[173] V. Torra and Y. Narukawa. *Modeling Decisions: Information Fusion and Aggregation Operators*. Springer, Berlin, 2007. 32, 80, 88, 176

[174] J. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. 39

[175] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November 2008. 51, 52, 115, 152, 189

[176] N. Vanderavero, X. Brouckaert, O. Bonaventure, and B. Charlier. The HoneyTank: a scalable approach to collect malicious internet traffic. In *Proceedings of the International Infrastructure Survivability Workshop (IISW'04)*, Dec. 2004. 24, 174

[177] Virustotal. Available online at `http://www.virustotal.com`. 27, 174

[178] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. Snoeren, G. Voelker, and S. Savage. Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm. In *Proceedings of the ACM Symposium on Operating System Principles (SOSP), Brighton, UK*, Oct. 2005. 24, 174

[179] X. Wang, X. Wang, D. S. Reeves, D. S. Reeves, S. F. Wu, S. F. Wu, J. Yuill, and J. Yuill. Sleepy watermark tracing: An active network-based intrusion response framework. In *in Proc. of the 16th International Information Security Conference*, pages 369–384, 2001. 22

[180] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proc. Network and Distributed System Security (NDSS)*, San Diego, CA, February 2006. 24

[181] Z. Wang and G. Klir. *Fuzzy Measure Theory*. Plenum Press, New York, 1992. 33

[182] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press; 1 edition (November 25, 1994), 1994. 31

[183] T. Werner. Honeytrap. Available online at `http://honeytrap.carnivore.it/`. 23, 174

[184] C. Westphal. *Data Mining for Intelligence, Fraud & Criminal Detection: Advanced Analytics & Information Sharing Technologies*. CRC Press, 1st edition (December 22, 2008), 2008. 31, 175, 176

[185] K. J. Wheaton. Top 5 intelligence analysis methods, http://sourcesandmethods.blogspot.com, [sep 2009]. 32, 176

[186] D. Wheeler and G. Larsen. Techniques for Cyber Attack Attribution. *Institute for Defense Analyses, Oct 2003*, 2008. 21, 22

[187] WOMBAT Project. Worldwide Observatory of Malicious Behaviors and Attack Threats. deliverable D03 (D2.2). Analysis of the state-of-the-art. Available online at `http://www.wombat-project.eu`, November 2008. 23

[188] WOMBAT Project. Worldwide Observatory of Malicious Behaviors and Attack Threats. deliverable D07 (D3.2). Design and prototypes of new sensors. Available online at `http://www.wombat-project.eu`, December 2008. 140, 194

[189] WOMBAT Project. Worldwide Observatory of Malicious Behaviors and Attack Threats. deliverable D13 (D3.3). Sensor deployment. Available online at `http://www.wombat-project.eu`, December 2009. 140, 194

[190] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1101–1113, 1993. 47, 180

[191] Z. Xu. An overview of methods for determining owa weights: Research articles. *Int. J. Intell. Syst.*, 20(8):843–865, 2005. 83

[192] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision-making. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988. 32, 72, 73, 80, 176, 183

[193] R. Yager. Connectices and quantifiers in fuzzy sets. *Fuzzy Sets and Systems*, 40:39–75, 1991. 83

[194] R. Yager. Quantifier guided aggregation using owa operators. *Intelligent Systems*, 11:49–73, 1991. 83

[195] V. Yegneswaran, P. Barford, and U. Johannes. Internet intrusions: global characteristics and prevalence. In *SIGMETRICS*, pages 138–147, 2003. 26, 124, 174, 191

[196] V. Yegneswaran, P. Barford, and V. Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005. 28, 127, 175

[197] K. P. Yoon and C.-L. Hwang. *Multiple Attribute Decision Making: An Introduction.* SAGE Publications, Quantitative Applications in the Social Sciences, 1995. 32, 176

[198] Y. C. Z. Li, A. Goyal and V. Paxson. Automating analysis of large-scale botnet probing events. In *Proc. of ASIACCS*, March 2009. 28

[199] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, 1971. 47, 180