Institut Eurecom
Corporate Communications Department
2229, route des Crêtes
B.P. 193

06904 Sophia Antipolis
FRANCE

Research Report RR-03-093

White Paper:

Alert Correlation: Review of the state of the art[1]

November 28, 2003

Fabien Pouget, Marc Dacier

Institut Eurecom
Email: {pouget,dacier@eurecom.fr}

[1] This research is supported by a research contract with France Telecom R&D, contract N. 425-17044

Table of contents

White Paper:
“Alert Correlation: Review of the state of the art[2]”

Fabien Pouget, Marc Dacier
Institut Eurécom
Email: {pouget,dacier@eurecom.fr}

Institut Eurécom
2229, Route des Crêtes ; BP 193
06904 Sophia Antipolis Cedex ; France

Abstract

The purpose of this document is to offer a review of the state of the art concerning the emerging field of so-called «alert correlation». Despite the fact that several recent publications seem to present this domain as a new one, we will show the close connections that exist with another well established one, namely network management and its event correlation approaches. We try to highlight the core notions embedded within the term “correlation” thanks to the definition of several building blocks used to design “correlation engines”. We focus on the techniques used within the intrusion detection domain and present a survey not only of papers published in that field but also of currently available tools. We show the gap that exists, as of today, between sophisticated techniques presented in research papers and actual implementations that are readily available.

1 Introduction

Several reasons explain the increasing number of research projects, tools and products addressing the issue of alert correlation, within the intrusion detection field. First of all, current sensors are relatively verbose. For a given attack, or anomalous phenomenon (such as a port scan), they can generate a possibly large amount of alerts.

[2] This research is supported by a research contract with France Telecom R&D, contract N. 425-17044

This large volume of alerts is quite likely to overwhelm the operators in charge of looking at suspicious events. The situation gets even worse when several sensors are used to monitor the same systems from a different viewpoint. In that case, a given attack will generate several threads of alerts from various sensors, but not necessarily from all of them. Furthermore, it is now well agreed upon within the ID community that most of these alerts represent false alarms [Julisch00]. Therefore, there is a need for techniques that could automatically recognize and discard all such false alarms in order to minimize the cost of dealing with the real alarms. Last but not least, experience shows that the interpretation of the alerts usually requires more than the sole messages provided by the sensors. As a consequence, there is a need for techniques that can analyze the alerts within the context in which they have been generated. This might require the ability to correlate them with some other, external, contextual information provided by means of other devices than the sole intrusion detection sensors.

As we see, there are several open issues that need to be addressed in the "alert correlation" field. In this document, we provide a survey of all the proposed techniques, not only by the ID community but also by the network management community which has been facing a similar problem for a longer period, namely the correlation of events arriving from all network devices in the presence of system failures. We also look at the various tools and products that have been proposed so far and we show the existing gap that exists between sophisticated research approaches and rather down-to-earth, pragmatic, techniques used in most implementations.

This paper is organized as follows. Section 2 presents the notion of "correlation", and more precisely, of "alert correlation". We provide a detailed analysis of the basic functions that characterize alert correlation and we show that complex correlation operations can be broken into a few fundamental building blocks. Section 3 offers a survey of the methods proposed, mostly, in research papers. Section 4, on the other hand, looks at existing tools and products that are readily available. Section 5 concludes this white paper.

2 Definitions

2.1 General Correlation definition

The term "*correlation*" is relatively vague and has been used in many different ways. *Correlation* stems from the Medieval Latin and is composed of two Latin roots: *cum*, which means 'with' (relationship) and *relation* from *relatus*, the past participle of *referre*, which means 'to carry back'.

***Correlation* can be defined as an "action to carry back relations with each other".**

This word is particularly used to express a relationship between variables. Thus, *correlation* can be a statistic relationship between at least two variables such that systematic changes in the value of one of them induce changes in the others. In other words, *correlation* permits to express how closely variables co-vary. It can be represented by a numerical value from -1 (perfect negative correlation) through 0 (no correlation) to +1 (perfect positive correlation).

Correlation has been used in several domains, to denote the various treatments to be applied on large input sets, potentially created by a small amount of common causes. This is the case in areas such as:

- event correlation
- alert correlation
- alarm correlation
- attack correlation

Event Correlation is a widely accepted technology for managing the complexity of modern telecommunication and data networks. It has been used for various network management tasks, but the majority of its applications have been for network fault management. Different elements of a given system can emit numerous signals, also called events (simple SNMP traps for instance), that are sent to the administrator in order to provide him information on the current system status. For instance, specific correlation applications have been developed to manage switched, SS7, wireless, ATM, SONET, IP and other networking devices and environments. The objective consists in carrying back *essential* information to the administrator from the multitude of events.

Alarm correlation is used in Network Management but in Security also. The principle remains similar. Alarms are external manifestations of faults, where a fault is a disorder occurring in an element of the managed network. *Alarm correlation* is defined by Jakobson et al. as a conceptual interpretation of multiple alarms such that new meanings are assigned to these alarms [Jakob93, JakWeBre]. It may be used for network fault isolation and diagnosis, selective corrective actions, proactive maintenance, and trend analysis.

The difference between *alarm correlation* and *alert correlation* is so tight that we intend to use only one expression instead of these two.

Attack correlation is not so common. It is used in a very specific situation: some security experts try to model attacks by building some scenarios. The process which aims at building these scenarios from primary attacks models is called attack correlation. We refer the interested reader to Section 3.3.2 for more information on this notion.

One first remark: these expressions are conceptually identical. Based on our readings and our experience, we find that concepts hidden behind those expressions are very similar. Consequently, we decide to readapt Jacobson’s definition and to extend it to all ‘event’, ‘alert’ and ‘attack’ inputs. This leads to the following correlation interpretation [Jakob93]:

“Alerts [alarms/attacks/events] correlation is a conceptual interpretation of multiple alerts [alarms/attacks/events], such that a new meaning is assigned to these alerts [alarms/attacks/events]”.

2.2 Alert Correlation

2.2.1 The needs

We call **alert,** a message sent by any component of an intrusion detection system. Thus, an alert results from the detection of a suspicious or a malicious action. Moreover, it typically contains information about the unusual activity that was detected, as well as the specifics of the occurrence.

The diversity in the elements of any network presents a significant security and management challenge. Each security device or application, such as firewall, intrusion detection, anti-virus, VPN, or specialized device, has its own console that is provided by a particular vendor. Mission critical applications are running on servers whose operating systems vary, and the landscape is further diversified by the new generation of small and powerful network and security appliances and their need to be monitored. There is limited integration within each class of device, and virtually no integration among device types, making simpler and more efficient event and alert management at some higher level difficult.

The main challenge of a security management scheme is to converge alerts from these multiple alert data sources, so that relationships or patterns can be determined among seemingly disparate alerts from the network [Open01]. Armed with these patterns and profiles, security administrators can develop security alert response strategies and plan some proactive measures for preventing alerts from occurring in the future. The enabling technique that transforms all of these alerts into more exploitable information is called *alert correlation*.

2.2.2 Correlation specifications

Many approaches offer to transform alerts in order to extract more exploitable information. Some of them are based on simple alerts. For instance, Carey et al. suggest in [Carey02] a simple approach based upon IDMEF alerts (see 2.2.3 for more details) generated by different Intrusion Detection Systems (IDSs). It relies on signatures of some alert patterns. When a signature match occurs, a synthetic alert is generated (which may, in turn, be used by another signature if another event is detected).

On the other hand, we can find more complex approaches. For instance, M2D2 (a Formal data Model for IDS Correlation) by Morin et al. distinguishes in [Morin02] four information types:

- the monitoring system

- its vulnerabilities
- its security monitoring tools
- alerts and observed events

Thus, alert correlation is based upon a combination of all these four information types.

The correlation process can take various types of data as input. Moreover, some approaches are applied in ‘real-time’, in parallel with incoming alerts, while others are launched *a posteriori*, when all alerts are logged in a given file.

This leads to the following observation: correlation appears with various degrees of complexity, reactivity, specificities, etc…

Alert correlation itself covers a large and diverse domain. In this document, we propose to characterize the *correlation* process by means of the following features:

- Its inputs (Section 2.1.3).
- Its type (Section 2.2.4).
- Its objectives (Section 2.3)

2.2.3 Input variety

Table 1: Examples of specific attack languages (non exhaustive list)

	BSM [SunMic]	Tcpdump [Jacob00]	Bishop [Bish95]	CASL [Casl98]	NASL [Derai99]	CISL [Cisl99]	IDMEF [Idmef]	Kumar [Kumar95]	BRO [Paxs98]	Snort [Roes99]	STATL [EckVig00]	Lambda [CupOrt00]	ADeLe [Mé00]
Event languages	X	X	X										
Exploit languages				X	X							X	X
Reporting languages						X	X						
Detection languages								X	X	X	X	X	X
Correlation languages												X	X
Response languages													

To describe an attack completely, several specific languages have been designed, with different purposes. Vigna & Al. propose in [Vigna00] six different classes of such *attack languages*: event, exploit, detection, correlation, reporting and response. Event languages describe the format of events used during the detection process [Jacob00, Bish95]. Exploit languages are used to describe the stages to be followed to perform an intrusion [Casl98, CupOrt00, Derai99]. Detection languages allow the expression of the manifestation of an attack in terms of occurrences of events [Paxs98, EckVig00, Roes99, Mé98]. Correlation languages permit analysis of alerts provided by several IDSs in order to generate meta-alerts [CupOrt00]. Reporting languages describe the format of alerts produced by the IDSs [Idmef, Cisl99]. Finally, response languages are used to express countermeasures to be taken after detection of an attack. Table 1 is extracted from [Mé00] and provides some examples of attack languages.

As briefly explained in 2.2.2, correlation processes differ greatly according to what should be correlated. Inputs may have different aspects and we group them into three distinct categories:

☑ Row alert input:

Alerts which are directly obtained from Intrusion Detection sensors or other alerts transmitters.

These alerts are written in a specific format, sometimes even using a specific language such as the so-called *reporting language* by Vigna et Al [Vigna00].

The lack of interoperability between Intrusion Detection Systems (IDSs) has been recognized and in 1997, the US government's Defense Advanced Research Projects Agency (DARPA) initiated the Common Intrusion Detection Framework (CIDF) research project [All00, Kot02]. CIDF contains the Common Intrusion Specification Language (CISL), which is a Lisp-like language that is used to represent intrusion data and communicate this data between IDS components. More precisely, this language is based on the concept of generalized Intrusion Detection Objects, called GIDOs, which, in turn, are specified as S-expressions. We report the interested reader to [Rivest] for more details on the S-expressions (SEXP) data structure. CISL is quite expressive and vendors were invited to participate in the CIDF project, however they never showed much interest in it [Cisl99]. Development of the project ceased in 1999.

A more interesting *reporting format* is the Intrusion Detection Message Exchange Format, also called IDMEF. IDMEF is a product of the Intrusion Detection Working Group (IDWG) [Idmef]. This group has been created as a follow up to the CIDF one in order to reach a larger audience. The purpose of IDMEF is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems, and to the management systems which may need to interact with them. Communication protocols have also been considered within that working group, resulting in the creation of IDXP [Idxp]. The Internet-Draft [Idmef] describes a data model to represent information exported by intrusion detection systems, and explains the rationales for using this model. An implementation of the data model in the Extensible Markup

Language (XML) is presented, an XML Document Type Definition is developed and examples are provided.

The Intrusion Detection Message Exchange Format (IDMEF), and its transport mechanism, the Intrusion Detection Exchange Protocol (IDXP), are generally considered a useful first step. However, they are sometimes criticized because of the use of XML DTDs, which lack the ability to adequately represent inheritance. Undercoffer and Pinkston give a critique of IDMEF in [Underc02].

This being said, IDMEF remains currently the only advanced *reporting format* standard. Of course, there are many other approaches that permit to map *various alert inputs* into a common format but they are proprietary solutions and they are often limited to a given set of products.

☑ Meta-alerts:

All the correlation process inputs that have a higher expression level than native alerts.

This implies a preliminary task to translate *raw alert input* into a richer *meta-alert*. One example is the P. Ning *hyper-alert* concept [Ning02]. Alerts are enriched by complementary information which is named *prerequisites* and *consequences*. A more detailed description of hyper-alerts is given in section.

Another example comes from the ASAX project (Advanced Security Audit-trail Analysis on uniX). B. Le Charlier et al. introduce in [Asax92] a normalized audit file format (NADF) which is flexible enough to translate various audit file formats into a common way and enrich them with some contextual information when needed. The alert trail analysis, including correlation processes, is performed on normalized trails only. The meta-alerts would be these normalized trails (the ASAX architecture is given in 3.5.2).

One more example is the *fusion alert* concept of F. Cuppens et al. described in [Cuppens02] for the French MIRADOR and Dico Projects. The simple alerts generated by different IDSs detecting a same attack are merged into a single cluster. This is called *fusion process*. It generates a *fusion alert* (a meta-alert in our own terminology) that is treated by the *correlation process*. The correlation process receives the fusion alerts and tries to correlate them one by one using correlation rules. The *correlation process* itself is discussed in 3.3.2.

EMERALD, an acronym for "Event Monitoring Enabling Responses to Anomalous Live Disturbances" has a similar approach [ValdSkin]. It uses an extension of the IDWG IDMEF to introduce the concept of "alert thread". An EMERALD alert thread consists in alerts that come from the same sensor and are related to the same attack. A second concept that is used is that of *meta-alert*. A meta-alert is composed in EMERALD of one or more alerts that may originate from multiple heterogeneous sensors. The correlation process of EMERALD is then based on these meta-alerts.

☑ Other inputs:

This group contains all correlation inputs that do not belong to the two previous categories.

One classic example is the case of *heartbeats.* Analyzers use them as messages to indicate their current status to managers. They are defined in the IDMEF draft but they can be simple SNMP traps that are periodically sent to the manager. They are not alerts by themselves. However, they participate to the correlation process and are identified as another kind of input. It may also be contextual information such as names of users, configuration information, vulnerability reports (e.g. Nessus), etc.

2.2.4 Correlation specificities

Many whitepapers about correlation exist. Each of them presents new techniques to achieve particular objectives. We explain in Section 2.3 that all these correlation processes can be summarized by a few elementary operations. But without going any further, we notice that all of them have specificities, which allow us to make some groups according to the following criteria:

- *Offline vs. online*: some tools launch correlation processes dynamically, while inputs are still coming into the system. Others launch correlation processes once data are stored, with all the inputs available at one time. In the first case, administrators await real-time reports, which enable fast reactivity. In the second case, administrators extract information *a* *posteriori*, in order to have more global information. This is typically done for *post-mortem* analysis.

- *Stateless vs. Stateful*: Alert correlation approaches may be further classified as stateful [Houck95, Nyg95] or stateless [Katz95, Yemi96]. Stateless systems typically are only able to correlate alerts independently, without any time or context constraints. State-based systems support the correlation of alerts in an alert-driven fashion at the expense of the additional overhead associated with maintaining the system state. The stateless characteristic might seem contradictory with the *correlation* definition (action to report relation with others).

- *Lossless vs. Enrichment:* The output of some correlation processes 'may not contain all initial inputs that have created it. Loss of information might occur. On the contrary, other tools enrich initial inputs by adding new information.

- *Context awareness and Context independence:* Correlation may rely on some information sources which are peculiar to the context. A well-known example consists in using the network topology to help correlating the alerts. Some tools can be placed in various contexts, without further context-knowledge installation, while others require a longer and more fastidious installation to learn the operational context. Thus, we first distinguish correlation approaches that depend on the context from the others. Then, we identify those which acquire this context knowledge automatically from those which need a human/expert intervention.

2.2.5 Anti-correlation notion

This notion is sometimes used in literature. An *anti-correlation* process consists in incorporating during the correlation process some information to reduce correlation complexity without loosing accuracy of prediction. Actually, anti-correlation is a particular form of correlation where the kind of dependence it introduces is opposed to the one correlation is looking for. We sometimes see terms '*negative dependence*' vs. '*positive dependence'* to express this duality.

For instance, this mechanism can be useful to detect that an intrusion disables a given attack scenario in progress. It is used this way in [Cuppens02]. L. Langfeldt explains in [Lang02, section 2.2] the notion of dual rules, which contribute to build an anti-correlation system in an evaluation context (the evaluation is based on rules). He distinguishes two sets of rules. This double set of rules pictures an ambiguous situation: "there are two divergent sets of rules for 'good' evaluations" [Lang02]. Thus, they bring out the incompatible requirements confronting an evaluator.

2.3 Basic correlation operations

2.3.1 Presentation

Several types of correlation may be identified [JakWeis95], according to the operations executed on the alerts. Correlation can be seen as black box, with some inputs, and with some objectives to reach. Techniques used inside the black box are complex and we give in Section 3 a list of the most current ones. However, correlation can be characterized in a simpler way by observing its different aspects. We notice that a large number of correlation processes can be 'broken up'' into a few fundamental bricks, or essential operations. We discern seven of them:

- compression
- filtering
- selective suppression
- thresholding
- modification
- generalization
- specialization
- enrichment

They are presented in the following paragraphs.

2.3.2 Compression

Compression (or "compaction") consists of detecting, from the observation of the alerts received in a given time window, multiple occurrences of the same alert, and substituting the corresponding alerts, possibly indicating how many times the alert occurred during the observation period. It aims at saving storage space or transmission time. There are many compression methods, but they often induce some information losses. If the compression is lossless, we use the term ***aggregation*** instead. One example is illustrated in figure 1. During a given time window [t_1, t_2], five alerts are received, three of type A, and two of type B. The *compression* process emits at t_2 one single alert that compresses the information received during this time window: three alerts of type A and two alerts of type B. However, the generated alert may not contain information such as arrival time of each alert, that was initially known before compression.

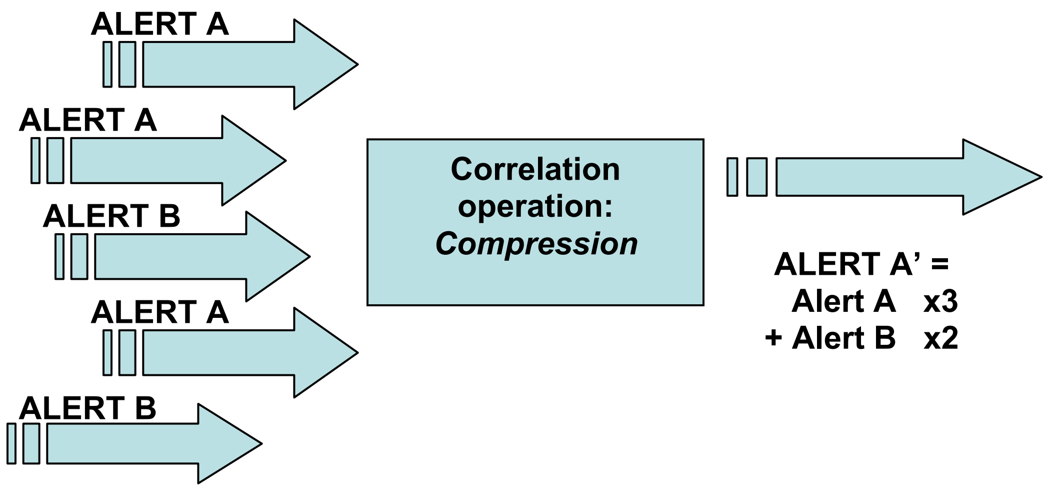


Figure 1: correlation operators, *Compression*

2.3.3 Filtering

Filtering consists of suppressing a given alert or set of alerts, depending on the values of certain parameters/attributes on the alert(s) to be discarded. Thus, filtering only takes into account the attributes of the alert which is being filtered. This stateless operation leads to many losses. One example is illustrated in figure 2. From three different alerts, only A3 is sent to the administrator as its attribute A matches the filtering rule.

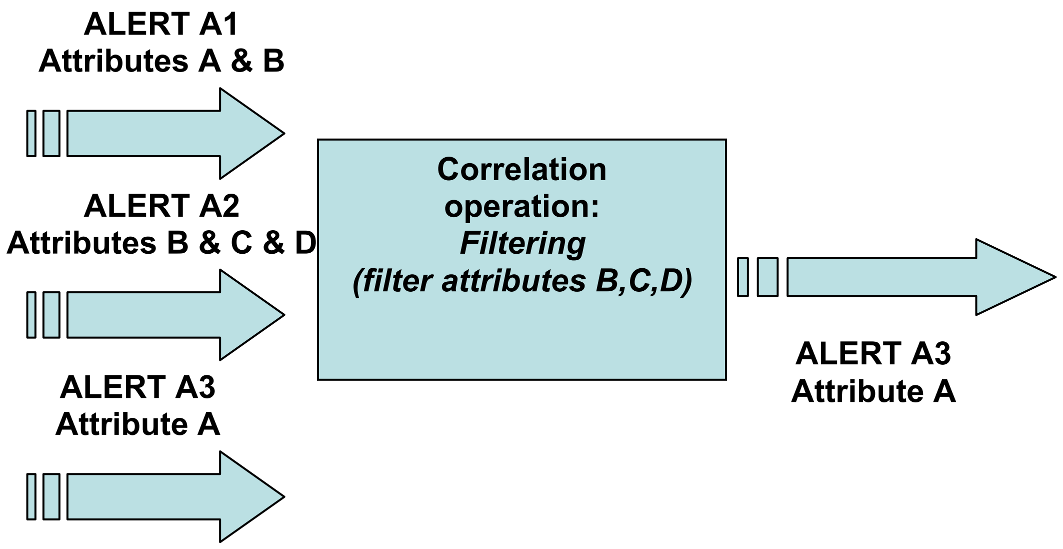


Figure 2: correlation operators, *Filtering*

2.3.4 Selective Suppression

Selective Suppression is the discarding of an alert or a set of alerts, according to criteria continuously evaluated by the correlation system and related to the dynamic context of the alert management process. The suppression criteria are generally linked to the presence of other alerts, to the temporal relationship among alerts or to the priorities established by the administrators. This stateful approach may lead to losses as illustrated in figure 3. A rule specifies that alert A2 should be discarded if received less than τ seconds after alert A1.

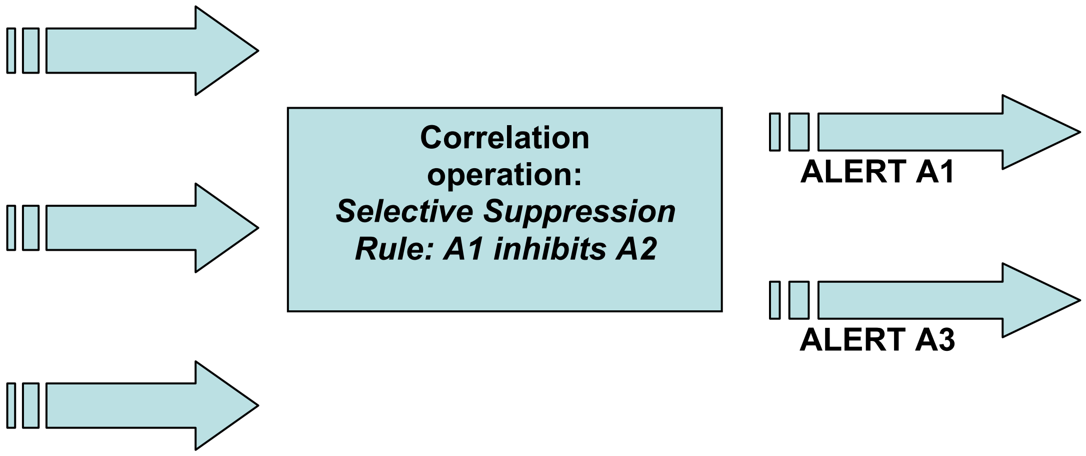


Figure 3: correlation operators, *Selective Suppression*

2.3.5 Thresholding

Thresholding consists in generating a new alert each time the number of occurrences of a given type of alerts reaches a predefined threshold within a certain time window. This approach looses information as alerts that do not reach the threshold are discarded by the correlation process. As illustrated in figure 4, Alert A1, A2 and A3 trigger the emission of alert A', as the given threshold of 3 is reached.

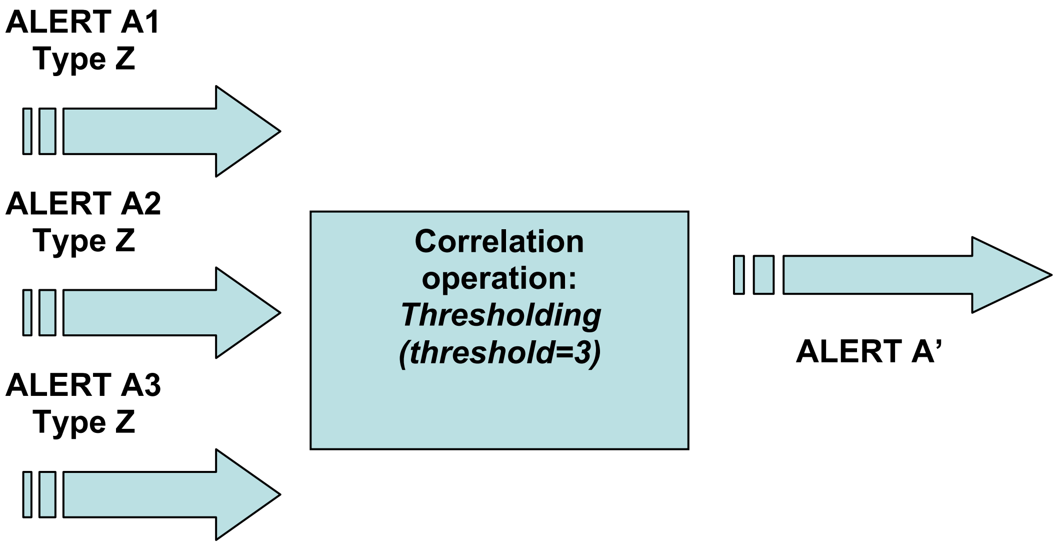


Figure 4: correlation operators, *Thresholding*

2.3.6 Modification

Modification is an operation in which, depending on the operational context, an alert is replaced by another one. This stateless operation can be lossless if the modification is reversible. It is illustrated in figure 5, where Alert A's severity is changed.

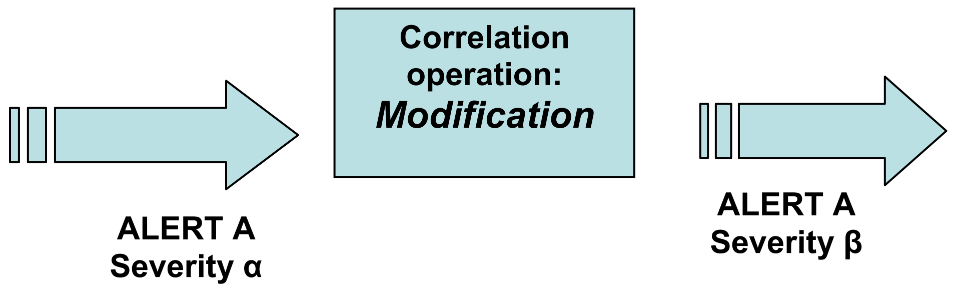


Figure 5: correlation operators, *Modification*

2.3.7 Generalization

Generalization consists in replacing an alert or a set of alerts, depending on the operational context, by the alert corresponding to its *super-class* [Bap94]. This is a stateful operation based on inductive type reasoning, which replaces one or more alerts by another informational alert. The meaning of the new alert is an induced generalization of received alerts. *Generalization* is illustrated in figure 6. Alerts A and B both belong to the DNS server attack category. Instead of transferring the two specific alert types, the alert class only is sent to the administrator. This implies some hierarchies among alerts to determine the so-called *super-classes*, and some information is lost during the *generalization* phase (the type in our example).

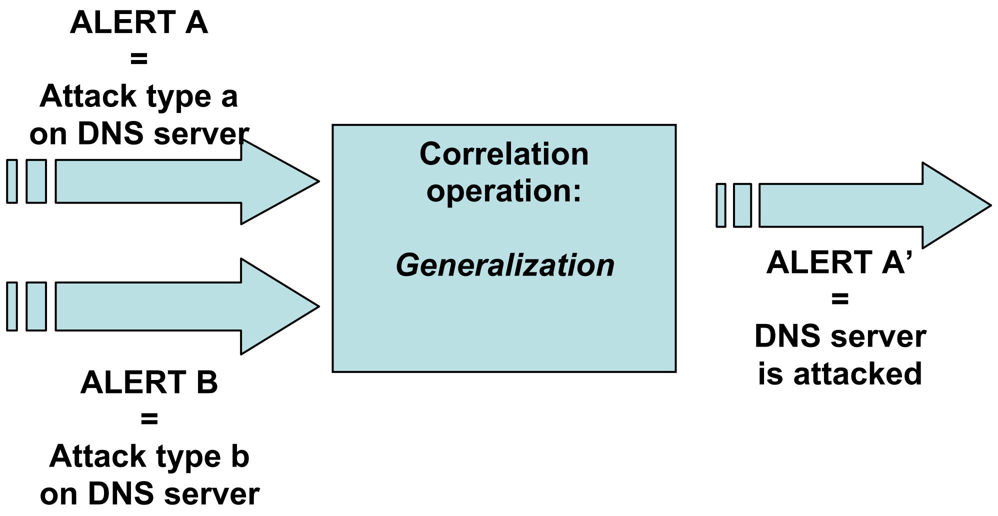


Figure 6: correlation operators, *Generalization*

2.3.8 Enrichment

Enrichment is an operation that consists in substituting an alert for another, corresponding to a *sub-class* [Bap94]. This implies some hierarchies among alerts, in order to determine the so-called *sub-classes*. However, unlike the *generalization* approach, there is no loss of information. One example is given in figure 7. The first incoming alert is alert A, which signals a successful scan on a well-known Trojan port. Attribute *a* gives the IP address of the target machine, and attribute *b* provides the alert type (successful tcp scan on port X). Alert A' keeps the same attributes, but contains also attribute *c* which gives the list of possible resulting attacks (CERT attack classifications for example), and attribute *d* which contains the number of similar scans alerts already stored in the database.

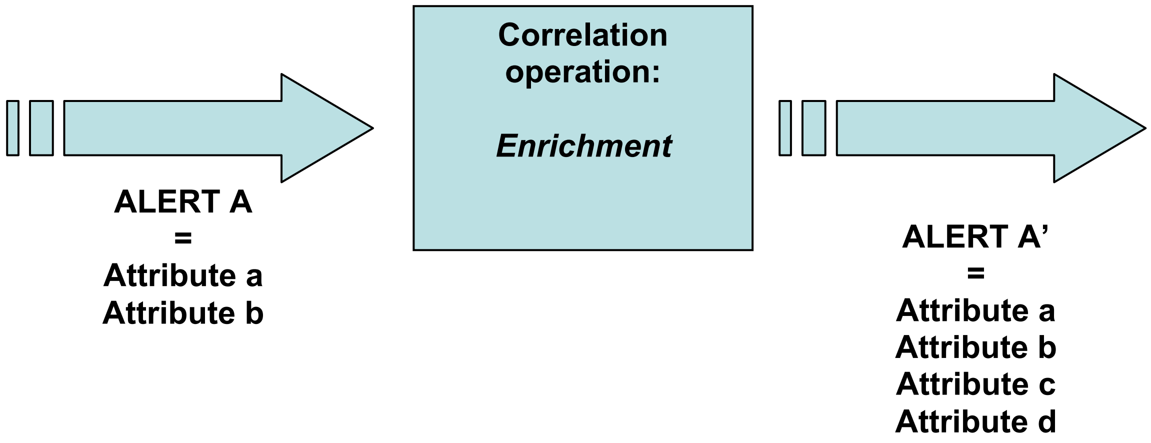


Figure 7: correlation operators, *Enrichment*

2.3.9 Specialization

Specialization is an operation which is the reverse of generalization. This operation, based on deductive type reasoning, does not add any new information besides the ones that were already implicitly present in the original alerts and in the configuration database, but it is useful in making evident the consequences that an alert in a given layer may cause in a higher layer. From a given alert, the correlation process generates multiple alerts, which should be more specialized to the administrator. One example is illustrated on figure 8. The correlation engine receives alert A. However, alert A can only be received if alert A1 and A2 were previously seen. Consequently, the correlation process generates these two alerts. These two specialized alerts can be used for verification purposes, for instance.

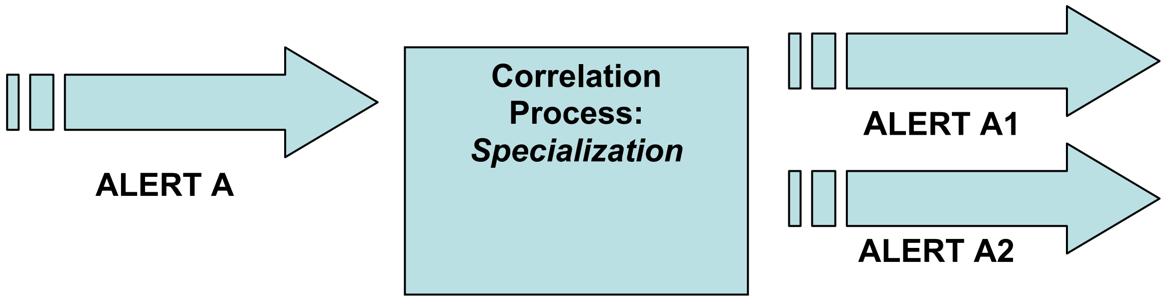


Figure 8: correlation operators, *Specialization*

3 Correlation approaches

3.1 Presentation

This section provides a panoramic view of alert correlation through the gathering of the main approaches existing in literature, classified according to the methods and algorithms utilized in the correlation process.

Many tools are presented in Section 4. They are either derived from research applications or distributed by companies as finished products. Thus, we intend to show in this section the main techniques and approaches implemented in these tools. We invite the interested reader to have a look at section 4 for more practical details on these tools.

Only two types of approaches have been identified by [Laz92]: the probabilistic approaches, on the one hand, and, on the other, the approaches in which system entities are modeled as finite state machines. Today, the number of available approaches is much larger. Some of these approaches are probabilistic, others utilize traditional artificial intelligence paradigms and others apply principles in non-conventional logics [Lew99, Smets88].There are also approaches which adopt *ad hoc* methods to deal with the alert correlation problem.

Generally speaking, we distinguish two main categories. The first category gathers all approaches that do not require a specific knowledge. As explained in Section 4, many tools which are currently available implement some of these approaches. They are often monitoring consoles, or simple log analysis tools.

Things become more complex when the correlation process relies on certain knowledge. Tools get scarce, but not research projects. The second category gathers all these approaches together. However, this knowledge can come from an expert, who reproduces it thanks to a given language, or it can be automatically deduced from learning techniques.

We propose to group the various research efforts on correlation according to the following categories:

1- rule-based

2- scenario-based

3- uncertainty reasoning

4- time reasoning

5- state transition graphs

6- neural networks

7- Bayesian belief networks

8- Context reasoning

This classification scheme is inspired by those presented in [Subra00], [Casey02] and [Lew99], the last one being the most complete summary of event correlation techniques observed in the literature.

They correspond to different reasoning approaches. Table 2 summarizes some of these reasoning with some associated research projects that implemented them. They are all detailed in the following sections. We explain in paragraph 3.11 how they can complement each other. Sections 2.2 to 2.6 provide a detailed insight of their characteristics.

Reasoning type	Rules-based	Attack Scenarios-based	Uncertainty	Temporal	Neural Networks-based	Bayesian-belief	Others
Manual knowledge acquisition	☑ Prolog tools ☑ SEC ☑ ASAX	☑ LAMBDA (MIRADOR Project) ☑ ADeLe ☑ JIGSAW ☑ Hyper-alerts	☑ Fuzzy Logic techniques ☑ Possibilistic models ☑ Dempster-Shafer Theory	☑ Chronicles	☑ Feed-forward Networks (Back propagation based algorithms) ☑ Self-Organizing Maps	☑ CIDS ☑ EMERALD e-Bayes	☑ STAT ☑ M2D2 ☑ IMPACT ☑ M-Correlator EMERALD
Automatic Knowledge acquisition	☑ Clustering techniques ☑ Data Mining: (Association rules, etc)			☑ LogWeaver		☑ SPICE	
Not Knowledge-based	Some log analysis tools (see Section 4.2) and database queries interfaces (see Section 4.3)						

An interesting sign that the field is maturing are attempts to build frameworks within which all alert correlation techniques can be evaluated. Haines et al. describe in [Haines03] the first experimental testbed to validate alert correlators. They give a framework to produce attacks and to compare correlators through specific *correlation metrics* (three dimensions: prioritization, multi-step correlation and multisensor correlation). Similar proposals should appear in the coming months. Furthermore, specific data sets can be created easily to test correlation tools. LLSIM, developed by the Massachusetts Institute of Technology, is an interesting configurable network simulator that allows producing a large variety of data sets without expensive testbeds [Llsim03]. It provides models of a small subset of the traffic types and sensor alerts that would be encountered in the real world. The objective is to provide realistic alert types, distributions, arrival rates, and also realistic content like source and destination addresses, and TCP/UDP ports. These features are important to many alert correlation systems.

It is worth mentioning here the following set of metrics suggested by Lewis as a way to compare different approaches for event correlation [Lew95]:

- knowledge representation
- knowledge acquisition
- computational overhead
- scalability
- learning
- adaptation

These dimensions are important to the evaluation of correlation methods. However, we are not using them in this section since it is quite difficult to apply these metrics on the abstract level used in this section. Nevertheless, they reveal themselves useful to categorize tools and that is the reason why we will apply them in section 4 when presenting existing tools.

3.2 Rule-based Methods

3.2.1 Rule-based approach: an introduction

Rule-based systems have received most of the attention and were the first to be developed. These are also referred to as expert systems, because the development of the rules requires intimate familiarity with the faults and their symptoms. This approach develops if-then rules for given sequences of events that arrive at the monitoring station. The general knowledge of a certain area is contained in a set of rules and the specific knowledge, relevant for a particular situation, is constituted by facts, expressed through assertions and stored in a database. A rule consists of two expressions -- well-formed formulas of predicate calculus [Nils80] – linked by an implication connective (=>), and is evaluated over a global database. The left side of each rule contains a prerequisite which must be satisfied by the database, so that the rule is applicable. The right side describes the action to be executed if the rule is applied. The application of a rule alters the database.

Every rule-based system has a control strategy (a.k.a. inference engine) which determines the order in which the applicable rules will be applied and which stops the computing process when a finishing condition is satisfied by the database. In comparison with the traditional programs, which contain in their code both the knowledge and the control information – which contributes to make them extremely complex and hard to maintain- a rule-based system is simpler, more modularized and easier to maintain, for it is organized in three levels [Cronk88]:

- an inference engine which contains the strategy to solve a given class of problems; it decides when to apply which rules.
- a knowledge base, containing a set of rules with the knowledge of a specific task, that is, an instance of that class of problems or attacks;
- a working memory, containing the data about the problem or attack being dealt with;

With reference to section 2.2.4, we observe that Rule-based approaches are in general, but not necessarily, executed *offline* and on a stateful mode. Rules are applied to alerts and *meta-alerts*. We distinguish two important questions in this approach: how are rules constructed? How are rules applied to inputs? *Rules Induction* answers the first question and is presented in section 3.2.2. *Rules Matching* answers the second one and is presented in section 3.2.3.

3.2.2 Rules Induction

There are two distinct ways to write rules. On one hand, experts can be asked to write them more or less explicitly. On the other hand, machine learning methods are developed to accomplish this tedious task. In the latter case, explicit symbolic classification rules are automatically constructed that generalize the training cases. One of the main attractions of rule induction is that the rules are much more transparent and easier to interpret than, say, regression model or a trained neural network (see 3.6 for more details).

The classification rule learning task can be defined as follows: Given a set of training examples (alerts or meta-alerts for which the classification is known), find a set of classification rules that can be used for prediction or classification of new instances, i.e. new incoming alerts or meta-alerts. A more formal definition of the classification rule learning task has to take into account the restrictions imposed by the language used to describe the data (data description language of alerts or meta-alerts) and the language used to describe the induced set of rules (hypothesis description language). The *language bias* refers to the restrictions imposed by the languages defining the format and scope of input data and knowledge representation.

We report the interested reader to [Berth03, chapter 7] for a more complete study of rules induction. They present two classical form of induction, named respectively *propositional rule induction* (the output are if-then rules) and *relational rule learning* (the output are relational rules, possibly in the form of Prolog clauses). Furthermore, they introduce the important notions of generalization and specialization, and they describe algorithms for rule and hypothesis construction. Moreover, they discuss some measures for evaluating the quality of rules (they explain especially the trade off between accuracy and generality).

The rule learning task can be applied by means of many algorithms and techniques, such as data mining (association rules). Mannila et al. first studied in [Manni95] how to identify frequent episodes rules in alert sequences. Their results have been used in the TASA system [Tasa96]. Subsequently, many researchers have tried to reuse data mining methods [Manni95, Agraw94, Agraw95, Srik96] to obtain rules. However, data mining often generates too many rules and usually requires administrators to identify the useful rules among all the results of the data mining process. In order to control this process, to make it more focus, and to reduce the number of rules being generated, many researchers have considered putting various constraints, including Boolean expressions, regular expressions, and aggregation constraints, etc., into the methods of mining association rules. Some of them can be found in [Srik97, NgLak98, Garof99].

In the intrusion detection domain, Manganaris et al. have applied these techniques to mine association rules in alert bursts in [Mang00]. Subsequently, alerts that are consistent with these association rules are deemed 'normal' and are discarded. Julisch et al. show in [Julisch02] that data mining can be used to support and partially automate the investigation of intrusion detection alerts. Specifically, they investigate two techniques, namely *episode rules* and *a conceptual clustering technique*. We report the interested reader to [Julisch03] for a thorough review of the usage of data mining techniques within the intrusion detection area.

3.2.3 Rules Matching

This operation consists in comparing rules to inputs. In the following, we adopt the definition of a "match" as proposed in [Wu94]:

Match: Find the rules in the rule base which LHSs (Left-hand side) are satisfied from the existing contents of the working memory.

Based on that definition, we distinguish, among the various techniques presented in the literature, two main families:

- **Exact Match:** the whole prerequisite expression of a rule must be matched. This is the case of most popular rule-based tools. One of them, called Simple Event Correlator (SEC) is presented in 4.3.1.
- **Partial Match:** The left hand side of a rule consists of a set of conditions (fact patterns) which have to be fulfilled in order to activate the rule and execute the actions located on the right hand side of a rule. There is a partial match if some, but not all, of these conditions are fulfilled. Then, decision techniques are applied to determine which action, if any, must be triggered for these partial conditions.

Two important rule-matching algorithms have been used and re-adapted for many years: RETE [Forgy82] and TREAT [Mira87]. The RETE Algorithm is generally recognized as the most efficient algorithm for the implementation of production systems[3]. The algorithm was originally developed by at Carnegie Mellon University in 1979 but has evolved dramatically over more than 2 decades. RETE is the only algorithm for production systems whose efficiency is asymptotically independent of the number of rules.

Many rule-based languages are based on this algorithm. Indeed, most rule languages used in expert systems have a common origin in the "Official Production Systems" (OPS) developed during the seventies at Carnegie Mellon University by several PhD students of Dr. A. Newell[4].

OPS5 was the first production system language based on the RETE Algorithm and the first AI language to succeed in commercial application when Dr. J. McDermott implemented a rule-based configurer of VAX computer systems for Digital Equipment Corporation. R1 was originally implemented in Lisp but was later ported to a Bliss version of OPS5 for performance reasons, after which DEC renamed R1 to XCON. XCON was tremendously successful and led to the development of a number of

[3] It is worth noting though that there are in the literature as well as on the web, several reports and white papers arguing the superiority of other ones, such as TREAT or LEAPS, over RETE.

[4] The historical presentation that follows is mostly based on the data available on line on the web at the following URL: www.haley.com/ReteAlgorithm.html

additional, substantial expert systems using OPS5, several of which were implemented by P. Haley. In 1984, P. Haley has developed the Automated Reasoning Tool (ART) and has implemented its own inference engine. ART extended the RETE Algorithm to support:

- Truth maintenance and logic programming
- Automatic subgoaling and backward chaining
- More expressive pattern matching language
- Support for arbitrary procedural functions and predicates
- More expressive knowledge representation capabilities
- Logical Quantifiers (other than *and* and *not*)
- A variety of other improvements beyond OPS5.

Although the ART syntax is much more expressive than that of OPS5, the ART syntax was designed as an incremental improvement beyond OPS5.

By 1985, NASA had standardized their AI development on ART but needed an implementation to run on IBM and Macintosh personal computers. As a result, the Software Technology Branch at Johnson Space Center cloned the forward chaining capabilities and syntax of ART and introduced the "C Language Integrated Production System" (i.e., CLIPS) into the public domain. Government funding for development or support of CLIPS was discontinued several years ago.

CLIPS is rarely used for commercial purposes due to the lack of continued funding and other, practical, commercial considerations, especially when compared with the functional advantages of inference engines available from other companies. Nonetheless, CLIPS remains popular for college-level courses on expert systems.

Following the distribution of NASA's CLIPS, Inference Corporation implemented a forward-chaining only derivative of ART/CLIPS called ART-IM, subsequently renamed "ART*Enterprise". The ART inference engine remains available from MindBox . This company is a spin-off from Brightware which, itself, is a spin-off from Inference [Mindb].

Developed by an employee of Sandia National Laboratories, JESS is a Java version of NASA's CLIPS that has added a few more of the capabilities and performance characteristics of contemporary, commercial rule engines. Its syntax remains roughly CLIPS compatible but has added some of the extensions first introduced in Eclipse by The Haley Enterprise. JESS is not public domain or open-source like CLIPS but can be downloaded at no charge for academic use. Therefore, it is also popular for college-level courses on rule-based programming.

In 1989, P. Haley founded The Haley Enterprise and developed Eclipse.

- Eclipse supports an extended version of the CLIPS syntax.
- Eclipse is the only C/C++ inference engine that uses the RETE Algorithm to support both forward and backward chaining.
- CIA Server (©Haley Enterprise) goes beyond the limitations of CLIPS, JESS, and other rules engines by sharing one knowledge base for scalability and multi-threaded performance
- Rete++ is a C++ class library encapsulates Eclipse and seamlessly integrates it within C++ on Windows, UNIX, and other operating systems
- Cafe Rete is a Java class library that implements the functionality of CIA Server for J2ME, J2SE, and J2EE environments

3.2.4 Additional Comments

The rule-based approach has been criticized because it has high maintenance costs, it does not automatically adjust to a changing environment, it does not scale well, and does not perform reliably under congested conditions [Smarts]. In other words, rule-based correlation is appropriate if the administrative domain is not large and the environment does not change much. A supplementary technique used with the rule-based approach is to parse network logs to create a database of rules, referred to as data mining [Sterr00, Gard98] or *machine learning* techniques [Michal83, Good91] (see Section 3.2.2).

Having its knowledge limited to the rules of its database, the system can not deal with the situations to which these rules do not apply. This affects its robustness [Lew95], since the system may lack alternatives in many common situations.

Another limitation of this approach is the fact that they do not take advantage of past experiences in the deductive process, that is, they lack '*memory'*. Therefore, a purely rule-based system that has triggered thousands of rules in order to, from a given set of alerts, deduce the occurrence of a given fault or attack, will trigger again all those rules whenever it is submitted to the same set of alerts, getting once again the same conclusion. Because they do not make use of past experiences, the rule-based systems are subject to repeating the same errors over and over again, which contributes to degrade the precision and performance of the system.

Despite these weaknesses, the approach is intuitive to develop and widely used. Examples in Network Management are BMG Patrol, Tivoli TME, Computer Associates TNG, Platinium ServerVision, etc [Lew99].

These tools are presented in Section 4.

3.3 Attack Scenario Methods

3.3.1 Attack Scenario-based approach: an introduction

Attack Scenario-based correlation is a knowledge-based method, like the rule-based approach described in 3.2. The specificity of this approach lies in the fact that rules are not generic but, instead, very precise to represent well-defined scenarios of attacks. The language used to express the scenarios plays an important role in the applicability of the method. As we see here below, this leads to the creation of new languages, instead of reusing well known ones (such as, e.g., first order logic, etc...).Like rules, *attack scenarios* can be specified by human users or learned through training datasets. This is currently a hot topic, and many research groups are developing their own approaches. We propose to have a deeper look at the most important of them.

3.3.2 LAMBDA and ADeLE (MIRADOR Project)

☑ LAMBDA (MIRADOR Project)

MIRADOR is a project initiated by the French Defense Agency (DGA) and is led by Alcatel in collaboration with three research laboratories (Onera, ENST Bretagne and Supelec). It aims at building a cooperative and adaptive IDS platform [CupMie02].

In this approach, every alert is modeled using the IDMEF format. However, the correlation function does not directly deal with this XML representation of alerts. It is automatically converted into a set of logical facts.

On the other hand, attacks are specified in the LAMBDA language. In this language, an attack is specified using five fields:

- Attack pre-condition: a logical condition that specifies the conditions to be specified for the attack to succeed.
- Attack post-condition: a logical condition that specifies the effect of the attack when this attack succeeds.
- Attack scenario: the combination of events the intruder performs when executing an attack.
- Detection scenario: the combination of events which are necessary to detect an occurrence of the attack.
- Verification scenario: A combination of events to be launched to check if the attack has succeeded.

The pre-conditions and post-conditions of an attack correspond to description of conditions over the *system's state*. For this purpose, a language based on the logic of predicates is used. Predicates are used to describe properties of the state relevant to the description of an attack. In practice, predicates are combined with conjunctions and negations. Other fields of an attack description (attack/detection/verification scenarios) are specified using event calculus algebra. This algebra enables the authors to combine several events using operators such as *sequential composition (";")*, *parallel unconstrained execution ("|")*, *non deterministic choice ("?")*, *synchronized execution ("&")* and *exclusion of an event when another event occurs ("if_not")*.

Once attacks are correctly modeled, some predicates are defined and some simple algorithms are applied in order to perform *attack correlation* [CupMie02, Cuppens01, Cuppens02]. Intuitively, correlation between attacks A and B means that A enables the intruder to perform attack B.

They suggest to automatically generate correlation rules, which would always have the form alert_correlation(Alert1, Alert2); this simply means that Alert1 and Alert2 can be correlated as part of a given attack scenario.

The method they propose is modeled by specifying possible logical links between the post-condition of an attack A and a pre-condition of an attack B. If such a link exists, then it is possible to correlate an occurrence of Attack A with an occurrence of attack B because we can assume that the intruder has performed A as a step that enables him to perform B. Some practical definitions of correlation are given in [CupMie02, Cuppens02] and are implemented in the Mirador platform.

☑ **ADeLe**

The Adele language has been developed simultaneously with the Lambda language within the Mirador Project. The specificity of the ADeLe language is to offer a common formalism to express not only the detection and correlation rules but also the actual code of the attack itself.

The motivation is quite ambitious: "ADeLe is designed to allow attack descriptions which are readable (…), comprehensive (it is possible to represent every aspect of an attack, i.e. from the attacker's and the defender's points of view, in only one description), generic (…), and modular (defining an attack composed of several attacks already described is allowed and defining a meta-alert also)" [MicMe01]. Actually, the primary goal of ADeLe is to combine all the knowledge available for a given attack in one and only one readable description.

The body of the attack description is made of three parts: the exploit part, the detection part and the response part. The description itself is written in an XML-like code.

However, ADeLe, which was designed in parallel with LAMDA for the Mirador project, does not appear to be used within the MIRADOR project. LAMBDA has been chosen to represent attacks. Cuppens et al. mention in [CupMie02] the possibility to include other fields in the attack description. For instance, the ADeLe language suggests introducing a *reaction* field to specify the actions to be launched when the attack is detected. As a consequent, ADeLe can be seen as an interesting complement of LAMBDA.

3.3.3 Hyper-Alerts approach vs. JIGSAW

Another approach closely related to LAMBDA is proposed by the Department of Computer Science from the North Carolina State University, and more specifically by P. Ning et Al [Cui02, Ning02, NingX02, Ning03]. These two approaches present multiple similarities since both are related to *attack scenarios* (i.e. steps that attackers use in their attacks). Peng Ning first suggested his approach to address some limitations of JIGSAW [Templ00].

JIGSAW was originally proposed to represent complex attacks, and the authors envisaged to apply it to correlate intrusion alerts. It is based on the preconditions and consequences of individual attacks; it correlates alerts if the preconditions of some later alerts are satisfied by the consequences of some earlier alerts. However, several problems make it difficult for JIGSAW to be a practical alert correlation technique. First, JIGSAW requires all the preconditions (i.e. required capabilities in [Templ00]) of an attack to be satisfied in order to consider its consequences. This is fine in theory; however, it has a negative impact on alert correlation in practice. In particular, if an IDS sensor fails to detect one of the attacks that prepare for later attacks, JIGSAW will miss the opportunity to correlate the detected attacks. Moreover, JIGSAW treats low-level attacks individually, and does not correlate an alert if it does not prepare for (or are prepared for by) other alerts. For example, if an attacker tries several variations of the same attack in a short period of time, JIGSAW will treat them separately, and only correlate those that prepare for (or are prepared for by) other alerts. In addition, JIGSAW ignores failed attempts of attacks even if they belong to a sequence of well planned attacks. Finally, no mechanism has been provided in JIGSAW to process alerts, though this has been speculated as an application of JIGSAW in [Templ00].

P. Ning et al. present in [Ning03] an alert correlation technique to address these limitations. The concept remains identical, based on the following observation: most intrusions are not isolated, but related as different stages of attacks, with the earlier stages preparing for the other ones.

First and foremost, they use two important definitions:

- *a hyper-alert type*. It encodes the knowledge about a type of attack; it includes all the prerequisites and consequences information. *Prerequisite* specifies what must be true in order for the attack to be successful, and *consequence* described what is true if the attack indeed succeeds.
- *A hyper-alert instance*. Given a hyper-alert type, it can be generated if the corresponding attack is detected and reported by an IDS sensor.

The knowledge comes from the *Knowledge base*. It contains the necessary information about hyper-alert types, as well as relationships between all predicates (*prerequisites* and *consequences* predicates). This table is filled by experts in a XML-like format.

Alerts are processed by the *alert-preprocessor* to generate *hyper-alerts* and instantiate the prerequisite and consequence sets of each hyper-alert. The knowledge comes from the *knowledge base*.

Then, the correlation engine tries to find out all the prepare-for relationships between these previously instantiated hyper-alerts. After that, they output these relationships as hyper-alerts correlation graphs. It represents attack scenarios constructed through the alert correlation.

All that has been recently implemented in an open-source tool called Intrusion Alert Correlator (IAC). In particular, Ning et al. discuss in [Ning04] additional techniques to hypothesize and reason about attacks missed by IDSs (false negatives), based on the indirect causal relationship between intrusion alerts and the constraints they must satisfy (prerequisites and consequences). They introduce two types of classic correlation methods: correlation based on prerequisites and consequences of attacks, which they call *causal correlation method*, and correlation based on similarity between alert attribute values, which they call *clustering correlation method.* In the latter case, the authors propose a technique, based on similarity between graphs of attacks, which is such that scenarios can be recognized even in the case of missing events. It is worth noting that Cuppens et al., in [CupMie02], in parallel, had proposed another approach able to deal with missed alerts. Their technique is based on abductive rules. If an alert, which is the known consequence of an event that should have generated another alert, is received by the correlation engine and if that other event has not been received by the same correlation engine then, by means of a simple abductive reasoning, the missed event can be identified. This approach, very different from the one by Ning et al., finds its limitations in the fact that it can not cope with missing events that are not linked to other events by means of cause/consequence relationships. Also, identifying all these relationships is a non trivial task that comes on top of the writing of the scenarios.

3.3.4 CAML

CAML (Correlated Attack Modeling Language) is a recent language presented by SRI International in [Caml03]. It aims at modeling multistep attacks, in the same way than previous approaches. Attack scenarios are trees. They identify logical steps (sub-goals) in attack scenarios and they specify some relationships among these steps: temporal, attribute values, and prerequisites. More precisely, attacks are described by *attack patterns*, which characterize common attack techniques from the detection point of view. As a result, detecting attacks in a correlation phase can be reduced to detecting instances of the attack patterns and their relationships.

Furthermore, CAML enables one to specify multistage attack scenarios in a modular fashion. A CAML specification contains a set of *modules*, which describe a correlation step. The relationships among modules are specified through pre- and post-conditions. Thus, modules can be linked together to recognize attack scenarios. This makes the approach very modular.

The structure of CAML events is based on IDMEF. CAML has been tested within the EMERALD framework. This work is very similar to those of LAMBDA, JIGSAW and P. Ning. However Cheung et al. have apparently spent a large effort to solve implementation issues (modules reuse and modules interfacing with the EMERALD existing).

3.4 Context reasoning approaches

3.4.1 Introduction

There are many ways to model knowledge. In the previous section, attack scenarios are built to correlate incoming alerts. However, other approaches are built on different knowledge models. In this section, we present some approaches that rely on ‘context models’. The environment where alerts are issued is modelized in different manners.

3.4.2 IMPACT

This prototype [Jacob93] was developed for network management purposes. They describe a network configuration model, and a network-element class hierarchy. The network configuration models describe the Network-Elements (managed objects) and the connectivity and containment relations between them. The network-element class hierarchy describes the NE types and the class/subclass relationships between the types. Then, the conditions under which the correlations are asserted are described by correlation rules, which are provided by expert knowledge.

This example simply shows that the model-based approach has been used for many years in the Network management area. Other examples exist in different domains. For instance, [Zheng03] implements a classic model-based approach to ease the alarm correlation process in GSM Networks.

3.4.3 EMERALD M-Correlator

EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) is an environment for anomaly and misuse detection and subsequent analysis of the behavior of systems and networks. EMERALD is being developed under DARPA Contract by SRI International [Neum99, Porr97, Porr02]. It employs a building-block architectural strategy using independently tunable distributed surveillance monitors, which should detect and respond to malicious activity on local targets, and should interoperate to form an analysis hierarchy. The description of this architecture, with its specific sensors is beyond the scope of this paper. However, we are particularly interested in one component called Mission-Impact Intrusion Report Correlation System, or M-Correlator [Porr02].

M-Correlator is designed to consolidate and rank a stream of alerts relative to the needs of the analyst, given the topology and operational objectives of the protected network. More concretely, alerts are vetted against the network model. A *relevant score* is produced through a comparison of the alert target's known model against the known vulnerability requirements of the incident type (i.e. incident vulnerability dependencies). Consequently, this approach is quite different from the previous ones. There is no attack scenario built. However, a network model is created and alerts are then processed using the vulnerability requirements of the incident type together with the model knowledge.

In addition to the *relevant score,* another parameter is computed: the importance of the asset that the alert targets. Based on these two parameters, M-Correlator selects alerts that will cause greatest risks to the monitored network.

M-Correlator maintains an internal topology map of the protected network, which is dynamically managed by the analyst. Automated topology map generation is supported using *nmap* [Fyo97], through which M-Correlator can identify the available assets on the network, IP address to hostname mappings, OS type and version information, active TCP and UDP network services per host, and hardware type. Nmap is periodically run to update the topology database.

3.4.4 M2D2

Morin et al. introduce in [Morin02] a formal information model for security information representation and correlation. The model includes four types of information: information system's characteristics, vulnerabilities, security tools and events/alerts.

The information system's characteristics depend on both the topology (network of hosts) and the products used. Vulnerability affects a configuration. In that way, a host is vulnerable if its configuration is a superset of one vulnerable configuration. Security tools group IDSs and vulnerability scanners that detect exploited or latent vulnerabilities.

A formal format to structure these information types is detailed in [Morin02]. Furthermore, some examples of correlation are specified, using the formal bases of

M2D2. They consist in aggregating alerts. In other words, they group alerts following various criteria provided by the M2D2 information (by targeted hosts, by vulnerabilities, etc).

3.5 Time reasoning approaches

3.5.1 Chronicles

The system of chronicle recognition aims at giving an interpretation of the system evolution given dated events [Douss93, Douss94]. It takes as input a stream of dated events and recognizes instances of chronicles as they are developing. It is mainly a time-series reasoning system that relies on the *reified temporal logic formalism* [MaKnig, Morin03]. In the Artificial Intelligence literature, chronicles are related to other approaches such as *plan* recognition[5] and *event calculus*[6]. It is predictive in the sense that it predicts forthcoming events relevant to its task, it focuses its attention on them and it maintains their temporal windows of relevance. Its main function is to efficiently recognize complex temporal patterns on the fly, as they occur.

Each chronicle can be viewed as a set of patterns and a set of temporal and contextual constraints over them. If the observed events match the chronicles patterns and if they occur as the contextual and temporal constraints allow them to, then an instance of the modeled chronicle is recognized. However, some hypotheses are made on the events. First, all events specified in a chronicle must be observable, i.e. unobservable (or

[5] Plan recognition is the act of reasoning from a set of observed actions to goal, a possible next action, or a complete plan *recipe* (a sequence of steps for completing a plan). It can be viewed as another instance of the classification problem.

[6] The event calculus is an attempt to codify intuitive reasoning about action and change in such a way that the frame problem is avoided. It is based upon the supposition that all change must be due to a cause–spontaneous changes that do not occur. In this way it achieves a formalization of the common sense principle of inertia: ‘normally, nothing changes’. That is, if an action *a* does not affect a property F, then if F is true before doing *a*, it will be true after.

unrecognized) activities are not included in the chronicle expression. It is also assumed that the events are reported to the system as they occur, and that they must be collected in the same order as they occur (*synchronization hypothesis)*.

This approach has several advantages:

- Chronicles based system gives an efficient recognition process.
- The hypothesis stating that all actions are observable makes unnecessary the abduction of unobserved events.
- It is possible to define so-called “deactivating events” that invalidates a partially recognized chronicle (similar to the anti-correlation definition in Lambda).
- The explosion of the search space is more limited than with other approaches, such as the plan recognition approach.

The chronicles main advantages are consequences of the strong hypotheses made. Unfortunately, the synchronization hypothesis and the hypothesis which states all intruder’ s actions specified in a chronicle can be detected , are very hard to fulfill in security domain. These assumptions and the fact that this system is based on a chronicle library lead to the following drawbacks:

- As for plan recognition systems (and many other approaches), the exhaustively of the plan library is a main concern.
- Including or not in the chronicle an event which is sometimes not detected may lead to false positives and false negatives.

Consequently, a chronicle system is especially efficient to recognize stereotyped attack scenarios, such as the ones launched by automatic intrusion tools. In this case, it is quite straightforward to represent each attack scenario by a chronicle. But it has some important limitations in other cases. A noticeable exception comes from Morin et al. They introduce in [Morin03] a new chronicle formalism. Instead of applying chronicles to model attack scenarios, they use chronicles to represent known phenomena which involve several alerts, and to strengthen and enhance single alerts by combining them with other events, as well as other information not found in the alerts. For this purpose, they suggest using M2D2 (described in Section 3.4.3). M2D2 information is modeled and served as non temporal information to the chronicles logic. Actually, there is a two-way relationship between M2D2 and the chronicles: as a correlation system, the chronicles take advantages of the data provided by M2D2 and acts as an alert provider for M2D2.

3.5.2 ASAX

The Project ASAX, “Advanced Security Audit-trail Analysis on UniX” aims at supporting intelligent analysis of audit trails [Asax92]. Initially, ASAX was not really a correlation engine, rather a detector, but it could fit nicely that objective. The audit trail is normalized thanks to the specific language called NADF described in 2.2.3. In particular, ASAX uses a general rule-based language to carry out its analysis, named RUSSEL (Rule-baSed Sequence Evaluation Language). This language was initially tailored for processing large sequential files in one pass, efficiently. It aims at recognizing particular patterns in files and triggering appropriate actions, e.g. activating other alerts, or sending reports to administrators.

Thus, RUSSEL can be viewed as a procedural language, including a particular *predefined control structure*, which allows making reasoning about sequences of records. This control structure is based on a rule triggering mechanism that can be roughly summarized as follow:

- The audit trail is analyzed sequentially, record by record, by means of a collection of rules. At a given time, there is a current record being analyzed and a collection of rules which are active.
- Active rules encapsulate all the relevant knowledge about the past of analysis. This knowledge is then applied to the current record by executing the rules for that record. The process, in turn, generates new rules to be applied latter.
- From the programmer point of view, a rule is like a classical parameterized procedure but which may involve statements for a particular kind of actions; triggering rules off for instance.
- The process is initiated by a set of rules activated for the first record.

Consequently, programming correlation in RUSSEL consists in writing a number of adequate rules which will be activated according to the global scheme above.

However, ASAX has some limitations: There is no real rules database and data types are limited within RUSSEL. This makes ASAX approach constraining and difficult to apply in practice. A new Java-based module is under development which should relax some of the limitations observed with the existing implementations and facilitates a wider usage of that tool.

3.5.3 LogWeaver

LogWeaver is a log auditing tool which checks a log file against temporal logic signatures [Rog01]. This approach is very similar to ASAX's one. *LogWeaver* could, most likely, be adapted to correlate alerts coming from various sources rather than events stored in a single log file. The authors implement a logic, consisting of flat, Wolper-style, linear-time formula,, which is very promising. It allows detecting complex correlation of events, on-line, using a declarative set of signatures, expressed in a variant of temporal logic. However, one drawback is the potential combinatorial explosion of the algorithm if signatures are not carefully designed. It is worth noting that the same problem exists with ASAX.

3.6 Reasoning in the presence of Uncertainty

3.6.1 Fuzzy logic approach

Due to the complexity of managed systems, it is not always possible to build precise models. The knowledge of cause and effect relations among attacks and alerts is generally incomplete. Besides that, some of the alarms generated by an attack or a fault are frequently not made available to the correlation system in due time, because of losses, corruption or delays in the route from the element which originated them. Finally, due to the fact that configuration might change frequently, the more detailed a model is, the faster it becomes outdated. The imprecision of the information supplied by experts very often causes great difficulties. Consequently, fuzzy logic has been proposed as an alternative to deal with the uncertainty and the imprecision which characterize some applications.

The basic concept underlying fuzzy logic is the *fuzzy sets*. In classic logic, given a set A and an element X, the expression “X is a member of A” is supposed to be evaluated either as *true* or *false*. When it comes to fuzzy sets, each element X has, in relation to the set, a certain *grade of membership*, which may take any value between 0 (when the element definitely does not belong to the set) and 1 (when the element is certainly a member of the set). The concept of fuzzy set brings in itself the novelty that any given proposition does not have to be only true or false, but that it may be partially true, in any degree in a scale 0 to 1. Through a specific algebra, several operations on fuzzy sets are defined (for example, complementation, intersection and union).

Other similar models exist to handle the issue of reasoning in the presence of uncertainty, namely possibilistic models[7] and Dempster-Shafer Theory (DST)[8].

Classical expert systems are programs that emulate the reasoning of human experts or perform in an expert manner in a domain for which no human expert exists. Rule-based and model-based approaches (attack-scenarios or context models) belong to this category. However, these expert systems typically try to cope with uncertain and imprecise information (false positives, false negatives, alert omission, etc), using various ad hoc methods. On the other hand, these new models provide built-in capability to express rules based on uncertain knowledge. Each model will provide different mechanisms to define and to handle the variables “important” and “high”. The rules are typically of a form similar to the following:

IF alert is *important* AND target is *vulnerable* THEN attack is *very likely*

---

[7] A possibilistic propositional base is a collection of logical statements associated with qualitative certainty levels. We report the interested reader to [Dubo92] for more information on this approach.

[8] DST is designed to deal with the distinction between uncertainty and ignorance. It is very interesting in the case of incomplete models as some beliefs can be left unspecified. We report the interested reader to [Klir94] for more information.

Where 'alert' and 'target' are (linguistic) input variables, 'important' is one of the possible linguistic values of the variable 'alert' and 'high' a linguistic value of 'target'.

Unlike rule-based approaches, which use knowledge-based rules, these systems use uncertainty-based rules (rules coping with uncertain knowledge). However, it is correct to say that they are a special case of rule-based systems.

A number of applications using these systems have been implemented in several areas such as Strategic Planning, Geology, Medicine, Environmental Sciences or Electrical Engineering. However, some researchers argue that all problems that may be solved by means of these new logics can be equally solved by means of probabilistic models such as, for example, Bayesian methods, with the advantage of counting, in the latter case, on a solid mathematical basis which, they claim, is lacking in the other models [Luna94].

To conclude, we want to point out that all complex correlations systems that were built so far required more than just one basic technology in order to be successful. In a large measure, techniques from fuzzy logic and from artificial intelligence for instance are complementary rather than competitive. These kind of resulting hybrid systems will (most likely) be more and more important in the future [Klem94, Abra01, PerrMe].

3.7 State Transition graphs

3.7.1 State transition graphs approach: an introduction

The concept behind a state-transition graph is very similar to a rule-based approach. The difference lies in the representation of the rules which is implemented by means of a state, a token, an arc and the movement of a token from one state to another via an arc [Lew99]. The advantages and disadvantages of the approach are similar to the rule-based approach: building the states and transitions are similar to building rules. Thus, it is appropriate for smaller, stable systems. Examples in network management are HP OpenView used with Veritas' NerveCenter [Lew99].

3.7.2 STATL: overview

STATL is an extensible state/transition-based attack description language.

In other words, it is an extensible language that is used to represent graphically attack scenarios. The language defines the domain-independent features of the STAT formalism. Attack scenarios are represented as sequences of transitions that characterize the evolution of the security state of a system. This characterization of attack scenarios allows for an intuitive graphical representation by means of *state transition diagrams* (see Figure 1).

In an attack scenario, *states* represent snapshots of a system's security-relevant properties and resources. A description of an attack has an "initial" starting state and at least one "compromised" ending state. States are characterized by means of *assertions*, which are predicates on some aspects of the security state of the system. For example, in an attack scenario describing an attempt to violate the security of an operating system, assertions would state properties such as file ownership, user identification, or user authorization. *Transitions* between states are annotated with *signature actions* that represent the key actions that, if omitted from the execution of an attack scenario, would prevent the attack from completing successfully. For example, in an attack scenario describing a network port scanning attempt, a typical signature action would include the TCP segments used to test the TCP ports of a host.

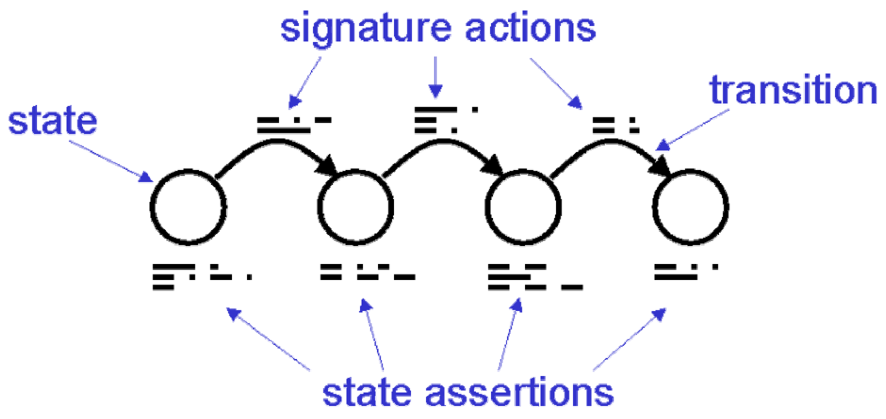


Figure 1: A State Transition Diagram

The STATL language can be extended to express the characteristics of a particular domain and/or environment. The extension process includes the definition of the set of *events* that are specific to the particular domain or environment being addressed and the definition of new *predicates* on those events.

For example, to extend STATL to deal with events produced by the Apache Web Server one would define one or more events that represent entries in the application logs. In this case an event would have the fields `host`, `ident`, `authuser`, `date`, `request`, `status`, and `bytes` as defined by Apache's Common Log Format (CLF). After having defined new events it may be necessary to define specific predicates on those events. For example, the predicate `isCGIrequest()` would return true if an event is a request for a CGI script. Event and predicate definitions are grouped in a *Language Extension Module*. Once the event set and associated predicates for a Language Extension Module are defined, it is possible to use them in a STATL scenario description by including them with the STATL *use* keyword. Extensions for TCP/IP networks, Sun BSM audit records, IDMEF Alerts, and Apache event logs have been developed.

STATL scenarios are matched against a stream of events by a system called 'STAT Core'.

The State Transition Analysis Technique has been initially used as the basis for developing a host-based intrusion detection system called USTAT. Later on, the technique has been extended to network traffic analysis and a network-based intrusion detection systems, called NetSTAT, has been developed. In 1998 and 1999 NetSTAT and USTAT were evaluated as part of both the MIT Lincoln Laboratory's off-line intrusion detection system evaluation and the Air Force Research Laboratory (AFRL) real-time evaluation. They showed promising results.

However, we are more interested by a more recent tool called AlertSTAT.

AlertSTAT is a STAT-based intrusion detection system whose task is to fuse, aggregate and correlate alerts from other intrusion detection systems –or sensors. Therefore, alertSTAT uses the alerts produced by other sensors as input and matches them with respect to attack scenarios that describe complex, multi-step attacks, using a STAT core system. Resulting alerts are an aggregated report that conveys a higher level view of the overall attack process.

AlertSTAT operates on alerts formatted according to the IETF's Intrusion Detection Message Exchange Format (IDMEF) proposed standard [Idmef]. Furthermore, a number of attack scenarios have been developed, including the detection of complex scans, "many-to-one" and "one-to-many" attacks, island hoping attacks, and privilege escalation attacks [Vigna03].

3.8 Neural networks

An Artificial neural network (ANN) is a system constituted by elements – *neurons*- interconnected according to a model inspired by the neural network existing in the human brain. Conceptually, each *neuron* may be considered as a simple autonomous processing unit, provided with local memory, and with unidirectional channels for the communication with other *neurons*. The functioning of an input channel in an ANN is somehow similar to the operation of a *dendrite* in the biological neurons. In an analog way, an output channel has an *axon* as its model. A neuron has only one axon, but it may have an arbitrary number of dendrites. The output 'signal' of a neuron may be utilized as the input for an arbitrary number of neurons [Kumar94].

In its simplest form, the processing carried out in a neuron consists in calculating the weighted sum of the signals present in their inputs and of generating an output signal if the result of the sum reaches a certain threshold. In the most general case, the processing may include any type of mathematical operation on the input signals, also taking into consideration the values stored in the neuron's local memory [Meira97].

The distributed control and storage of data and parallelism are interesting features of the ANNs. Besides that, an ANN does not require a previous knowledge of the mathematical relationship between inputs and outputs, which may be automatically *learned,* during the system's normal operation (thanks to algorithms such as *Backpropagation*, *Quickprop*, *Cascade Correlation*, etc). This makes them a good alternative for applications such as alarm correlation or fault diagnosis where the relationships between faults and alarms are not always well defined or understood and where the available data is sometimes ambiguous or inconsistent.

More concretely, Neural Networks seem not to be frequently applied in Alert Correlation tools. One first and important reason is that it is difficult to 'know' what is exactly done inside the ANNs. This is problematic for the administrator's understanding of outcoming alerts.. A second reason is that most of the efficient ANN implementations are commercial and proprietary solutions. As a result, it is ever hard to know which approaches are implemented inside. There are is a notable exception presented in [Wiet97] where Wietegrefe et al. apply neural networks to correlate alarms from Cellular Phone Networks.

Nevertheless, neural Networks are widely used in Intrusion Detection Systems [Deb92, Lippm99, Biv02, DuSha03]. Neural Networks is a strong and well-studied domain. Thus, it is possible to see this technology being soon applied to a larger number of alert correlation tools.

3.9 Bayesian Belief Systems

3.9.1 Bayesian belief approach: an introduction

Probabilistic reasoning using Bayesian statistics is an approach that attempts to overcome the problems of knowledge acquisition, scalability, learning and adaptation. Typically, Bayesian models are used to monitor performance and determine abnormalities. This is done by profiling critical system parameters under normal conditions, then monitoring the system: a *subjective* probability expresses the *degree of belief* of an expert related to the occurrence of a given event/alert, based on the information this person has available up to the moment [Henr91]. The use of subjective probabilities is very often the only resource, in situations where analytical or experimental data is very hard, or even impossible, to obtain. Consequently, Bayesian statistical methods are able to combine prior knowledge with partial statistical data to make inferences on other parts of the system. However, it is difficult to implement in practice. Very few commercial products are known to use this technique specifically, but it is being seriously studied [Hood97, Sterr00, Thot98]. One notable exception comes from the Honeywell technology Center, in cooperation with DARPA. They produce Argus, "*an Architecture for Cooperating Intrusion Detection and Mitigation* *applications"*. This tool correlates information from multiple intrusion detectors by applying qualitative Bayesian estimation technology [Argus].

We present in the following sections two projects that implement this approach. Both were included in tools, named eBayes and SPICE.

3.9.2 EMERALD

Valdes et al. introduced in [Val00] a probability-based sensor called eBayes. Ebayes analyses TCP sessions, which are temporally contiguous burst of traffic from a given source. The analysis is done by Bayesian inference at periodic intervals in a session, where the interval is measured in number of events or elapsed time. Between inference intervals, the system state is propagated according to Markov model. So, after each inference, the system emits an alert for sufficiently suspicious sessions. Even if Valdes writes in [Val01] that "the Adaptive Model –based Monitoring effort of eBayes permits to correlate and comprehend the results of various sensors, both for improved sensitivity and for false alarm suppression", this is not really exact. eBayes is a sensor (or a couple of sensors: respectively eBayes-TCP and eBayes-Host) without any alert correlation capacities. However, Valdes et al. provide in [ValdSkin] a more serious alert correlation proposition. First, they introduce a new approach, called *alert fusion*.

They maintain a permanent list of 'meta-alerts' that are possibly composed of several alerts, potentially from heterogeneous sensors. For two alerts, typically a new alert and a meta-alert, they identify features they have in common (*feature overlap*). Such features include the source of attack, the target IP and port, the class of the attack and time information. With each feature, they have a similarity function that returns a number between 0 and 1, with 1 corresponding to a perfect match. For attack class similarity, they maintain a matrix of similarity between attack classes, with values of unity along th diagonal and off-diagonal values that heuristically express similarity between the corresponding attack classes.

The probabilistic approach provides a technique to use efficiently the deductive knowledge from the model (meta-alerts). This simply means that the model and the way to use it are two distinct parts that can be implemented by various techniques. One clear example of this is given in the next paragraph.

3.9.3 Collaborative Intrusion detection System (CIDS)

[Cids] presents the design and implementation of a framework of a collaborative intrusion detection system [Bagc03]. CIDS employs multiple specialized detectors (at different layers) and a manager based framework for "aggregating the alerts from the different detectors to provide a combined alert to the administrator". We are particularly interested in this last aspect. First and foremost, we need to clarify a terminological issue. Aggregation, for the authors of CIDS, is different from what we have defined above; it corresponds to the generic notion of "correlation" as discussed in this paper.

The correlation is the *Inference Engine* task. It processes alerts in the queue and it comes up with the determination of potential intrusion detection. The determination is performed by matching the observed events against a signature database (or deductive knowledge).

However, the authors design two ways to use this knowledge. One is a graph-based inference engine, and the other one is a Bayesian network based inference engine.

They build the Bayesian graph construction starting from the signatures. It describes the conditional probability relationship among alerts and the conditional probabilities to specify. Then, when a new alert comes into the Inference queue, the inference function provides the probability of the root node, which is the probability of the attack based on the observed evidence.

Their presentation is very clear. It highlights the fact that many techniques can be applied to use previously-defined knowledge. There are no exclusive techniques. We can even imagine using many techniques in parallel, in order to exploit pros and cons of each of them and to obtain complementary correlation information.

3.9.4 SPICE

SPICE (Stealthy Probing and Intrusion Correlation Engine) is a tool proposed by Silicon defense in [StanHoag] to detect portscan. Even if they use simple log events as input for their correlation process, we believe that their approach can be extended to other inputs, and specifically alerts.

It combines events based on similarity of certain events attributes. To be more precise, source and destination IP addresses and ports are used for determining similarity, and Bayesian graphs are drawn with links between related alerts.

Their design maintains a probability model for the total activity on the network. To do so, they use nested self-balancing binary trees to encode the joint probability tables. Then, they maintain historical state for anomalous events (keeping them longer, the more anomalous they are). They apply a *simulated annealing* method (see [StanHoag] for further details) to cluster events together into portscans, using some heuristics they developed from real scans.

The approach uses classic probabilistic methods and is very interesting. One drawback, however, is that it might miss out a quite large set of related events.

3.10 Other approaches

3.10.1 Codebooks

The codebook approach is somewhat very similar to the rule-based approach, but rather than being treated separately, all events are grouped into an alarm vector, which is then matched to problem signatures, stored in a so-called codebook [Klig95].

Relationships between known events/alerts sequences are characterized by a correlation matrix, essentially a binary matrix that maps incoming events/alerts to attack signatures. That is, a matrix of *ones* and *zeros*, where events/alerts are the rows, and attack signatures the columns. A *one* in the matrix indicates the event/alert in that row is necessary to perform successfully the attack in that column; attacks can require more than one initial event/alert to occur.

In real time, each situation of abnormality may be described by means of an attack vector $\bar{a}$ where each element indicates the occurrence or not of the corresponding attack. The correlation is made through the choice, in the codebook, of the problem p, whose code is closest to the vector $\bar{a}$, in terms of a given distance, for instance the Hamming one [Ham50].

The codebook approach always produces a diagnosis, as opposed to the rule-based scheme, though some diagnoses will be more likely than others. Unfortunately, the codebook approach requires the same expert knowledge as a rule-based system in order to accurately populate the codebook, though Yemini et al. claim in [Yemi96] that problem signatures can be created automatically. Furthermore, many of the details of this system appear hidden behind corporate patents; so it is hard to find evidence to backup their claims.

This approach has been pioneered in Network Management by System Arts Management, Inc. with their InCharge product. Research examples of this integrated to HP OpenView or IBM NetView have been published [Lew99, LoChen00]. However, no solution currently exists in Alert Correlation.

3.10.2 Vote

In correlation by voting, each element must express its opinion on a specific topic. Then, a majority rule (absolute majority or k-majority for instance) is applied on this set of opinions (i.e. votes). The rule's consequence aims at representing the opinion of the majority of elements. A further extension can then be added to handle abstention from voting. We report the interested reader to [Dough02] for more details on voting rules. Rules are compared in their ability to fulfill some criteria, such as the Pareto criterion, the Expected Social Gain, etc…

Such a technique is presented in network management by Houck et al. in [Houck95]. When a node detects a strange behavior in the system, it sends an alert to the correlation system. In its alert, the node includes a *WHERE description* field, which indicates the possible location of the problem. Then, the correlation system forwards this *WHERE* information (which might be quite vague) to other nodes in the system and asks each of them to give its opinion concerning the location of the problem. The causal node is determined on the basis of these collecting votes (i.e. opinions).

A similar technique can be adapted to intrusion detection systems in a distributed environment. Du et al. implement such an approach in [DuSha03] to identify *poison messages*. Each subnetwork manager outputs a score to design a particular message type (score based on certainty of such messages detection). Votes are then combined to obtain more relevant information on the message type. Actually, this operation is executed each time a ‘message poison’ alert is detected.

The correlation by voting technique may be associated to other techniques such as dependence tree search [Houck95], which allows the identification among the components of the most voted elements, the most probably responsible for the fault that caused the alerts. Moreover, the correlation by voting allows each node to be an active element in the correlation process. Nevertheless, we are not aware of alert correlation tools that implement this approach beside the one presented in [DuSha03].

3.10.3 Genetic Algorithms

Genetic algorithms were invented by John Holland in the 1960s. It is a method for moving from one population of "*chromosomes*" to a new population by using a kind of natural selection together with the genetics-inspired operators of *crossover* (exchanging genetic material between two single chromosome haploid parents), *mutation* (flipping the bit at a randomly chosen locus), and *inversion* (rearranging the order in which genes are arrayed). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones.

It turns out that there is no rigorous definition of ‘Genetic Algorithms’ accepted by all in the evolutionary-computation community. However, it can be said that most methods called "GAs" have at least the following elements in common: initial populations, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

One illustration is presented by L. Mé in [Mé98]: GASSATA, which is a GA-based tool designed to deal with audit trails. However, there are some drawbacks which make this solution very limited. First, an *attack-event* matrix which gives the set of events generated by each attack must be initially defined. This work can be very tedious.

Secondly, GASSATA does not take into account missing events. Then, it cannot detect multiple realizations of a particular attack.

Finally, there exist many variations of GAs, and they are often used in combination with other methods (fuzzy logic, machine learning…). However, we are not aware of other alert correlation tools that implement such a technique.

3.10.4 An endless list

Due to the importance of the Event Correlation domain, techniques and paradigms coming from different fields have been proposed as potential good candidates to solve the issues. As of today, none of them has managed to bring some concrete solutions on the table. This is why we choose not to discuss them here. However, certain concepts are certainly worth being investigated, and could offer interesting result in the near future. This is certainly true for the following ones:

- Visualization techniques [SteCu00]
- Event Forwarding Discriminator (EFD) [Ietf00]
- Divide and Conquer paradigms [Bayer02]
- Constraint Satisfaction Problem (CSP) [Sabin97]
- Distributed and Cooperative architectures (multicast) [Elvin]

Alert Correlation, however, is a very recent topic. There are currently few available solutions, and the current approaches are almost all listed in Sections 3.2 to 3.7. Worse, these approaches are not implemented in practical tools. This gives the negative impression that tools development and research approaches are totally isolated. Moreover, we notice a non-surprising opacity of techniques used in the few existing commercial products, perhaps because of patent restrictions.

3.11 Complementary approaches

The categories summarized above cover a large field of research. The field is being driven by the increasing problem of event and alert correlation as systems increase in complexity and importance. Achieving standard representation of correlation events and alerts is a recognized industry goal. Although some private sector efforts claim success using rule-based, context-based and codebook-based approaches, these appear to be based on highly specific examples that have not been subjected to open evaluation from outside sources [Yemi96, Lew95]. Published reviews that attempt to compare event or alert correlation products have been rare [Board01]. Without further evidence, effective solutions to date appear to be so costly that they have been, and are likely to remain, limited to the relatively few organizations that can afford them.

There is not a unique solution that is the 'best', in terms of precision and/or complexity, to solve a generic problem of alert correlation. Recent researches indicate a tendency for the adoption of combinations of different approaches for the solution of the problem in complex networks [Lew99].

For instance, in the 'codebook' approach description we claimed that the codebook could be seen as a matrix where each row corresponds to a symptom (or event or alert) and each column corresponds to a problem (or fault, or attack). Each element of a column vector contains a measure of the causality of the problem to the corresponding symptom. We said that the value could be either 0 or 1. However, it is not completely exact and it is not demanded that the values of the causality measures belong to the set (0,1); the model allows these values to belong to any *semi-ring*, which constitutes a special class of partially ordered sets. This leaves open possibilities to use several approaches to describe the *likelihood* of causality, such as deterministic, probabilistic, fuzzy logic and temporal models.

Other examples are SPICE [StanHoag] and the probabilistic alert correlation method [ValdSkin]. They correlate alerts based on the similarities between their attributes. Though they are effective for correlating some alerts (e.g. alerts with same source and destination IP addresses), they do not discover causal relationships between related alerts and they can be complemented by scenarios-based methods [Ning03].

This simply shows that many solutions currently exist. Each of them proposes a new solution with different input information and correlation technical approaches. We find them quite complementary, even if they tend, at first glance, to ignore each other. The choice depends mainly on the conditions where correlation should be applied. The first step consists in defining carefully the operational constraints:

- the input information;

- the correlation objectives (approximate vs. exact summary of alerts for instance);
- the available resources. Some techniques require high-level knowledge to be installed and/or maintained;

These hypotheses contribute to select some solutions among the panel of methods. Actually, a correlation approach is often composed of multiple correlation approaches. Thus, many papers propose a combination of correlation techniques, in order to carry out a more efficient correlation.

3.12 Comparisons and summary

A comparison between methods and algorithms presented in the previous section is a difficult process and must take into account some factors among which, in general, there is a tacit agreement:

1- facility for constructing a theoretical model
2- implementation complexity
3- facility to adapt the changes
4- performance
5- precision

Rule-based approaches are appropriated for the correlation in systems whose configuration is rarely altered; the high costs of implementation and adaptation to changes make it difficult to apply these strategies to large systems (with large amount of alerts). Other approaches, such as the probabilistic ones are less sensitive to changes in the system.

Furthermore, the nature of the *application* to which the correlation is applied (for example, a reduction of the amount of information to be analyzed by the administrator, the identification of the faults that originated alerts, a prediction as to the occurrence of faults in the future, etc) determines the *type* of correlation to be used (see 2.3), which by its turn, must be taken into consideration in the choice of the method or algorithm to be adopted in a given situation. Alert compression, selective suppression, simple filtering, counting and specialization are examples of correlation types that may be implemented by using rule-based conventional approaches.

Applications involving filtering, scaling or temporal relationship may also be implemented. Nevertheless, the increase in the complexity of the problem is a consequence of *exceptions* (i.e. unknown cases) which are not always suitably treated by some traditional approaches. For instance, simple techniques which are based on some expert knowledge do suit well with so-called *exceptions*. Beside that, the possibility of the occurrence of non-identified exceptions affects the system robustness, which does not have alternatives to deal with these situations.

The most interesting applications of alert correlation, such as fault diagnosis, generally involve *generalization* or *clustering*. In these cases, the problem complexity makes it difficult to obtain of exact solutions, *uncertainty* being an important factor in the correlation process. Among several approaches to deal with uncertainty in alert correlation, fuzzy logic, Bayesian networks, voting-based reasoning, codebooks and artificial neural networks are noteworthy. There is much controversy involving advantages and drawbacks of each alternative. For example, defenders of fuzzy logic based approaches argue that they rather simplify the development of applications and result in products that work and that have an excellent performance.

We have listed five comparison factors at the beginning of this section. The first three factors that must be taken into account while comparing methods are easily evaluated in general. However, it is harder to understand the two last ones, respectively *performance* and *precision*. Nevertheless, noticeable exception has been recently published by the IEEE Computer Society [Haines03]. Haines & Al. point out the need to compare correlation systems. They describe an environmental setup to validate various correlation systems with the explicit goal of quantifying their ability to recognize cyber attacks and correctly designate their targets. It is a first and interesting step. Therefore, they compare in [Haines03] five correlators to validate their testbed. However, a more scientific comparison needs to be performed between combinations of correlation algorithms. Indeed, the five correlators which are used in the experiment are very different (some of them having *generalized capacities* and others being very *attack specific*).

To conclude, this chapter has demonstrated that there currently exists a large variety of correlation approaches. Most of them are implemented in network management solutions. Some are still in their experimental phase. In alert correlation, things are different. We do not find such a variety of approaches. Research groups tend to focus on same sorts of techniques: for instance, a particular attention is given on *scenario-based* approaches. We believe, however, that this will change in the coming months. New techniques are emerging, trying to adapt advances of Event Correlation to the security domain.

Chapter 4 intends to illustrate our previous statement. We present some tools that are currently available. Each of them offers some correlation properties that will be presented in more details.

4 Existing tools

4.1 Generalities

By simply searching alert correlation tools on the Internet, we obtain a long list of names. Indeed, the 'correlation' word is used in a generic way (see Section 2.1), and many products use it as a commercial asset.

This section intends to present the main existing tools, in relation with general approaches presented in Section 3. Generally speaking, we notice that tools can be grouped into three categories:

- log analysis tools (Section 4.2.1)
- management consoles (Section 4.2.2)
- testbed research tools (Section 4..2.3)

These tools are listed in the next three subsections. Lists are not exhaustive, but we try to present the more representative solutions. Furthermore, a major part of these tools are commercial, and some proprietary patents prevent us from clearly understand which kinds of correlation approaches are implemented. There might be a gap between whitepapers and executable codes. Consequently, we invite the interested reader to download and test open-source solutions or free-trial versions, in order to strengthen his (or her) opinion.

4.2 Classification

4.2.1 Log analysis tools

4.2.1.1 Tools presentation

We present in table 1 some tools belonging to the so-called 'log analysis tools' category. The 'Current version' column provides the tool's version at the time of this writing. The 'Open source' column indicates that the tool can be freely downloaded and tested or not. We give the specific system requirements of each tool in the 'System Requirements' column. The 'tool maintenance' column helps determining if the tool is well maintained (based on tool website modifications and last tool updates).

Table 1: Log Analysis Tools Description

Name	Authors	Current Version	Open Source	System Requirements	Tool maintenance
SnortSnarf [SnSnf]	Silicon Defense	v021111.1	Yes	Perl support	Yes
WOTS [Wots]	Tony Curtis	v1.22	Yes	Perl support	Not really (stable version)
Swatch [Swatch]	Todd Atkins	v3.0.8	Yes	*nix OS, Perl support	Yes
Lire [Lire]	LogReport Foundation	v1.4	Yes	*nix OS, Perl support	Yes (project 2.0)
AWACS [Awacs]	Georg Greve	v0.1.1 (alpha version)	Yes	Linux (GNU)	No
XlogMaster [Xlogmst]	Georg Greve	v1.6.0	Yes	*nix OS, GTK+ v1.2	No
LogRep [Logrep]	ITEF!X Consulting	v1.4.2	Yes	Perl support	Yes
ZoneLog Analyzer [Zlana]	MCS	v1.18	Yes	Windows, Visual Basic 6 runtime files	Yes
SEC [Sec]	Reto Vaarandi	v2.1.11	Yes	Perl support	Yes
LogSurfer [Lsurf]	DFN-CERT	v1.5a	Yes	*nix OS C support	No
LogIDS [LogIds]	Adam Richard (SecurIT Informatique Inc.)	v2.0	Yes & No	Windows	Yes
ipLog [IpLog]	Ojnk Software Design	v2.2.3	Yes	*nix OS	No
LogWeaver [LgWeav]	J. Goubault-Larrecq : DICO Project	Evaluation version	Yes	Linux (with glibc)	Yes

First and foremost, this list is not exhaustive. We only present the most important solutions. We notice however that they in general are quite simple and that they require few system resources: most of them are built in Perl, and are open source. Finally, we would like to point out that many of them were not initially designed to deal with alert correlation specifically. This aspect is more widely developed in the following section.

4.2.1.2 Tools specificities

Table 2 details their correlation approaches, as well as their requirements, such as the input and output formats.

Table 2: Log Analysis Tool Specificities

Name	Input formats	Output formats	Correlation types	Remarks
SnortSnarf [SnSnf]	Snort alerts	HTML report	Filtering (colorizing): src/dst IPs, snort signatures	A simple tool to obtain Snort alerts summary. Its correlation capacities are limited.
WOTS [Wots]	Any file type	Reports and actions	Pattern matching rules (regular expressions)	Simple pattern matching through various file inputs
Swatch [Swatch]	Syslog files	Reports and actions	Pattern matching rules (regular expressions)	Similar to Wots but more restrictive on the input format
Lire [Lire]	Many log files	(X)HTML, PDF, Excel, charts, LogML	Filtering (given some schemas). Other approaches are possible.	Its xml-driven reporting engine allows a large variety of input formats. Very flexible. A good framework.
AWACS [Awacs]	Any (in theory)	Reporting	Not defined	This project seems abandoned since 2001.
XlogMaster [Xlogmst]	Any file type	Reporting and actions	Two-levels filtering rules (stateful/stateless)	More complex than Swatch. Filters can be applied at *display time (Xwindow)* or *read time*
LogRep [Logrep]	Many log files (snort, squid, syslog, etc…)	HTML reports and graphs	Not mentioned (pattern matching rules apparently)	There is little (if no) documentation on its working.
ZoneLog Analyzer [Zlana]	Log files generated by ZoneLabs' ZoneAlarm	DB visualization and report	Data mining rules	Limited to ZoneAlarm products. A friendly graphical interface.
SEC [Sec]	Any file type	Any (reports, scripts, programs, etc…)	Rule-based correlation (pattern matching rules)	Important effort to help building rich rules. Very promising.
LogSurfer [Lsurf]	Any text-based log files	Reports and actions	Pattern matching rules	Similar to primary versions of Swatch
LogIDS [LogIds]	Any ASCII files	Network map of events	Filtering with pattern matching rules and given a network model	Interesting tool for multi-IDS log monitoring
ipLog [IpLog]	Pcap data	Reports	Not defined. Very basic operations	A tcp traffic logger with poor correlation capacities
LogWeaver	Syslog files	Reports	Time Event rules	It can filter, count and match regular

[LgWeav]				expressions, but also detect correlations between events, while maintaining temporal relations.

Some observations result from table 2. First of all, most of these tools implement rule-based approaches. To be more precise, they exploit Perl pattern matching capacities to extract interesting information from the log files. The expert must write rules and introduce regular expressions before launching any correlation process.

We distinguish two sub-classes:

- Tools which were designed to fulfil a very specific goal: some listed tools were initially designed to deal with specific input formats. Some of them are designed for instance to help analyzing Snort IDS output (SnortSnarf). Others are designed to help analyzing firewall logs (ZoneLogs Analyzer) or Unix Syslog reports (Swatch). This makes them hard to be reused in other situations. They are often optimized for some given inputs and thus, have limited applications.
- Tools which were designed as general solutions: they often accept any kind of input formats and are very flexible. One interesting example is SEC (Simple Event Correlator). However, this flexibility implies some drawbacks: writing complex correlation rules can become a tedious work. Tools in this category can be compared by their rules *expression capacity.* We want to point out that even if many tools seem very similar (WOTS, SEC, Swatch, LogSurfer), they do not have the same facility to express rules. Intuitively, we would classify them as follows: E(LogSurfer) < E(Swatch) < E(WOTS) < E(SEC), with E(X) being the rules *expression capacity* of tool X.

Many of these tools were not initially designed to deal with alerts only. They can be very convenient for basic correlation operations. However, they may imply some difficult work in more complex correlation requirements, as they do not suit well for these tasks.

Finally, IDMEF is the current *alert* format standard. Even if none of these tools is specific to this input format, we can easily imagine adapting some of them to it. Along this line, two tools seem very promising: SEC and Lire [Sec, Lire].

SEC is a Perl application released around midyear 2000 by R. Vaarandi [Sec]. It provides an effective rule-based correlation engine that is simple to maintain. The product's author has published thorough technical descriptions, the results of rigorous testing of the product and he actively supports its current development [Vaar02]. There are nine basic operations that can be combined to create rules of arbitrary complexity. Furthermore, a time window can be specified for the alerts observation.

Lire is an Open Source reporting and analysis software maintained by LogReport Foundation. Its xml-driven reporting engine can theoretically parse any kind of log, given that a service driver is written to convert the raw log into a Lire DLF (Distilled Log Format) file. The engine performs calculation on these data which go into a report. This operation consists of two phases: analysis (extract derived information) and aggregation (statistics for instance). Furthermore, Lire allow an important choice of output formats.

4.2.2 Management consoles

In order to make IDSs suitable for corporate environments, the dispersed and various agents need to report to a central console. This console aims at gathering information in a single place and at analyzing it. We present in table 3 such central tools. The 'Description' column gives a short overview of the tools' characteristics. We do not give their exact system requirements, as they are in general quite important. Management consoles centralize information. Thus, it implies secure data transfers, secure central machines, secure data storage and so on. Consequently, we report the interested reader to the tools home page for more information.

Table 3: Management Consoles Description

Name	Authors	Open Source	Description
eTrust Network Forensics (formerly known as SilentRunner) [Etrust]	Computer Associates (previously Raytheon Company)	No	It is a network discovery, analysis and visualization tool for safeguarding some information assets.
TruThreat Risk Correlation [TruTh]	Arcsight Inc.	No	It combines the severity of potential threats and attacks with the value and vulnerability of business processes and assets to calculate and communicate the intrinsic risk of a particular security event [Annex A].
ACID [Acid]	R. Danyliw (CERT)	Yes	It is a PHP-based analysis engine to search and process a database of security events generated by Snort, iptables, ipchains, ipfw, tcpdump, etc..
Dragon Server [DragSer]	Enterasys Networks	No	It is similar to ACID but dedicated to Dragon deployments.
IPFC [Ipfc]	Conostix	Yes	It is a simple tool to manage and monitor multiple types of security modules such as packet filters (netfilter, pf, ipfw, IP Filter, checkpoint FW1...), NIDS (Snort, ISS RealSecure...), webservers (from IIS to Apache), etc…
Lightning Console [LighCon]	Tenable Security	No	It includes vulnerability scanning; reporting and scheduling; asset management; real time aggregation of IDS events and correlation with vulnerabilities.
Prelude [Prelu]	Y. Vandoorselaere	Yes	It adds support to Prelude IDS of Vulnerability correlation with Nessus Report.
Tivoli Risk Manager [TivoMa]	IBM	No	It manages security incidents from a single web-based security console by correlating security information and risk alerts from firewalls, routers, networks, host- and application-based intrusion detection systems, desktops, and vulnerability-scanning tools. It may include IBM Tivoli Enterprise Console and IBM Tivoli NetView to provide the capability to drill down to the network topology to see where the affected resources are located and delivers root cause problem determination.
PureSecure [PureSec]	Demarc Security	No	It is a commercial product similar to ACID, but with more functionalities and flexibility.

SnortCenter [SnoCen]	S. Dens	Yes	It is a web-based client-server management system written to help configuring Snort and querying the alert database.
Real Secure Site Protector [RealSec]	Internet Security Systems (ISS)	No	It is a centralized security management for all ISS products (Proventia, RealSecure, etc).
Net Forensics Console [NetFor]	NetForensics	No	It collects, analyzes, and correlates security device information from across the network in a series of four distinct phases: Normalization, Aggregation, Correlation, and Visualization.
neuSecure [NeuSec]	GuardedNet	No	It is a security management and incident response platform implementing interesting correlation features (see below).
IDSCenter [IdsCen]	Engage Security U. Kistler	Yes	It is another Snort front-end to walk through the alert database and display reports.
LogIDS [LogIds]	Adam Richard (SecurIT Informatique Inc.)	Yes & No	It is a simple but interesting tool for multi-IDS log monitoring: a limited open-source version and a commercial version.
InCharge [InCha]	SMARTS	No	It is a complex management suite, including the Service Assurance Manager. It integrates and correlates management information, including infrastructure topology, polled data, events, and authentic problems from various sources spanning network, system, application, and business objects, seamlessly integrating with customer relationship management (CRM), provisioning, and other OSS systems.
Shadow [Shad]	US Navy	Yes	It performs traffic analysis; a sensor collects packet headers from all IP packets that it sees; the analyzer examines the collected data and displays user defined "interesting" events on a web page. More an IDS than a correlation tool…

Table 3 observation leads to the following remark: there is a larger number of commercial products than in the previous category. Actually, most companies which product IDS sensors design their own central monitoring tools to offer to their customers a complete suite (Computer Associates, IBM, ISS, GuardedNet, etc). However, there are some noticeable exceptions. Indeed, some IDSs are open-source (Snort or Bro for instance) and require a central monitoring console as well. Thus, some open source consoles are designed to fulfill this requirement: ACID, SnortCenter, IDSCenter, etc…

We do not give in table 3 the tools' requirements. Actually, they often need particular setups at their installation phase (databases, secure communication, sensors' configurations…). Thus, they require a more complex setup before being put into service than log analysis tools. Table 4 presents the correlation characteristics they implement. Some descriptions are shortened for lack of available information on these tools.

Table 4: Management Consoles Characteristics

Name	Correlation characteristics
eTrust Network Forensics (formerly known as SilentRunner) [Etrust]	Clustering based on database queries. Visualization.
TruThreat Risk Correlation [TruTh]	Correlation based on three factors: (Real time events from heterogeneous devices, Results of vulnerability scans and other sources of threat data, Value of the host, database or application to the organization). The correlation system combines the severity of potential threats and attacks with the value and vulnerability of business processes and assets to calculate and the intrinsic risk of a particular security event. In addition, it provides a very promising alert monitoring interface.
ACID [Acid]	Basic correlation operations (filtering, etc..) on a database. Visualization.
Dragon Server [DragSer]	Correlation based on database queries (data mining) and vulnerabilities reports (thanks to a vulnerability scanner).
IPFC [Ipfc]	Correlation based on data mining queries (data mining).
Lightning Console [LighCon]	Correlation based on the network topology (vulnerability scanners, traceroute results): Filtering and data mining queries.
Prelude [Prelu]	Selective suppression, based on Nessus report for instance, aggregation.
Tivoli Risk Manager [TivoMa]	aggregation, clustering, prioritization, state-based correlation [Deb01].
PureSecure [PureSec]	Simple correlation operations based on database queries.
SnortCenter [SnoCen]	Simple correlation operations based on database queries.
Real Secure Site Protector [RealSec]	Not clear. Alert aggregation and prioritization. Filtering.
Net Forensics Console [NetFor]	A rule-based correlation (explicit time-aware security policy rules) and a statistical correlation (event categorization and threat scoring to determine the threat potential of security-based anomalies).
neuSecure [NeuSec]	A multi-phased alert correlation: the first is based on vulnerabilities (impact correlation), The second tries to detect unknown attacks and anomalous behavior (statistical correlation). The last one enforces security policies (rule-based correlation). It is called Deterministic Threat Analysis (similar to Arcsight)

IDSCenter [IdsCen]	Simple correlation operations based on database queries.
LogIDS [LogIds]	Simple correlation operations. Alerts are grouped per hosts, and rules can be defined to display them.
InCharge [InCha]	A complex correlation process based on the codebook approach, the state-based approach (root cause maps), and some user-defined rules (filtering), etc…
Shadow [Shad]	Statistics and clustering. Simple operations.

We have presented some management consoles in tables 3 and 4. The list is not complete, but it gives a good understanding of their correlation capacities. Some of them are simple interfaces to query back-end databases. Others complement alert information with logs coming from vulnerability scanners, anti-virus or any other security devices. Thus, with regard to these various characteristics, we distinguish two categories:

- *Simple Consoles*: they are the simplest management console solutions. They generally consist in a database interface and they permit simple data reports. They include ACID, Prelude, IPFC, IDSCenter, LogIDS, PureSecure, etc. In short, this category contains almost all open source solutions plus the simplest commercial ones.
- *Specific consoles*: Most Network-based IDSs (NIDS) vendors produce a central console for their own products. We list some of them in table 4, for instance ISS, Enterasys Networks, IBM, etc. These tools are often opaque and little information is available. In general they are just one part of a complex security suite. This implies *de facto* a complex installation, a long maintenance formation and huge costs.

In usual cases, efforts are made on the interface. Almost all visited sites show proudly the graphical aspects of their tools (systematic *screenshots* sections) and the convenient manners to launch information requests. However, technical details are quite disappointing. This seems all the more surprising that implemented approaches are correlation key points. If we have a deeper look at the correlation approaches they implement, we realize that they are limited to:

- data mining requests
- basic operations: counting , filtering, aggregation

Some of them look for additional information. One solution consists in using results from vulnerability scanners (it is the case of eTrust Network Forensics, Prelude, etc) or network management requests (it is the case of Tivoli Risk manager in association with IBM Netview, InCharge, etc). But very few implement original correlation approaches. Exceptions are:

- InCharge from SMARTS: It implements the codebook approach described in [Klig95] (among other methods)
- TruThreat Risk Correlation from Arcsight: They use a wide range of information to calculate the intrinsic risks. Furthermore, interesting rule matching algorithms are implemented. This makes the monitoring interface very complete.
- NetForensics Console from netForensics: Beside the traditional rule-based approaches, they implement a statistical approach that complements the first one. They build some attributes profiles and they check that incoming events adopt these profiles. Anomalous profiles are thus detected and reported (like some IDSs do). However, they do not precise how attributes are chosen in their product whitepaper.

4.2.3 Experimental tools

We group in this category tools which are designed to validate research ideas. They are not mature enough to be used in a production system, but this can change in a near future. At the time writing, there are more whitepapers applications than applicable tools.

Some tools that belong to this category are presented below:

- *Intrusion Alert Correlator (version 0.2)* developed by the North Carolina State University. It implements the approach presented in section 3.3.3. It identifies the pre-requisite (e.g. existence of vulnerable services) and the consequence of each type of attacks, and correlates the corresponding alerts by matching the consequence of some previous alerts and the prerequisite of some later ones. A resulting *hyper-alert graph* is displayed to show the structure of series of attacks. More practically, the tool is currently able to process alerts generated by ISS RealSecure network Sensor 6.0. It is written in java and can be downloaded from [Iac02].

- *AlertSTAT* (version 2.0) developed by the Department of Computer Science in Santa Barbara. It is a STAT-based analyzer (see section 3.7.2). AlertStat extends xStat with an IDMEF extension module, an IDMEF event provider, and a number of scenarios (3 are furnished with the downloadable version). The alerts processed by the vent provider must be in the standard format defined by the Intrusion Detection Working Group (RFC 2026).

- *Threatman*: It is an IDMEF compliant threat manager application which makes use of a multi-tier architecture. It aims at correlating event and vulnerability with alerts sent by IDSs, firewalls and other IDMEF compliant applications. The tool is in pre-alpha development phase [Threat].

- *OSSIM*: it claims to unify network monitoring, security, correlation and qualification in one single tool. It is a complex mix of all Snort, Acid, mrtg, NTOP, OpenNMS, nmap, nessus and rrdtool. The 0-7 beta-version has been released in November 2003 [Ossim]. The tool whitepaper describes two correlation methods:
    - *Correlation using sequences of events:* It uses if-then rules defined within a xml-configuration file in order to detect event sequences.
    - *Correlation using heuristic algorithm:* It is based on CALM (Compromise and Attack Level Monitor), an assessment algorithm that uses event accumulation over time. Little information is given on the algorithm implementation (how *risk levels* are determined for instance).

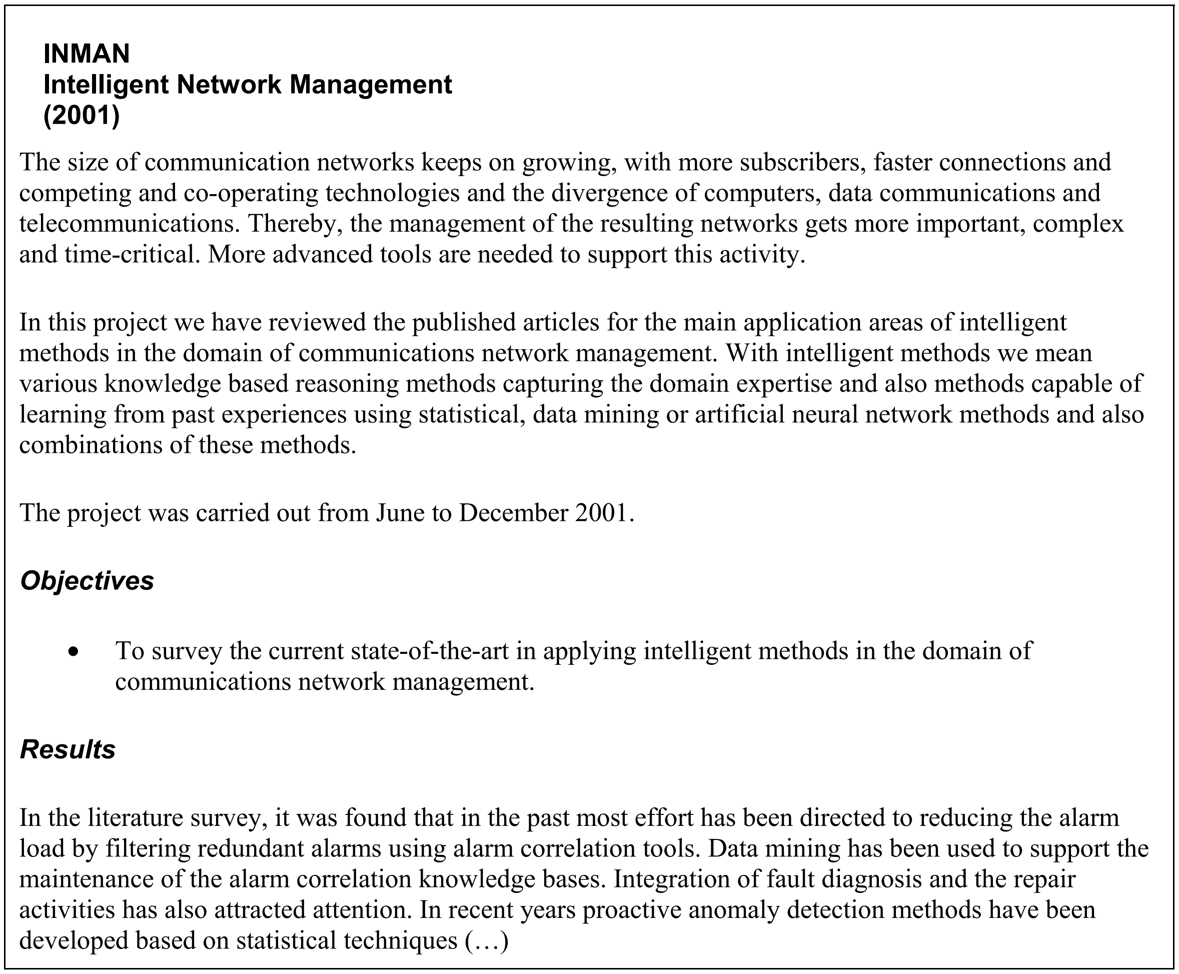
- *MACE – Meta Alert Correlation Engine*: It is built upon the CLIPS expert system (see Section 3.2.3), creating a distributed application that is capable of filtering and correlating Intrusion Detection alerts into Meta-alerts from multiple sources [Mace]. Moreover, it uses IDMEF to represent alerts and meta-alerts. Finally, two remote modules are particularly interesting:
    - *ARPMonitor:* it sniffs ARP traffic in promiscuous mode. MAC/IP address pairs are stored in a given database. Each time a new or changed MAC address appears, the tool generates an IDMEF alert to MACE.
    - *Bandwidth Monitor:* it measures the bandwidth on the network, split out per source/destination port or IP address. Values are statistically analyzed to detect anomalous bandwidth activity. In this last case, an IDMEF alert is sent to MACE for further processing.

  However, there is no explanation in [Rieb03] reagarding how these additional alerts are correlated with the others.

4.2.4 Miscellaneous

We have presented so far many tools that present correlation capacities. Some of them are based on pattern matching techniques, others on data mining approaches and few on more original methods. This leads to two major remarks:

First, the alert correlation domain is not mature enough. Many techniques exist, and only a few of them are currently implemented in practical tools. Furthermore, we observe that the Network Management domain is still very active. An illustration of this is shown in figure 9. It is a summary of INMAN, a Network Management Project (Intelligent Network Management) that was carried out from June to December 2001. This summary was found on the Project home page:


INMAN
Intelligent Network Management
(2001)

The size of communication networks keeps on growing, with more subscribers, faster connections and competing and co-operating technologies and the divergence of computers, data communications and telecommunications. Thereby, the management of the resulting networks gets more important, complex and time-critical. More advanced tools are needed to support this activity.

In this project we have reviewed the published articles for the main application areas of intelligent methods in the domain of communications network management. With intelligent methods we mean various knowledge based reasoning methods capturing the domain expertise and also methods capable of learning from past experiences using statistical, data mining or artificial neural network methods and also combinations of these methods.

The project was carried out from June to December 2001.

Objectives

- To survey the current state-of-the-art in applying intelligent methods in the domain of communications network management.

Results

In the literature survey, it was found that in the past most effort has been directed to reducing the alarm load by filtering redundant alarms using alarm correlation tools. Data mining has been used to support the maintenance of the alarm correlation knowledge bases. Integration of fault diagnosis and the repair activities has also attracted attention. In recent years proactive anomaly detection methods have been developed based on statistical techniques (…)

Figure 9: correlation operators, *Selective Suppression*

Secondly, there is a prevalent trend to adapt Event Correlation Techniques to alerts. For instance, this is the case of MACE, which uses CLIPS as backbone, or chronicles approach described in [Morin03]. Otherwise, simplest tools are basic database interfaces or text parsers and commercial tools implement complex and fuzzy solutions that need to be evaluated. As a result, we wonder if the correlation problem has well been stated. Instead of applying traditional methods blindly, it would be good to first define correlation requirements properly. This is quite a paradox with our first remark. However, we are convinced that a major prerequisite consists in initially stating correlation objectives. After such an in-depth analysis, technical methods can be chosen. This is an *a posteriori* approach, by opposition to *a priori* approaches which seem currently applied.

5 Conclusion

Intrusion Detection Systems are parts of the technologies employed in keeping computers and their data secure. These systems are powerful solutions for detecting suspicious or anomalous behavior. They are often defined as a second line of defense. However, security professionals have a tough challenge coping with the vast amounts of alerts (and mainly false positives) that are produced by IDSs. Alerts often come from multiple sensors, spanning multiple complex subsystems. There may also be data from multiple brands or types of IDSs, as well as from other tools. This complexity implies that such systems require constant monitoring and maintenance. Human capacities are not sufficient, and the 'alert correlation' field tries to address these issues.

In this document, we have provided a survey of all the proposed techniques. They are not only from the ID community but also by the Network Management community, which has tried to solve similar problems for many years. Descriptions in Section 3 show that many approaches currently exist. They come from various research domains and have proven their efficiency in many cases. However, Section 4 leads to a deceiving observation: among tools and products that have been proposed so far in Alert Correlation, very few implement such approaches. Most of them are limited to down-to-earth, pragmatic techniques, such as pattern matching or database queries.

As a consequence, the Alert Correlation field is far from being totally beaten track. We believe that on one hand, products are going to use more and more approaches that currently exist in Network Management. Thus, alert correlation research will widen out to other techniques than scenario-based solutions. We do not say that these solutions are useless, but we believe that they only cover one part of the alert correlation issues. On the other hand, we claim that solutions must be thought from precise issues. Instead of applying naively fashionable techniques, designers must first analyze specific correlation requirements.

6 Bibliography

[Abra01] A. Abraham. *"Neuro Fuzzy Systems: State-of-the-art Modeling Techniques"*. School of Computing & Information Technology Monash University, Australia. 2001.

[Acid] *ACID , Analysis Console for Databases*, by R. Danyliw, home page: http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html

[Agraw94] R. Agrawal, R. Srikant. *"Fast Algorithms for Mining Asssociation rules"*. In Proceedings of the 20th Int'l Conference on Very Large Databases, Santiago, Chile. Sep. 1994.

[Agraw95] R. Agrawal, R. Srikant. *"Mining Sequential Patterns"*. In Proceedings of the 11th International Conference on Data Engineering. Taiwan, 1995.

[All00] J. Allen. *"State of the Practice of Intrusion Detection Technologies"*. Technical Report, Carnegie Mellon Software Eng. Institute, Jan. 2000.

[Argus] Honeywell Technology Center. Argus Project home page: http://www.htc.honeywell.com/projects/argus/

[Asax92] N. Habra, B. Le Charlier, A. Monji, I. Mathieu. *"ASAX: Software Architecture and Rule-Based Language for Universal Audit Trail Analysis"*. In the Proceedings of ESORICS'92, Toulouse, Nov. 1992.

[Awacs] *AWACS Project*, from G. Greve, available at: http://www.gnu.org/software/awacs/

[Bagc03] S. Bagchi. *"Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS"*. Paper 127 to ACSAC 2003.

[Bap94] S. Bapat. *"Object-Oriented Networks: Models for Architecture, Operations and Management"*. PTR Prentice Hall, NJ, USA, 1994.

[Bayer02] R. bayer, G. Carls, B. Frohnhoff. *"Divide et Impera – A Layered Appraoch to Rule based Alarm Correlation for SDH networks"*. Apr 2002.

[Berth03] M. Berthold, D.J. Hand. *"Intelligent Data Analysis: An Introduction"*. Second Edition, Springer Eds. 2003.

[Bish95] M. Bishop. *"A standard audit trail format"*. Technical report, department of Computer Science, University of California Davis, 1995.

[Biv02] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, M. Embrechts. *"Network-based Intrusion Detection using Neural Networks"*. NY, US. 2002

[Board01] B. Boardman. *"Peregrine perches atop the pack"*. Network and Systems Management. *2001.*

[Caml03] S. Cheung, U. Lindquist, M. Fong. *"Modeling Multistep Cyber Attacks for Scenario Recognition"*. SRI International presentation, April 2003. Available on line at: http://www.iaands.org/discex3/DISCEX3-AsPresented/25-Cheung-SRI.ppt

[Carey02] N. Carey, A. Clark, G.M. Mohay. *"IDS Interoperability and Correlation using IDMEF and Commodity Systems"*. ICICS 2002, p. 252-264. 2002.

[Casl98] Secure Networks. *"Custom Attack Simulation Language (CASL)"*. January 1998.

[Casey02] H.R. Casey. *"The Simple Event Monitor, A Tool for Network Management"*. Master thesis, University of Colorado. 2002.

[Cids] S. Bagchi. *"Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS"*. Purdue University. Submitted to ACSAC 2003.

[Cisl99] R. Feiertag, C. Kahn, P. Porras, D. Schnackenberg, S. Staniford-Chen, B. Tung. *"A common intrusion specification language (cisl)"*. Specification draft from the Gidos Project.

[Cronk88] R. Cronk, P. Callahan, L. Bernstein. *"Rule-based expert systems for network management and operations: an introduction"*. IEEE Networks, 2(5):7-21, 1988.

[Cui02] Y. Cui. *"A toolkit for intrusion alerts correlation based on prerequisites and consequences of attacks"*. PhD Thesis from the North Carolina State University. Raleigh, US. Dec. 2002.

[CupMie02] F. Cuppens, A. Miege. *"Alert Correlation in a Cooperative Intrusion Detection Framework"*. IEEE Symposium on Security and Privacy, 2002.

[CupOrt00] F. Cuppens, R. Ortalo. *"Lambda: a language to model a database for detection of attacks"*. In Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000), Oct. 2000.

[Cuppens01] F. Cuppens. *"managing alerts in a multi-intrusion detection environment"*. In Proceedings of the 17th Annual Computer Security Applications Conference, (ACSAC'01). Dec. 2001.

[Cuppens02] F. Cuppens, F. Autrel, A. Miege, S. Benferhat. *"Correlation in an intrusion detection process"*. Sécurité des Communications sur Internet (SECI'02). Sept. 2002.

[DainCunn] O. Dain, R.K. Cunningham. *"Fusing a heterogeneous alert stream into scenarios"*. In proceedings of the 2001 ACM Workshop on Data Mining for Security Applications, p. 1-13, Nov. 2001.

[Deb01] H. Debar, A. Wespi. *"Aggregation and correlation of intrusion-detection alerts"*. In Recent Advances in Intrusion Detection, LNCS 2212, p. 85-103, 2001.

[Deb92] H. Debar, M. becker, D. Siboni. *"A Neural Network Component for an Intrusion Detection System"*. In the Proceedings of the IEEE Symposium on Research in Computer Security and Privacy. Oakland, CA. May 1992.

[Derai99] R. Deraison. *The Nessus attack scripting language reference guide.* Available on line at: http://www/nessus.org

[Dough02] K.L. Dougherty, J. Edwards. *"Sim[le vs. Absolute m*

[Douss93] C. Dousson, P. Gaborit, M. Ghallab. *"Situation recognition: Representation and Algorithms"*. In Proceedings of the 13th IJCAI, pp166-172, Aug. 1993.

[Douss94] C. Dousson. *"Suivi d'évolutions et reconnaissance de chroniques"*. PhD Thesis 1994, available at: http://dli.rd.francetelecom.fr/abc/diagnostic/

[DragSer] *Dragon Server* products by Enterasys Networks, home page: http://www.enterasyspartner.com/

[Dubo92] D. Dubois, J. Lang, H. Prade. *"Possibilistic logic"*. Handbook of Logic in Artificial intelligence and Logic Programming, volume 3: Nonmonotonic Reasoning and uncertainty Reasoning. 1992.

[DuSha03] X. Du, M. Shayman, R.A. Skoog. *"Using Neural networks in distributed Management to identify Control and Management to identify Control and management plane poison messages"*. University of Marynland, MD, US. Research supported by DARPA. 2003.

[Gard98] R.D. Gardner, D.A. harle. *"Pattern discovery and specification techniques for alarm correlation"*. IEEE Network Operations and Management Symphosium, 3:713-722. Feb. 1998.

[EckVig00] S.T. Eckmann, G. Vigna, R.A. Kemmerer. *"Statl: An attack language for state-based intrusion detection"*. In proceedings of the ACM Workshop on Intrusion Detection, Nov. 2000.

[Elvin] Elvin Product, from DSCT Inc. home page: http://elvin.dstc.com/

[Etrust] *eTrust Network Forensics* from Computer Associates (formerly known as SilentRunner, by Raytheon): http://ca.com/acq/silentrunner/

[Forgy82] C. Forgy. *"RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem"*. Artificial Intelligence, 19, pp.17-37, 1982.

[Fyo97] Fyodor. *"The Art of Port Scanning"*. Phrack magazine, Vol. 7, Issue 51 Home. Dec. 1997. Tool home page: http://www.insecure.org/nmap/

[Garof99] M.N. Garofalakis, R. Rastogi, K. Shim. *"SPIRIT: Sequential Pattern Mining with Regular Expression Constraints"*. The VLBD Conference, Scotland, UK. 1999.

[Good91] R.M. Goodman, H. Latin. *"Automated knowledge acquisition from network management databases"*. In IFIP International Symphosium on Integrated Network Management (ISINM'91), p. 541-549, 1991.

[Haines03] J. Haines, D.K. Ryder, L. Tinnel, S. Taylor. *"Validation of Sensor Alert Correlators"*. IEEE Computer Society, p. 46-57. 2003

[Ham50] R.W. Hamming. *"Error detecting and error correcting codes"*. Bell System Tech. J., 29:147-160, Apr. 1950.

[Heg99] H. Hegering, S. Abeck, B. Neumair. *"Integrated Management of Networked Systems"*, p.326-335. Morgan Kaufman, 1999.

[Holl86] J. H. Holland, K. J. Holyoak, R. E. Nisbett, P.R. Thagard. *"Induction: Processes of Inference, Learning and Discovery"*. MIT, USA, 1986.

[Hood97] C.S. Hood, C. Ji. *"Proactive network fault detection"*. Proceedings of INFOCOM 1997, 1997.

[Houck95] K. Houck, S. Calo, A. Finkel. *"Towards a practical alarm correlation system"*. In IFIP/IEEE International Symphosium on Integrated network management (ISINM'95), p. 226-237, 1995.

[Idmef] H. Debar, D. Curry. *"Intrusion Detection Message Exchange Format: Extensible markup Language (XML) Doument Type"*. IETF Draft 10, expired July 2003. URL:http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt

[IdsCen] *IdsCenter*, written by U. Kistler for Engage Security. Home page: http://www.engagesecurity.com/products/idscenter/

[Idxp] The IETF Intrusion Detection Exchange Protocol (IDXP) draft (v0.7), available at: http://www.ietf.org/internet-drafts/

[Ietf00] IETF Presentation, *"Event Forwarding Discriminator Behavior"*. In Proceedings of the 47th IETF Workshop, Marc 2000. Available at: http://www.ietf.org/proceedings/00mar/slides/tmnsnmp-bof-00mar/tsld023.htm

[Ilgun95] K. Ilgun, R. Kemmerer, P. Porras. *"State Transition Analysis: A Rule-Based Intrusion Detection System"*. IEEE Transactions on Software Engineering, 21(3):181-199, March 1995.

[InCha] *InCharge Suite* from SMARTS Inc. home page: http://www.smarts.com/

[Inman] *INMAN, Intelligent Network Management Project* from VTT Information Technology. Home page: http://www.vtt.fi/tte/projects/inman/inman.html

[Ipfc] *IPFC Inter-Protocol Flexible Control* from Conostix S.A., home page: http://www.conostix.com/ipfc/

[IpLog] *ipLog* tool from Ojnk Software Design, available at: http://stat.iplog.md/

[Jacob00] V. Jacobson, C. Leres, S. McCanne. *Tcpdump 3.5 documentation*. Available on line : http://www.tcpdump.org

[Jakob93] G. Jakobson, M.D. Weissmann. *"Alarm Correlation"*. IEEE Network, p.52-59. Nov. 1993.

[JakWeis95] G. Jakobson, M. Weissman. *"Real-time telecommunication network management: extending event correlation with temporal constraints"*. In IFIP/IEEE International Symphosium on Integrated Network Management, IV, page 290-301. 1995.

[JakWeBre] G. jakobson, M. Weissman, L. Brenner, C. Lafond, C. Matheus. *"GRACE: Building Next Generation Event Correlation Services"*. GTE Laboratories Incorporated, USA, IEEE, 0-7803-5864-3, p.701-714, 2000.

[Julisch00] K. Julisch. *"Dealing with False Positives in Intrusion Detection"*. In Extended abstract at the 3rd International Symposium on Recent Advances in Intrusion Detection (RAID'00). 2000.

[Julisch02] K. Julisch, M. dacier. *"Mining Intrusion detection Alarms for Actionable Knowledge"*. In Proceedings of SIGKDD'02, Canada. 2002.

[Julisch03] K. Julisch. *"Using Root Cause Analysis to handle Intrusion Detection Alarms"*. Dissertation, PhD Thesis. Dortmund, 2003.

[Katz95] I. Katzela, M. Schwartz. *"Schemes for fault identification in communication networks"*. IEEE Transactions on Networking, 1995.

[Klem94] P. Klement, W. Slany. *"Fuzzy Logic in Artificial Intelligence"*. CD-Technical Report 94/67 from Technische Universitat of Wien, Austria. June 1994.

[Klig95] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo. "I*ntegrated Network Management"*. 1995.

[Klir94] G. J. Klir, "*Measures of Uncertainty in Dempster-Shafer Theory of Evidence*," in R. R. Yager, J. Kacprzyk, and M. Fedrizzi, editors, Advances in the Dempster-Shafer Theory of Evidence, Wiley, pp. 3549, 1994.

[Kot02] P. Kothari. *"Intrusion Detection Interoperability and Standardization"*. Feb. 2002, available on line at: http://rr.sans.org/intrusion/interop.php

[Krueg03] C. Kruegel, T. Toth. *"using Decision Trees to Improve Signature-based Intrusion Detection"*. In 6th Symphosium on Recent Advances in Intrusion Detection (RAID'03). Springer Verlag, USA, 2003.

[Kumar94] S. Kumar, J. A. Meech. *"A Hypermanual on Expert Systems"*. Canada Center for Mineral and Energy Technology, 1994.

[Kumar95] S. Kumar, E.H. Spafford. *"A software architectureto support misuse intrusion detection"*. Technical Report CSD-TR-95-009, The COAST Project Department of Computer Sciences, Purdue University, 1995.

[Lang02] L. Langfeldt. *"decision-making in expert panels evaluating research: Constraints, Processes and bias"*. PhD Thesis, Oslo. 2002.

[Laz92] A. A. Lazar, W. Wang, R. H. Deng. *"Models and algorithms for network fault detection and identification: A review"*. In IEEE International Conference on Communications'92 (ICC92), p. 999-1003. Nov. 1992.

[Lew95] L. Lewis. *"Managing Computer Networks: A Case Based reasoning Approach"*. Artech House: Norwood, MA, 1995.

[Lew99] L. Lewis. *"Service level Agreements for Enterprise Networks",* p. 158-190. Artech House: Norwood, MA, 1999.

[LgWeav] LogWeaver from J. Goubault-Larrecq (DICO Project), available at: http://www.lsv.ens-cachan.fr/~goubault/DICO

[LighCon] *Tenable Lightning Console*, from Tenable Security, home page: http://www.tenablesecurity.com/console.html

[Lippm99] R. Lippmann. *"Using key String and Neural networks to Reduce False Alarms and Detect new Attacks with Sniffer-Based Intrusion Detection Systems"*. In Proceedings of the 2nd Int'l Workshop on the Recent Advances in Intrusion Detection (RAID'99). Oct. 1999.

[Lire] *LIRE* from LogReport Foundation, available at: http://logreport.org/

[Llsim03] J.W. Haines, S.A. Goulet, R.S. Durst, T.G. Champion. *"LLSIM: Network Simulation for Correlation and Response Testing"*. In Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03), IEEE. 2003.

[LoChen00] C. Lo, S. Chen, B. Lin. *"Coding based schemes for fault identification in communication networks"*. J. Network Mgmt; 10:157-164. 2000.

[LogIds] *LogIDS 2.0* from Securit Informatique Inc., available at: http://iquebec.ifrance.com/securit/download.html

[Logrep] *LogRep* tool from ITEF!X Consulting, available at: http://logrep.sourceforge.net/

[Lsurf] *LogSurfer log analysis tool* from DFN-CERT, available at: http://www.cert.dfn.de/eng/logsurf/

[Luna94] H.P. Luna *"Sistemas de apoio a desiciao"*. Brazil, 1994.

[Mace] *MACE, Meta-Alert Correlation Engine Project* home page: http://mace-project.sourceforge.net/

[MaKnig] J. Ma, B. Knight. *"A reified Temporal Logic: An overview"*. University of Greenwich. Artificial Intelligence Review, pp.189-217. 2001.

[Mang00] S. manganaris, M. Christensen, D. Zerkle, K. Hermiz. *"A Data Mining Analysis of RTID Alarms"*. Computer Networks, 34(4), Oct. 2000.

[Manni95] H. Mannila, H. Toivenen, A.I. Verkamo. *"Discovering frequent episodes in sequences"*. KDD-95, p. 210-215. Aug. 1995.

[Mé98] L. Mé. *"Gassata: a genetic algorithm as an alternative tool for security audit trail analysis"*. In Proceedings of the First international workshop on the Recent Advances in Intrusion Detection (RAID'98), 1998.

[Meira97] D. M. Meira. *"A Model For Alarm Correlation In Telecommunications Networks"*. PhD Thesis, University of Minas Gerais, Brazil. Nov. 1997.

[Michal83] R.S. Michalski, J.G. Carbonell, T.M. Mitchell. *"Machine learning: An Artificial Intelligence Approach"*. Springer-Verlag, Germany, 1983.

[MicMe01] C. Michel, L. Mé. *"ADeLe: An Attack Description Language for Knowledge-Based Intrusion Detection"*. In Proceedings of the 16th International Conference on Information Security (IFIP Sec.). June 2001.

[Mindb] *MindBox* from ART Enterprise Technologies, web page: http://www.mindbox.com/home.html

[Mira87] D. P. Miranker, "TREAT: A better match algorithm for AI production systems", In *Proceedings of the Sixth National Conference on Artificial Intelligence* (AAAI-87), pages 42-47, August 1987.

[Morin02] B. Morin, H. Debar, L. Me, M. Ducasse. *"A Formal Data Model for IDS Alert Correlation"*. In Proceedings of the 5th Int'l Symposium on Recent Advances in Intrusion Detection (RAID'02), Oct. 2002.

[Morin03] B. Morin, H. Debar. *"Correlation of Intrusion Symptoms: an Application of Chronicles"*. In Proceedings of the 6th Int'l Symposium on Recent Advances in Intrusion Detection (RAID'03), Sept. 2003.

[NetFor] *NetForensics Console*, from NetForensics Inc. home page: http://www.netforensics.com/

[Neum99] P.G. Neumann, P.A. Porras. *"Experience with EMERALD to Date". In Proceedings of the 1st USENIX Workshop on Intrusion Detection and network Monitoring, CA-US. Apr. 1999.*

[NeuSec] *NeuSecure* from GuardedNet, home page: http://www.guarded.net/

[NgLak98] R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang. *"Exploratory Mining and Pruning Optimizations of Constrained Association Rules"*. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Washington, US. June 1998.

[Nils80] N. J. Nilsson. *"Principles of artificial intelligence"*. Palo Alto, US, 1980.

[NingX02] P. Ning, D. Xu. *"Learning Attack Strategies from Intrusion Alerts"*. Dept. of Computer Science, North Carolina, US. 2002.

[Ning02] P. Ning, Y. Cui, D.S. Reeves. *"Constructing Attacks Scenarios through Correlation of Intrusion Alerts"*. CCS'02, Washington DC, US. 2002.

[Ning03] P. Ning, D. Zu. *"Adapting Query Optimization techniques for Efficient Intrusion Alert Correlation". In* Proceedings of the 17th IFIP WG 11.3 Working Conference on Data and Application Security*, August, 2003.*

[Ning04] P. Ning, D. Xu, C. G. Healey, R. St. Amant. *"Building Attack Scenarios through Integration of Complementary Alert Correlation Methods"*. To appear in the 11th Annual Network and Distributed System Security Symposium (NDSSS'04). Feb. 2004.

[Nyg95] Y.A. Nygate. *"Event correlation using rule and object based techniques"*. Chapman and Hall, p. 278-289, USA, 1995.

[Open01] OpenService Inc., *"Event Correlation: The enabler of Active Internet Security Management"*. White paper available at: http://www.open.com

[Ossim] *OSSIM Project* home page: http://www.ossim.net/

[Paxs98] V. Paxson. *"Bro: A system for detecting network intruders in real-time"*. In Proceedings of the 7th Usenix Security Symphosium, Jan. 1998.

[PerrMe] N. Perrot, L. Me, G. Trystram, J.M. Trichard, M. Decloux. *"An hybrid approach based on fuzzy logic and genetic algorithms to control a crossflow microfiltration pilot plant"*. INRA and SUPELEC, France.

[Porr02] P.A. Porras, M.W. Fong, A. Valdes. *"A Mission-Impact-based Approach to INFOSEC Alarm Correlation"*. Supported by DARPA, SRI International Paper, 2002.

[Porr97] P. A. Porras, P.G. Neumann. *"EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances"*. Computer Science Laboratory, SRI International, 1997.

[Prelu] L. Oudot. *"Tools to make correlation in Prelude-ids"*, home page: http://www.rstack.org/oudot/prelude/correlation/

[PureSec] *Demarc PureSecure* from Demarc Security, home page: http://software.freshmeat.net/projects/demarc/

[RealSec] *Real Secure Site Protector*, from Internet Security Systems (ISS), home page: http://www.iss.net/

[Rieb03] M. R. Rieback. *"The Meta-Alert Correlation Engine"*. MSc. Thesis Report from Delft University of Technology, NL. July 2003.

[Rivest] R. L. Rivest. *S-Expressions –SEXP tutorial*, MIT, available on line at: http://theory.lcs.mit.edu/~rivest/sexp.html

[Roes99] M. Roesch. *"Snort- lightweight intrusion detection for networks"*. In Proceedings of the Usenix Lisa'99 conference. Nov. 1999.

[Rog01] M. Roger, J. Goubault-Larrecq. *"Log auditing through Model-Checking"*. In Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01), Canada, June 2001.

[Sabin97] M. Sabin, R.D. Russel, E.C. Freuder. *"Generating diagnostic tools for network fault management"*. In Integrated Network Management V, pp 700-711. 1997.

[Sec] *Simple Event Correlator (SEC)* from R. Vaarandi, available at: http://www.estpak.ee/~risto/sec/

[Smarts] System Management ARTS Inc. Codebook vs. rulesbased event correlation: Automated root cause and effects analysis of network problems. Technical Report, System Management ARTS Inc., Dec 2000.

[Smets88] P. Smets, A. Mamdani, D. Dubois, H. Prade. *"Non-Standard Logics for Automated Reasoning"*. Academic Press, England, 1988.

[SnSnf] *SnortSnarf*, a product from Silicon Defense available at: http://www.silicondefense.com/software/snortsnarf/

[SnoCen] *SnortCenter*, written by S. Dens, home page: http://users.pandora.be/larc/

[Shad] *NSWC Shadow*, from the Naval Surface Warfare Center (US Navy) home page: http://www.nswc.navy.mil/ISSEC/CID/

[Srik96] R. Srikant, R. Agrawal. *"Mining Sequential patterns: Generalizations and performance Improvements"*. In Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96). France, 1996.

[Srik97] R. Srikant, Q. Vu, R. Agrawal. *"Mining Association Rules with Item Constraints"*. In proceedings of the 3rd Int'l Conference on KDD, California, Aug. 1997.

[StanHoag] S. Staniford, J.A. Hoagland, J.M. McAlerney. *"Practical automated detection of stealthy portscans"*. Journal of Computer Security, 2000.

[SteCu00] R. Sterritt, E.P. Curran, K. Adamson, C.M. Shapcott. *"Visualisation for Data Mining telecommunications network data"*. Data Mining II, (eds.), WIT Press, Southampton, UK, pp 445-454. 2000.

[Sterr00] R. Sterritt, A. H. Marshall, C. M. Shapcott, S. I. McClean. *"Exploring dynamic Bayesian belief networks for intelligent fault management systems"*. Proc. IEEE Int. Conf Systems, Man And Cybernetics, pages 3646-3652. Sept. 2000.

[Subra00] M. Subramanian. "*Network management: Principles and practice*", p. 514-531. Addison-Wesley, 2000.

[SunMic] Sun Microsystems, Inc. *"Installing, Administrating, and Using the Basic Security Module"*. CA, US. Dec. 1991.

[Swatch] *SWATCH* log analysis tool from T. Atkins, available at: http://freshmeat.net/projects/swatch/

[Tasa96] K. Häthonen, M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivenen. *"TASA: Telecommunications Alarm Sequence Analyzer, or How to enjoy faults in your network"*. IEEE/IFIP 1996, Network Operations and Management Symphosium (NOM'96), Kyoto, Japan. Apr. 1996.

[Templ00] S. templeton, K. Levit. *"A requires/provides model for computer attacks"*. In proceedings of New Security Paradigm Workshop, p.31-38. Sept. 2000.

[Thot98] M. Thottan, C. Ji. *"Adaptive Thresholding for proactive network problem detection"*. IEEE Network: Special issue on Network Management, 1998.

[Threat] *Threatman Project* home page: http://sourceforge.net/projects/threatman/

[TruTh] *TruThreatRisk Correlation*, from Arcsight Inc., home page: http://www.arcsight.com/

[TivoMa] *Tivoli Risk Manager*, from IBM Software, home page: http://www-3.ibm.com/software/tivoli/products/risk-mgr/

[Underc02] J. Undercoffer, J. Pinkston. *"Modeling Computer Attacks: A Target-Centric Ontology for Intrusion Detection"*. 2002.

[Val00] A. Valdes, K. Skinner. *"Adaptive, Model-based Monitoring for Cyber Attack Detection"*. In Proceedings of the Int'l Workshop on Recent Advances in Intrusion Detection (RAID'00). 2000.

[Val01] A. Valdes, K. Skinner. *"An approach to Sensor Correlation"*. SRI International report, 2001.

[ValdSkin] A. Valdes, K. Skinner. *"Probabilistic alert correlation"*. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001), p. 31-38, 2001.

[Vaar02] R. Vaarandi. "SEC – A Lightweight Event Correlation Tool". Estonia Science Fundation, IEEE. 2002.

[Vigna00] G. Vigna, S.T. Eckmann, R.A. Kemmerer. *"Attack languages"*. In Proceedings of the IEEE Information Survivability Workshop, Oct. 2000.

[Vigna03] G. Vigna, F. Valeur, R.A. Kemmerer. *"Desiging and Implementing a Family of Intrusion Detection Systems"*. ACM 1-58113-743-5. University of California Santa Barbara, 2003.

[Wiet97] H. Wietgrefe, K.D. Tuchs, K. Jobmann, G. Carls, P. Frohlich, W. Nejdl, S. Steinfeld. *"Using neural Networks for Alarm Correlation in Cellular Phone Networks"*. Hannover University, Germany. 1997.

[Wots] *WOTS* tool from T. Curtis, available at: http://www.tony-curtis.cwc.net/tools/

[Wu94] X. Wu. *"Rule Schema+Rule Body: A 2-level Representation Language"*. Dept. Of Computer Science, James Cook University, Australia. 1994.

[Xlogmst] *XlogMaster* log analysis tool from G. Greve, available at: http://www.gnu.org/software/xlogmaster/

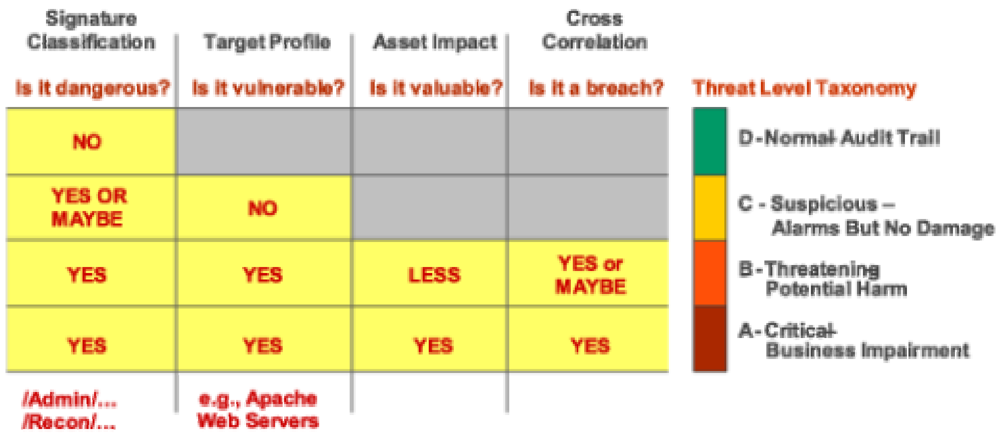
[Yemi96] S.A. Yemini, S. Kliger, E. Mozes, Y. Yemini, D. Ohsie. *"High Speed and robust event correlation"*. IEEE Communications. May 1996.

[Zheng03] Q. Zheng, K. Xu, W. Lv, S. Ma. *"Intelligent Search of Correlated Alarms for GSM Networks with Model-Based Constraints"*. Beijing University of Aeronautics and Astronautics, 2003.

[Zlana] *ZoneLog Analyzer* from MCS, available at: http://zonelog.co.uk/

ANNEX A

This Security Information Taxonomy is developed by ArcSight in real time to provide both the security analyst and the business manager with the information needed to protect important assets. By classifying attacks according to their level of threat and degree of success, and targets according to their vulnerability and value, a simple and powerful four stage warning system is generated. This combination of technical and business filters clearly communicates business-oriented security information while identifying the most important areas of focus for the security staff.



Signature Classification	Target Profile	Asset Impact	Cross Correlation	
Is it dangerous?	Is it vulnerable?	Is it valuable?	Is it a breach?	Threat Level Taxonomy
NO				🟩 D-Normal Audit Trail
YES OR MAYBE	NO			🟨 C - Suspicious – Alarms But No Damage
YES	YES	LESS	YES or MAYBE	🟧 B-Threatening Potential Harm
YES	YES	YES	YES	🟥 A-Critical Business Impairment
/Admin/... /Recon/...	e.g., Apache Web Servers			

ArcSight Security Intelligence Taxonomy