The Leurre.com Project: Collecting Internet Threats Information using a Worldwide Distributed Honeynet

C. Leita [1], V.H. Pham [1], O. Thonnard [2], E. Ramirez-Silva [1]
F. Pouget [3], E. Kirda [1], M. Dacier [1]

Institut Eurecom, Route des Cretes 2229, Sophia Antipolis (France)
[1] $\{leita, pham, ramirez, kirda, dacier\}@eurecom.fr$
[2]$olivier.thonnard@rma.ac.be$,[3] $fabien.pouget@gmail.com$

Abstract

This paper aims at presenting in some depth the Leurre.com project and its data collection infrastructure. Launched in 2003 by the Institut Eurecom, this project is based on a worldwide distributed system of honeypots running in more than 30 different countries. The main objective of the project is to get a more realistic picture of certain classes of threats happening on the Internet, by collecting unbiased quantitative data in a long-term perspective. In the first phase of the project, the data collection infrastructure relied solely on low-interaction sensors based on Honeyd [24] to collect unsolicited traffic on the Internet. Recently, a second phase of the project was started with the deployment of medium-interaction honeypots based on the ScriptGen [15] technology, in order to enrich the network conversations with the attackers. All network traces captured on the platforms are automatically uploaded into a centralized database accessible by the partners via a convenient interface. The collected traffic is also enriched with a set of contextual information (e.g. geographical localization and reverse DNS lookups). This paper presents this complex data collection infrastructure, and offers some insight into the structure of the central data repository. The data access interface has been developed to facilitate the analysis of today's Internet threats, for example by means of data mining tools. Some concrete examples are presented to illustrate the richness and the power of this data access interface. By doing so, we hope to encourage other researchers to share with us their knowledge and data sets, to complement or enhance our ongoing analysis efforts, with the ultimate goal of better understanding Internet threats.

1. Introduction

Understanding the existing and emerging threats on the Internet should help to effectively protect the Internet economy, our information systems and the net citizens. To reach this goal, it is thus necessary to collect sound measurements about the ongoing attack processes observed worldwide on the net. In the last years, the experimental study of Internet threats has gained much attention and many valuable initiatives now exist for monitoring malicious activities or for capturing malware binaries. Important contributions have been made in the field such as: i) the so-called Darknets and Internet telescopes [18, 28, 23], ii) various projects based on the development of low- or high-interaction honeypots [9, 27, 24, 2, 32], and iii) other initiatives aiming at collecting and sharing firewall and IDS logs [10].
In this paper, we present in depth the Leurré.com project and its data collection infrastructure. Launched in 2003 by the Institut Eurécom, this project is based on a worldwide distributed system of honeypots running in more than 30 different countries covering the five continents. The main objective of the project is to get a more realistic picture of certain classes of threats happening on the Internet by collecting unbiased quantitative data in a long-term perspective. We have decided to keep in one centralized database very precise information concerning a limited number of nodes under close scrutiny. Concretely speaking, we deployed identically configured honeypots based on Honeyd [24] on the premises of several partners around the globe. Recently, we started the development and the deployment of new honeypot sensors based on the ScriptGen technology [15] to improve the interaction with the attackers and to enrich our data collection. We record all packets sent to or from these machines, on all platforms, and we store the whole traffic into a database, enriched with some contextual information and with meta-data describing the observed at-

tack sessions. Special care has been taken in the design of such a database to offer an easy and intuitive way to retrieve meaningful information very efficiently. In the next Sections, we present not only the structure of this database but also how it can be used to bring to light some ongoing attack processes on the Internet.
The structure of the paper is as follows: Section 2 presents the initial data collection infrastructure that is based on the deployment of low-interaction honeypots. Section 3 gives some implementation insights into our central repository. We briefly highlight the rationales behind the database design, and we explain why, from a usability point of view, we explicitly included redundant information in the database to make the execution of complex queries more efficient. Section 4 gives a series of practical examples on how to take advantage of this huge database via a convenient interface. We show how this powerful data access interface can greatly facilitate the extraction of rich and meaningful patterns, which can then easily be used as input for data mining tools. In Section 5, we present how we extended our infrastructure with the SGNET deployment, which has recently been opened to anybody willing to host one of its sensors. We explain how we have been able to combine different techniques and analysis tools to increase the richness of the information collected by the honeypots. Then, we present also a practical example on how to retrieve very rich information about the collected threats using SGNET data. Finally, we conclude in Section 6.

2. Leurre.com v1.0 Honeyd

2.1. Historical background

The Institut Eurécom has started collecting attack traces on the Internet in 2003 by means of honeypot responders. The first platform consisted of three high interaction honeypots built on top of the VMware technology (the interested readers in the platform configuration are invited to read [8] for more information). As shown in [8, 7], these first experiments allowed us to detect some locality in Internet attacks: activities seen in some networks were not observed in others. To validate this assumption, we decided to deploy multiple honeypots in diverse locations. With diversity, we refer both to the geographical location and to the sensor environment (education, government, private sectors, etc). However, the VMware-based solution did not seem to be scalable. First, this solution had a high cost in terms of security maintenance. Second, it required significant hardware resources. In fact, to avoid legal issues we would have needed to ensure that these systems could not be compromised and could not be exploited by attackers as stepping stones to attack other hosts. For those reasons, we have chosen a low-interaction honeypot solution, honeyd [24]. This solution allowed us to deploy low-cost platforms, easy to maintain and with low security risk, hosted by partners on a voluntary basis. The low-cost of the solution allowed us to build a distributed honeynet consisting now of more than 50 sensors distributed all over the world, collecting data on network attacks and representing this information under the form of a relational database accessible to all the parters. Information about the identity of the partners and the observed attackers is protected by a Non-Disclosure Agreement signed by each entity participating to the project. We have developed all the required software to automate the various regular maintenance tasks (new installation, reconfiguration, log collection, backups, etc.) to reduce the maintenance overhead related to the management of such a complex system.

2.2. Some technical aspects

We describe here some important technical aspects, including the platform architecture, the logs collection mechanism, the DB uploading mechanism, and the data enrichment mechanism.

Platform architecture: As mentioned before, the main objective is to compare unsolicited network traffic in diverse locations. To make sound comparisons, the platform architecture must be the same everywhere. We tried to make our Honeyd-based solution as similar as possible to the initial VMware setup. We configured Honeyd to simulate 3 virtual hosts running on three different (consecutive) IP addresses. We configured Honeyd's personality engine to emulate the presence of two different configurations, namely two identical virtual machines emulating Windows 2000 SP3, and one machine emulating a Linux Kernel 2.4.20. To the first two configurations (resp. the last) correspond a number of open ports: FTP, Telnet, Web server, Netbios name service, Netbios session service, and Service Message Block (resp. FTP server, SSH server, Web server on ports (80), Proxy (port 8080,8081), remote shell (port 514), LPD Printer service (port 515) and portmapper). We require from each partner hosting the platform a fourth IP address used to access the physical host running Honeyd and perform maintenance tasks. We run tcpdump [29] to capture the complete network traces on each platform. As a security measure, a reverse firewall is set up to protect our system. That is, we accept only incoming connections and drop all the connections that could eventually be initiated from our system (in theory, this should never happen). The access to the host machine is very limited: SSH connections are only allowed in a two-hour daily timeframe and only if it is initiated by our maintenance servers.
Data collection mechanism: An automatized mechanism allows us, on a daily basis, to connect to the platforms

through an encrypted connection to collect the tcpdump traces. The script downloads not only the last day's log file but also the eventual older ones that could not have been collected in the previous days due to, for example, a connectivity problem. All the log files are stored on a central server.

Data uploading mechanism: Just after the data retrieval, the log files are then uploaded to a large database (built on top of Oracle) by a set of Perl programs. These programs take tcpdump files as input and parse them in order to create different abstraction levels. The lowest one corresponds to the raw tcpdump traffic. The higher level is built on the lower ones and has richer semantics. Due to space constraint, we do not present here all the concepts, focusing only on the most important notions.

1. Source: A source corresponds to an IP address that has sent at least one packet to, at least, one platform. Note that, in our Source model, a given IP address can correspond to several distinct sources. That is, an IP remains associated to a given source as long as there is no more than 25 hours between 2 consecutive packets received from that IP. After such a delay, a new source will be assigned to the IP. By grouping packets by sources instead of by IPs, we minimize the risk of gathering packets sent by distinct physical machines that have been assigned the same IP dynamically after 25 hours.

2. Large_Session: it's the set of packets which have been exchanged between one Source and a particular honeypot sensor. A Large_Session is characterized by the duration of the attack, the number of packets sent by the Source, the number of virtual machines targeted by the source on that specific platform, ...

3. Ports sequence: A ports sequence is a time ordered sequence of ports (without duplicates) a source has contacted on a given virtual machine. For example, if an attacker sends the following packets: ICMP, 135 TCP, 135 TCP, 139 TCP to a given virtual machine, the associated ports sequence will be represented by the string $I|135T|139T$. Each large session can have, at most, three distinct clusters associated to it.
This is an important feature that allows us to classify the attacks into different classes. In fact, as mentioned in [8], most attack tools are automatized, it is as likely that the same attack tools will leave the same port sequences on different platforms.

4. Tiny_Session: A Tiny_Session groups the packets exchanged between one source and one virtual host. A Large_Session is thus composed of up to three Tiny_Sessions, ordered according to the virtual hosts IP addresses.

5. Cluster: A Cluster is a set of Sources having exhibited the same network fingerprint on a honeypot sensor. We apply a clustering algorithm on the traffic generated by the sources. The first step of this clustering algorithm consists in grouping large sessions into bags. This grouping aims at differentiating between various classes of activity taking into consideration a set of preliminary discriminators, namely the number of targeted virtual hosts and the unsorted list of port sequences hitting them. In order to further refine the bags, a set of continuous parameters is taken into consideration for each large session, namely: its duration, the total number of packets, the average inter arrival time of packets, and the number of packets per tiny session. These parameters can assume any value in the range $[0, \infty]$, but some ranges of their values may be used to define bag subclasses. This is done through a peak picking algorithm that identifies ranges of values considered discriminating for the bag refinement. Large sessions belonging to a bag and sharing the same matching intervals are grouped together in a cluster. A very last refinement step is the payload validation. The algorithm considers the concatenation of all the payloads sent by the attacker within a large session ordered according to the arrival time. If it identifies within a cluster multiple groups of large sessions sharing similar payloads, it further refines the cluster according to these groups. In summary, a cluster is by design a set of large sessions that seem to be originating from a similar attack tool.

Information enrichment: To enrich the information about each source, we add to it three other dimensions:

1. **Geographical information**: To obtain geographical location such as: organization, ISP, country of a given IP address, we have initially used Netgeo [20], developed in the context of CAIDA Project. It provided a very surprising result which considered Netherlands and Australia as two of the most attacking countries. As a sanity check, we have used Maxmind [17] and we have detected problems with the Netgeo classification. [22] provides a comparison of these two tools. It comes out from this analysis that Netherlands and Australia were not among the top attacking countries anymore when using different sources of information for the geographical loctaion of attacking IP addresses.

2. **OS fingerprint**: To figure out the OS of attacking hosts, we have used passive OS fingerprinting techniques. We take advantage of disco [1] and p0f [33]. It has been shown that p0f is more accurate than disco. Active fingerprinting techniques such as Nmap, Quezo, or Xprobe have not been considered to mini-

mize the risk of alerting the attacker of our investigations.

3. **Domain name:** We also do the reverse DNS lookup to get the domain name of the attacking machine if it is available.

2.3. Generic picture

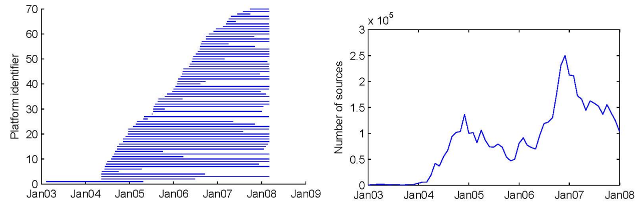

Figure 1. Left: Evolution of platforms, Right: number of sources

Figure 1 (left) shows the evolution of platforms. Each curve corresponds to the time life of a platform. As we can see, we started our data collection in January 2003 with one VMware honeypot and we have started to deploy the distributed low interaction honeypots in April 2004. Since then, the number of partners joining us has kept increasing. In total, we have around 50 offical partners and around 20 former partners. These platforms have, in total, covered 37 different /8 networks, locating in 28 different countries in five continents. In total, we have observed 5173624 sources corresponding to 3461745 different IP addresses. Figure 1 (right) shows the evolution of the number of sources over time. The variation of the curve is of course influenced by the number of platforms. Note that up to April 2004, the traffic is negligible. After that, the number of sources has increased. It is interesting to observe that the number of sources on the last six months of 2004 is much higher than that of the last six months of 2005 even through, in the second case, we have more platforms. In total, there are 155041 different clusters. Figure 2 (left) represents the cumulative distribution function of number of sources per number of cluster. Point (X,Y) on the curve means that Y*100% of the total amount of clusters contain less than X sources. As we can see, most of clusters are very small. There are, in fact, only 15521 clusters containing more than 10 sources each. Interestingly enough, by querying the database one can find that these clusters, ie. around 10% of the total number of clusters, contain in fact 95% of the observed attacks! In other words, the bulk of the attacks is found in a limited number of clusters whereas a very large number of diverse activities originate from a very limited number of sources. In term of attacking machines' OS, according to p0f, almost all attacking machines are Windows ones. This confirms again the results in [8, 7]. Figure 3 shows the top ten attacking countries with US in the head, followed by China and Canada. But the surprising thing is that CS (corresponding to former Serbia and Montenegro) is at the fifth position. The reason is that there is one (and only one!) platform which is heavily attacked by this country. In total, it shows up as one of the most attacking countries. Finally, as an example to show the diversity of the attacks over different platforms, Figure 2 (right) shows the distribution of the number of different clusters per platform. Each column represents the number of distinct clusters observed on a platform. We have as many columns as number of platforms. As we can see, the attacks are highly diverse. On some platforms, we observe just small number of clusters, but it is not the case for others.

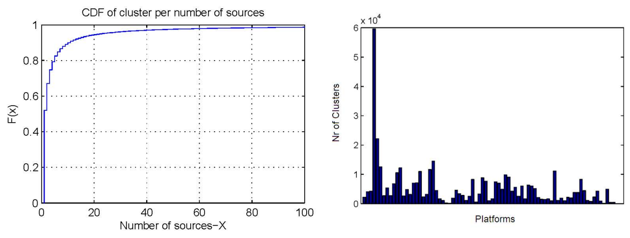

Figure 2. Left:Cumulative distribution function of number of source per cluster; Right:Distribution of number of clusters per platform

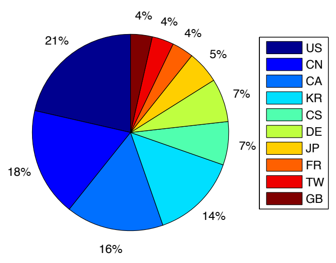

Figure 3. Top ten attacking countries

3. Implementation Insights

3.1. ERD Design Rationales

The purpose of an Entity-Relationship model is to allow the description of the conceptual scheme based on a practical relational model. We can usually optimize this process with respect to various types of constraints (speed, memory consumption, table sizes, number of indices, etc.). Best practices in designing relational databases address problems such as data redundancy and update/insertion anomaly issues. These problems are typically solved by transforming non optimal entity relationships models into their so-called "normal forms" (Third Normal Form, *Boyce-Codd Normal Form*). In our case, though, the problem is slightly different and offers us the freedom not to follow that path. The reasons for that are twofold. First, we do not care about update and insertion anomaly issues as we do not modify the data once it is inserted into the database, as it represents a fact of the past. Second, we do care about the efficiency of querying the database and we are not so concerned by space efficiency (be it on disk or in memory) as the total amount of data we deal with remains rather modest, compared to what existing database systems can handle. Therefore, we have consciously decided to integrate in our design some redundant information. In other words, certain tables contain information that could be retrieved by querying or joining other tables. However, having the results of such queries available at hand in ad-hoc tables proves to be extremely useful when using the database. As a consequence, we decide to keep them, acknowledging the fact that, without their presence, the database would be more 'optimal' according to the classical criteria.

3.2. Leurre.com Architecture

Main ERD Components The data we collect must be properly organized as it will be used for further analysis and experiments. In our setup, as we have three honeypots, a Large Session can be made of 1 to 3 Tiny Sessions (see Section 2). This idea of Tiny and Large Sessions is at the core of the design of the database. Therefore, the five most important tables in the database are the following ones:

- Host: this table contains all attributes (or links to other tables containing attributes) required to characterize one honeypot virtual machine.
- Environment: this table contains all attributes (or links to other tables containing attributes) required to characterize one honeypot platform, i.e. a group of three hosts.
- Source: this table gathers all attributes (or links to other tables containing attributes) required to characterize one attacking source as defined before.
- Large_Session: this table contains all attributes (or links to other tables containing attributes) required to characterize the activity of one Source observed against one Environment.
- Tiny_Session: this table contains all attributes (or links to other tables containing attributes) required to characterize the activity of one Source observed against one Host.

In Figure 4, we present the class diagram corresponding to the ERD model to highlight the respective roles of these main tables. The relationship between Source and Environment is called *Large_Session*. The relationship between one Source and one Host is called a *Tiny_Session*. Primary keys

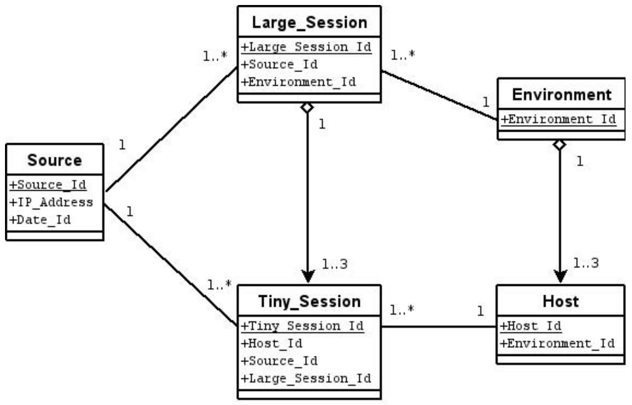

Figure 4. Class diagram of the main DB tables

are underlined in Figure 4. As explained previously, some redundancies have been introduced on purpose. Some examples of meta-data fields include the number of packets, the duration of the attack session, the number of hosts that have been hit, etc (a few other examples of meta-data fields are given a bit further). This redundancy is however beneficial for performance when executing complex queries (see Section 4). Moreover, it greatly simplifies the process of writing new queries.
As of March 2008, the whole database comprises about 40 tables. Recently, we included some new tables to represent data that is specific to the richer information collected thanks our new ScriptGen-based sensors (see Section 5). As of March 2008, the database contains a total of 135 Gbytes. It has been implemented on an Oracle DBMS, running on a cluster of two Red Hat Enterprise Linux servers equipped with a Dual Intel Xeon 3.6 GHz, 6GB RAM and 1.67 TB of SAN. The number of entries for the previously introduced tables are given in Table 1. In the following, we introduce a few other tables. Due to space limitations, we are not able to reproduce the whole database schema. So we will briefly sketch the relevant information that is available in the DB, and we invite the interested reader to contact us if she wants to obtain the whole diagram.

Table names	Number of entries
Source	5,177,283
Large_Session	5,891,633
Tiny_Session	9,854,328
Environment	81
Host	320

Table 1. Volume overview of the main tables (March 2008)

Packets information We store in the database detailed information on all the packets sent and received by all our honeypots following the same schema used the Snort Intrusion Detection System. Packets are either coming from a Source or sent by a Host. They are linked to a given Tiny_Session. Figure 5 presents the resulting new tables. Each packet has its unique identifier, called cid.

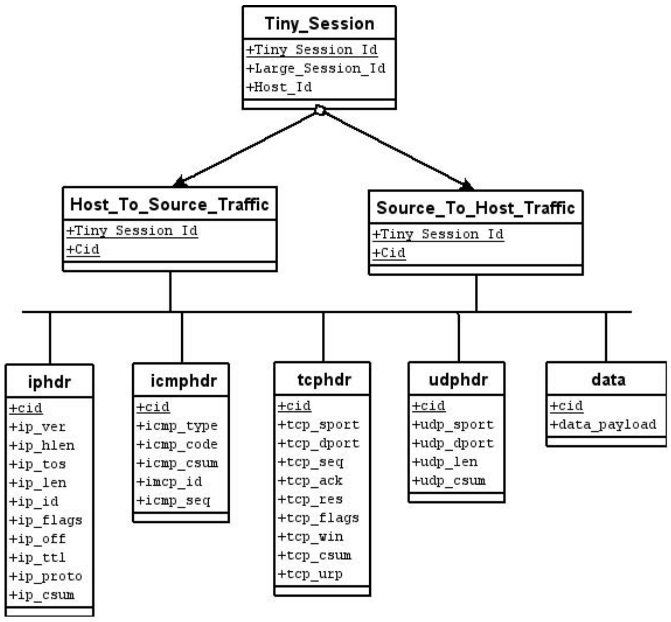

Figure 5. Class diagram of the packets tables

Additional Contextual Information

1. Geographical location. The geographical location of IPs can provide interesting information to better understand where the attacks are coming from. As there are differences between NetGeo and other tools such as MaxMind, IP2location or GeoBytes, we have decided to include into our database the geographical information provided by several tools and leave it up to the user to choose the ones he felt more comfortable to work with. We have subscribed to a commercial solution called MaxMind, and we kept NetGeo information as well in separated tables for comparison purposes. The geographical information can be found in two tables called Info_Source_Maxmind and NetGeoInfo [16, 20].

2. Passive OS Fingerprinting. Today, there is general consensus that most worms and botnets propagate through Windows machines. So it seems there exists some correlation between the Operating System of the attacker and the attack type it is performing. As we want our honeypot machines to remain passive, we run so-called passive fingerprinting tools on the collected packets in order to guess the OS of the attacker. Most of those fingerprinting tools compare packet fields to a given fingerprints database. As there may be some differences between those tools, we have decided to use two different passive OS fingerprinting utilities. They are respectively called Disco and p0f [1, 33]. Their output is in text format which we parse to store the information into the database. For practical reasons, we consider that the OS fingerprinting information is related to a given Large_Session and not to a given Source. So this information is stored in a new separate table called Info_OS.

3. Domain Name Resolution and Subnet Information. Another potentially interesting information can be the machine hostname and the domain it belongs to. Both Netgeo and Maxmind provide information on the network the Source is coming from. We derive from the IP-range network values the CIDR (Classless Inter-Domain Routing) network mask. So, we introduce into the database three new external important information types: the domain reverse resolution, the machine hostname and the network parameters. Since this information characterizes uniquely a given Source, we have decided to enrich the corresponding table with this information by adding attributes, the value of which point to new tables providing the imported information. Some interesting statistics can be done on the domain names like the extraction of the top most attacking domains involved in certain attack processes. Other statistics can be done quite easily by checking if the machine hostname belongs to a company network or is more likely to be a home computer. For instance, simple extractions of patterns like '%dsl%', '%cable%', '%dial%' are good indicators of home computers. On the other hand, patterns like '%web%', '%mail%', '%serv%' in the machine name are likely to show up for machines providing some well defined services.

Additional Meta-Data Meta data information can be seen as redundant information that is saved in specific tables but that could also be derived by means of queries against the other tables. Meta-data tables are built by means of automated scripts that are run whenever new data is entered into the database. We provide in the following a non-exhaustive list of meta-data information describing the attack sessions, that can be found today in the database:

- For each Large_Session: the identifier of the attack cluster, as determined by our clustering engine (see Section 2)
- The duration of the observation for a given Large_Session
- The average inter-arrival time between packets sent by a given Source
- An attribute that indicates how many virtual machines were targeted
- The sequence of ports that have been targeted during the attack on a virtual machine
- A boolean value indicating if a given Source has already been observed or not
- An attribute to mark Tiny_Sessions that are likely due to backscatter activities
- Another boolean value to indicate if attacks on multiple virtual machines were performed in sequence or simultaneously.

The main idea is that we do not want to compute this meta-data information whenever we need it. It is considered to be useful enough to be part of the database information.

4. Threats Analysis - Some Illustrative Examples

In this Section, we provide a series of illustrative examples on how to effectively use our database to perform meaningful analyses on the observed threats. These examples are based on the data collected with the low-interaction honeypots; we provide in the next Section some insights into the enriched information collected through the new ScriptGen-based sensors. For the interested reader, we give in Appendix 1 some global statistics about the traffic collected in Leurre.com on a yearly basis, starting from 2004 until March 2008.

4.1. Temporal Evolution of Attack Clusters

Time series analysis can provide valuable information (e.g. trends, abrupt changes, and emerging phenomena) to security practitioners in charge of detecting anomalous behaviors or intrusions in the collected traffic. In this first illustration we want to show how an analyst can retrieve the temporal evolution of a given attack cluster, by counting the number of sources belonging to that cluster on a chosen time scale (e.g. by day or by hour).

For that purpose, we chose the attack cluster 17718 that is mostly targeting two platforms (platform 14 in Belgium, and platform 42 in the UK.). This cluster is related to attack sessions that have created the port sequence |I|445T (i.e. ICMP followed by 445/TCP). By executing the Query 1 (see Appendix), we can easily retrieve the time series of the cluster 17718 for a time period ranging from 1-Dec-06 until 01-Mar-07, and by grouping the source count by day. Obviously, this groups all platforms together; so we could run similar queries for each platform separately just by adding one additional WHERE clause, so as to analyze the impact of this attack cluster on different platforms. The results of this kind of queries have been represented on Fig 6. One

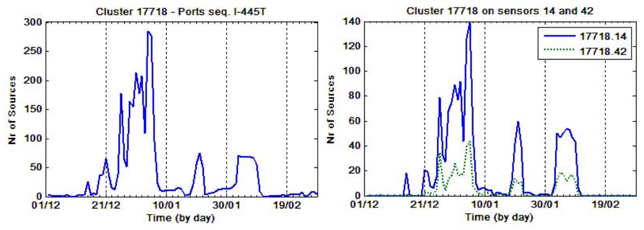

Figure 6. Left: the global time evolution of attack cluster 17718, with the sources aggregated by day. Right: the time evolution of the same cluster for the platforms 14 and 42 separately.

could wonder why we have chosen a time scale of one day. Indeed, for certain types of attack phenomena, it might be useful to look at a finer time granularity, such as intervals of 1 hour, for instance. This type of queries can easily be executed as well, and we give one more illustration with the case of a very large cluster (15611) which appears to be closely related to the previous cluster, as the time plot might suggests on Fig 7 (left). The cluster 15611 is related to ICMP traffic only. It might be correlated with the first stage of the same root attack phenomenon as cluster 17718, because of their very similar (and synchronized) temporal patterns.

To conclude this illustration, we counted the number of sources for the same cluster but on a smaller time window (from 31-Jan-07 until 10-Feb-07) and with a time scale of one hour (with minor modifications to the query given here above). The chosen time window corresponds to the last "wave" of attacks of that cluster, which is visible on Fig 7 (left). The result of this new query has been represented

Country of origin	Nr of Sources	Relative %
CN	1150	35.3
US	378	11.6
CA	255	7.8
FR	236	7.2
unknown	215	6.6
TW	137	4.2
JP	128	3.9
IT	120	3.6
DE	107	3.3
Others	524	16.1

Table 2. Geographical distribution for attack cluster 17718

Domain of origin (Level-1)	Nr of Sources
.net	817
.com	171
.jp	143
.cn	129
.it	120
.fr	61
.tw	51
.de	48
Others	353

Table 3. Distribution of Level-1 domains for attack cluster 17718

on Fig 7 (right). As the graph suggests, we can observe a strong recurrent diurnal/nocturnal activity pattern, and those regular attack waves last apparently for 7 or 8 days.

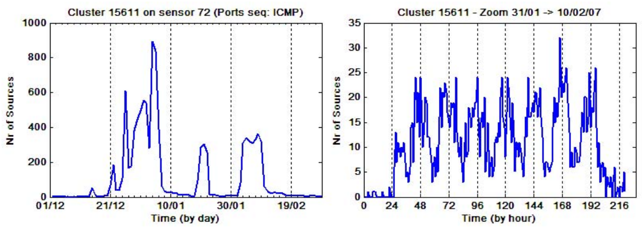

Figure 7. Left: the time evolution of attack cluster 15611 on sensor 72, with the sources aggregated by day. Right: A zoom by hour in the time window 31-Jan until 10-Feb for the same attack cluster.

4.2. Geographical Localization of Attackers

If we consider the previous example, we could wonder from which countries the sources belonging to attack cluster 17718 are coming from during the activity period of this attack process. We can easily retrieve such information with a query similar to Query 2 (see Appendix).

The result of executing this query gives a 2-column table (Table 2): the first column indicates the country of origin (represented with its ISO code) and the second column gives the number of sources belonging to that country. Also, we have put in a third column the corresponding relative percentage for each country with respect to the total number of sources for this attack process (i.e. 3250).

4.3. Attackers Domain Names

In the same way as we did for the geographical aspect, we could also analyze the domain information related to the attacking sources, which would give us a refined view on the origins of the attackers. If we execute the Query 3 given in the Appendix, we obtain a table with the Level-1 domains in the first column, and the corresponding number of attacking sources in the second column (Table 3). By slightly modifying the regular expression in the query, we can retrieve domain information at different levels (Level-1, Level-2, ...).

4.4. Attackers Subnets Information

Finally, an analyst could be interested in looking at the subnets of origin of the attackers. This is also a quite easy task to do thanks to the database scheme. For example, we can easily retrieve the class A information about the attacking sources belonging to the very same cluster as in the previous examples (cluster 17718) by executing the SQL Query 4 given in Appendix. The results are presented in Table 4 [1]. Again, the regular expression can be modified so as to catch a larger part of the IP addresses (Class B, C, ...).

5. Leurre.com v2.0: SGNET

5.1. Increasing the level of interaction

We have seen in the previous Section how we have been able to generate valuable dataset with quantitative information on the localization and the evolution of Internet unsolicited traffic. We are able to observe interesting behaviors, most of which are very difficult to justify or to attribute to a specific root cause. It is, indeed, very difficult to link a given observation to a class of activities, and our search for answers in this direction had to deal with a limited amount of information about the final intention of the attacker. The

[1]To preserve the confidentiality related to the IPs of the attackers, the first byte values have been replaced by letters in the table

Subnet of Origin - Class A	Nr of Sources
A.x.x.x	451
B.x.x.x	193
C.x.x.x	168
D.x.x.x	160
E.x.x.x	159
F.x.x.x	123
G.x.x.x	113
H.x.x.x	100
I.x.x.x	91
J.x.x.x	90
Others	1602

Table 4. Anonymized distribution of Class A-subnets for attack cluster 17718

low level of interaction of the Leurré.com honeypots is a limiting factor: when a honeypot receives a client request, it is not able to carry on the network conversation with the attacker, nor to "understand" it.

For instance, in our experience within the Leurré.com project, due to the lack of emulation scripts we have been able to observe only the first request of many interesting activities such as the spread of the Blaster worm [4]. But since Blaster sends its exploit in the second request of its dialog on port 135, we have never been able to observe such a payload. Therefore it becomes very difficult to distinguish Blaster's activity from other activities targeting the same port using solely the payload as a discriminating factor. Fortunately, experience shows that, even such limited amount of information, a large variety of analyses remain applicable and deliver useful results. In order to increase the amount of available information on attackers, we need to increase the level of interaction with the honeypots. However, in order to keep carrying on our deployment of sensors on a voluntary basis, we need to achieve this objective at the lowest possible cost. This led to the development of the ScriptGen approach.

5.2. ScriptGen

The ScriptGen technology [15, 14] was created with the purpose of generating honeypots with a high level of interaction having a limited resource consumption. This is possible by *learning* the behavior of a given network protocol when facing deterministic attack tools. The learnt behavior is represented under the form of a Finite State Machine representing the protocol language. The generated FSM can then be used to respond to clients, emulating the behavior of the real service implementation at a very low cost.

The ScriptGen learning phase is completely protocol agnostic: no knowledge is assumed neither about the structure of the protocol, nor on its semantics. ScriptGen is thus able to replay any deterministic run of a protocol as long as its payload is not encrypted. The ScriptGen learning takes as input a set of samples of network interaction between a client and the real implementation of a server. The core of the learning phase is the Region Analysis algorithm introduced in [15]: taking advantage of bioinformatics alignment algorithms [19], the algorithm exploits the statistical variability of the samples to identify portions of the protocol stream likely to carry a strong semantic meaning and discard the others. In order to build reliable representations of the protocol interaction, it is thus necessary to collect a clean set of samples with enough statistical variability to correctly identify semantically important regions. Figure 8 shows an example of semantic abstraction for an excerpt of SMTP FSM.

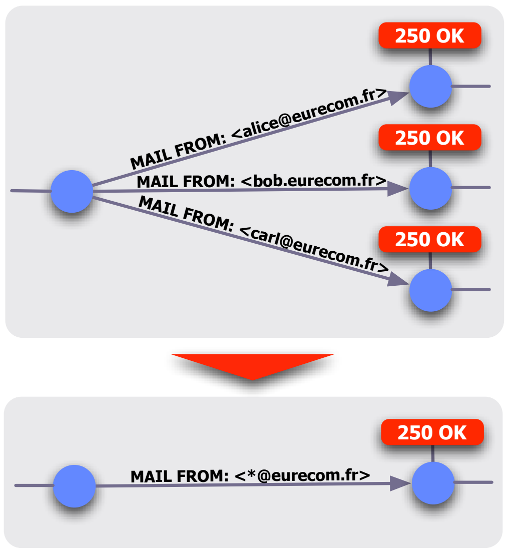
250 OK
MAIL FROM: <alice@eurecom.fr>
250 OK
MAIL FROM: <bob.eurecom.fr>
MAIL FROM: <carl@eurecom.fr>
250 OK
250 OK
MAIL FROM: <*@eurecom.fr>

Figure 8. ScriptGen FSM generalization

The properties of the ScriptGen approach allow to perform a completely automated incremental learning of the activities as shown in [14]. ScriptGen-based honeypots are able to detect when a client request falls out of the current FSM knowledge (a 0-day attack or, more exactly, a yet unseen attack) by simply detecting the absence of a matching transition. In such case, the honeypot is thus unable to provide a valid answer to the attacker. We showed in [14] how the honeypot can react to this situation relying on a real host (an *oracle*) and acting as a proxy between the attacker and the real host. This allows the honeypot to continue the conversation with the attacker, and to collect a new sample of protocol interaction that can be used to automatically refine the protocol knowledge.

ScriptGen is able to correctly learn and emulate the exploit phase for protocols as complex as NetBIOS [14].

ScriptGen thus allows to build highly interactive honeypots at low cost. The oracles needed to learn new activities can be hosted in a single virtualization farm and contacted by the honeypots through a tunneling system, in a structure similar to Spitzner's *honeyfarm* concept. Differently from classical honeyfarms, access to the real hosts is a rare event resulting from the occurrence of a new kind of attack. As a consequence, systems based on the ScriptGen honeypots potentially have a high degree of scalability.

5.3. SGNET: a ScriptGen-based honeypot deployment

We took advantage of this technology to build an experimental honeypot deployment, called SGNET, meant to follow the lines of the Leurré.com deployment but providing a significant improvement in the richness of the collected data.

SGNET and code injections SGNET is a scalable framework that offers almost the same amount of information than real high interaction systems for a specific class of attacks, namely server-based code injection attacks generated by deterministic scripts. We are aware of the fact that they correspond only to a subset of the possible attack scenarios. However, as of today, they are considered to be responsible for the creation of large botnets [25] and the preferred propagation mechanisms of a large number of different malware.

The final objective of a code injection attack consists in forcing the execution of executable code on a victim machine exploiting a vulnerable network service. Crandall et al. introduced in [6] the epsilon-gamma-pi model, to describe the content of a code-injection attack as being made of three parts.

Exploit (ϵ). A set of network bytes being mapped onto data which is used for conditional control flow decisions. This consists in the set of client requests that the attacker needs to perform to lead the vulnerable service to the control flow hijacking step.

Bogus control data (γ). A set of network bytes being mapped onto control data which hijacks the control flow trace and redirects it to someplace else.

Payload (π). A set of network bytes to which the attacker redirects the vulnerable application control flow through the usage of ϵ and γ.

The payload that can be embedded directly in the network conversation with the vulnerable service (commonly called shellcode) is usually limited to some hundreds of bytes, or even less. It is often difficult to code in this limited amount of space complex behaviors. For this reason it is normally used to force the victim to download from a remote location a larger amount of data: the malware. We extend the original epsilon-gamma-pi model in order to differentiate the shellcode π from the downloaded malware μ.

An attack can be characterized as a tuple $(\epsilon, \gamma, \pi, \mu)$. In the case of, old, classical worms, it is possible to identify a correlation between the observed exploit, the corresponding injected payload and the uploaded malware (the self-replicating worm itself). Thanks to the correlation between the 4 parameters, retrieving information about a subset of them was enough to characterize and uniquely identify the attack. This situation is changing. Taking advantage of the many freely available tools such as Metasploit [30, 26], even unexperienced users can easily generate shellcodes with personalized behavior and reuse existing exploit code. This allows them to generate new combinations along all the four dimensions, weakening the correlation between them. It is thus important to try to retrieve as much information as possible on all the 4 dimensions of the code injection. We designed SGNET in such a way to delegate to different functional components the 4 dimensions, and combine the information retrieved by these components to have an exact picture of the relationships among them.

The ScriptGen approach is suitable for the learning of the exploit network interaction ϵ, offering the required level of interactivity with the client required to lead the attacker into sending code injection attacks. For the previously stated reasons, in SGNET we extend this capability with the information provided by other tools in order to retrieve information on the other dimensions of the epsilon-gamma-pi-mu (EGPM) model. We take advantage of the control flow hijack detection capabilities of Argos [21] to detect successful code injection attacks, understand the bogus control data γ and retrieve information about the location of the injected payload π. We take advantage of the shellcode emulation and malware download capabilities of Nepenthes [2] to understand the payload π, emulate its behavior and download the malware sample μ.

When facing an attacker, the SGNET activity evolves through different stages, corresponding to the main phases of a network attack. SGNET distributes these phases to three different functional entities: *sensor*, *sample factory* and *shellcode handler*.

The SGNET sensor corresponds to the interface of the SGNET towards the network. The SGNET deployment aims at monitoring small sets of IPs deployed in multiple locations of the IP space, in order to characterize the heterogeneity of the activities along the Internet as observed in [8, 5]. SGNET sensors are thus low-end hosts meant to be deployed at low cost by different partners willing to join the project and bound to a limited number of IPs. The deployment of the sensors follows the same win-win partnership schema explained before. Taking advantage of the ScriptGen technology, the sensors are able to handle autonomously the exploit phase ϵ of attacks falling inside the

FSM knowledge with minimal resource requirements on the host.

The SGNET sample factory is the *oracle* entity meant to provide samples of network interaction to refine the knowledge of the exploit phase when facing unknown activities. The sample factory takes advantage of a real host running on a virtual machine and monitors the host state through memory tainting. This is implemented taking advantage of *Argos*, presented by Portokalidis et al. in [21]. Keeping track of the memory locations whose content derives from packets coming from the network, Argos is able to detect the moment in which this data is used in an *illegal* way. Argos was modified in order to allow the integration in the SGNET and load on demand a given honeypot profile with a suitable network configuration (same IP address, gateway, DNS servers, ... as of the sensor sending the request). The profile loading and configuration is fast enough to be instantiated on the fly upon request of a sensor.

The Argos-based sample factories provide information about the presence of code injections (γ) and are able to track down the position in the network stream of the first byte being executed by the guest host, corresponding to the byte B_i of the payload π. We have developed a simple heuristic to identify the injected payload π in the network stream starting from the *hint* given by the sample factory [12]. This allows to embed in the ScriptGen learning additional knowledge, namely the a tag identifying the final state of a successful code injection and information within the preceding transitions that allows to extract from the attacker's procotol stream the payload π.

The final steps of the code injection attack trace are delegated to the SGNET shellcode handler. Every payload π identified by the SGNET interaction is submitted to a shellcode handler. The shellcode handler is implemented reusing part of the functionality of the Nepenthes [2] honeypots. We take advantage of Nepenthes shellcode analyzer to "understand" the payload π and emulate its behavior using Nepenthes download modules. In the context of the SGNET, Nepenthes is thus used as an *oracle* for the payload emulation. Differently from the exploit phase, we do not try to learn the Nepenthes behavior in terms of FSM. We consider the payload emulation a too complex interaction to be represented in terms of a FSM.

SGNET Architecture The general architecture of the SGNET is presented in Figure 9. All the SGNET entities communicate through an ad-hoc HTTP like protocol called Peiros [13]. The Peiros protocol allows communication under the form of a set of service requests, allowing for instance a sensor to require the instantiation of a sample factory. The sensors, distributed over the IP space and hosted by partners of the project, are connected to a central entity called SGNET gateway, that acts as an application-level

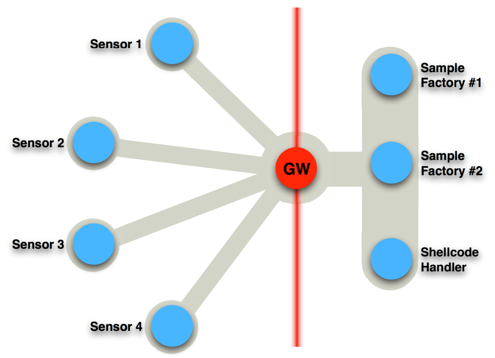

Figure 9. SGNET architecture

proxy for the Peiros protocol. The gateway receives service requests from the sensors and dispatches them to a free internal entity, performing a very simple load balancing. The architecture offers a clean separation between the sensors, relatively simple daemons running over inexpensive hosts, and the internal entities, having a higher complexity and higher resource requirement.

We saw how the ScriptGen learning exploits the variability of the samples to produce "good" refinements of the FSM knowledge. The architecture of Figure 9 shows how the SGNET gateway offers a unique standpoint to collect interaction samples: all the tunneled conversations between any sensor and any sample factory flow through the gateway. The gateway becomes thus the best candidate to perform ScriptGen refinements to the current FSM knowledge. Once a new refinement is produced, the gateway takes care of updating the knowledge of all the sensors pushing them the FSM updates. This makes sure that all the sensors online at a given moment share exactly the same knowledge of the protocols.

An important aspect related to the ScriptGen learning is the strict relation between the ScriptGen ability to learn exploits and the configuration of the sample factories. If a service is not installed or activated in the configuration of the virtualized host handled by the sample factory, the SGNET architecture will not be able to observe activities targeting it. It is thus important to carefully configure the sample factories in order to maximize the visibility of malicious activities. We chose to address this problem supporting the assignment of different profiles for the IPs of the SGNET sensors, similarly to what was done on the Leurré.com deployment. Each profile is assigned to a different sample factory configuration, with different services and different OS versions to maximize the visibility on network attacks of our deployment.

The description of the SGNET deployment clearly shows a difference with respect to the original Leurré.com deployment. SGNET is a more complex architecture, that succeeds in raising the level of interaction of the honeypots without raising the resource requirements for the partners hosting the sensors. Taking advantage of the ScriptGen learning, the deployment also allows to minimize the usage of expensive resources such as the sample factories, that are needed only to handle those activities that do not fall *yet* in the FSM knowledge. An important concern for the partner taking advantage of this deployment is the security of the solution. SGNET raises the level of interaction of the honeypots; it is thus important to guarantee that the increased interactivity does not impact the safety of hosting a honeypot platform. The network interaction driven by FSM knowledge is virtually as safe as any low-interaction honeypots: the attacker interacts with a simple daemon performing state machine traversals to provide answers to client requests. When a new activity is handled, the sensor acts as a proxy and the attacker is allowed to interact with a real (and thus vulnerable) host. Two measures are in place to ensure the safety of this process. Firstly, the tunneling system ensures that any outbound packet generated by the sample factory is directed only towards the attacking source (blocking any attempt of exploiting the honeypot as a stepping stone to attack others). Secondly, the memory tainting capabilities of Argos allow us to stop execution as soon as the attacker successfully hijacks the host control flow. This does not include for instance successful password brute-forcing attacks, but this class of attacks can be prevented by a careful configuration of the virtualized host.

5.4. Retrieving information on the attacker using SGNET

In this Section, we will show how we are able to exploit the increased level of interaction of the SGNET deployment to extract information about the attacker, following the 4 dimensions of the epsilon-gamma-pi-mu model. This Section aims at giving an overview on the nature of the information and on the main concepts, but does not aim at being exhaustive. The SGNET deployment has been running in an experimental form starting from late 2007, proving the validity of the approach [12]. We underlined in [13] how the whole design of the SGNET was conceived to ease the integration of different information sources or functional modules.

The experimental SGNET deployment is running on a limited number of sensors hosted by partners that agreed to participate to the experimentation. All the honeypot IPs have been assigned to a single sample factory profile, corresponding to an unpatched Windows 2000 host running IIS 5.0. We expect the implementation of the SGNET (as well as the representation of the collected data) to evolve further, and we strongly invite other research groups interested in these subjects to contribute along these lines.

SGNET follows a data collection schema similar to that proposed for the Leurré.com deployment. Network dumps are collected from each sensor and stored in a relational database. The increased information provided by SGNET is retrieved under the form of a set of logs that are combined together into additional concepts that extend those seen in Section 2.2.

- Sensor logs. Information about the network interaction (traversal of FSM, presence of code injections,...)
- Sensor dumps. Complete tcpdump trace of the network interaction of the attackers with the sensors.
- Gateway logs. Information about the status of the sensors (activity status, configuration) and about the FSM refinement.
- Nepenthes logs. Information about the nature of the shellcodes, on their network behavior, and downloaded malware samples.

The additional concepts derived from these logs are spread over the 4 dimensions of the epsilon-gamma-pi-mu model.

Exploit information Most of the SGNET concepts are based on the concept of ScriptGen Session (or SGSession). ScriptGen Sessions are a specialization of the concept of Tiny Session introduced in Section 2.2. A ScriptGen Session identifies a single TCP session or to a couple of UDP request and answer. A ScriptGen Session corresponds to the scope of a ScriptGen FSM: in the case of TCP, the FSM root corresponds to the establishment of a connection and the leaf to its completion through RST or FIN packets.

The sensor logs provide information on the way each ScriptGen Session is handled. Each session is characterized by a source and destination address, the corresponding ports and a timestamp. It is also associated to two main attributes: *type* and *path*. The type can correspond to one of these 4 values.

- Type SG. The session was handled taking advantage of the FSM knowledge only.
- Type HY. The session was partially handled by FSMs, but an unknown request required the instantiation of a sample factory.
- Type AG. When a sample factory was instantiated for a given activity, all the following sessions generated by the same attacker are relayed to that same sample factory and are marked with this type.
- Type NP. The session was handled by the shellcode handler to emulate the behavior of a successfully injected payload. These sessions correspond to the malware download phase.

The *path identifier* associates a ScriptGen Session of type SG to the corresponding FSM traversal produced by the network activity. The path identifier proved to be a very reliable indicator of the *type* of activity being observed. Since the ScriptGen algorithm aims at grouping together similar network activities and generalize their network behavior under the form of FSM traversals, each traversal is a very reliable identifier of a given type of activity. At the moment of writing, the SGNET experimental deployment identified 66 paths, 11 of which led to successful code injections. These 11 traversals classify into 11 different groups 2548 exploits leading to successful code injections observed between the 1st of November 2007 and the 1st of March 2008 and handled taking advantage of the sole FSM knowledge.

Bogus control data and shellcode information SGNET can identify a code injection either during the interaction with the sample factory (when handling a new exploit) or through the knowledge already embedded in a Finite State Machine. In both cases, SGNET is able to identify the position of the payload π within the protocol stream and extract its content. We are able to retrieve and store for further analysis the payload binary π.

The analysis of the payload π performed by the shellcode handler produces two main outputs: the CMD shell string and the download URI.

We have observed that certain classes of payloads force the victim host to open a socket listening for incoming connections, and associate this socket with a cmd.exe instance. This allows the attacker to connect to that port, upload a malware binary (using, for instance, tftp) and then execute such file. Also, in some cases the attacker directly embeds the shell commands to be executed by cmd.exe within the injected payload, that is then packed and inserted within the network stream. In both cases, the Nepenthes shellcode analyzer enables us to retrieve the stream of commands issued by the attacker. For instance, the sequence of cmd commands issued by the Blaster worm and retrieved by SGNET is the following: *"tftp -i 1.1.1.1 GET msblast.exe; start msblast.exe; msblast.exe; msblast.exe"* where 1.1.1.1 is the address of the attacker. This information is extremely valuable to characterize this class of payloads.

The output of the Nepenthes shellcode analyzer is a URI representing the location of the malware sample to be downloaded. This URI gives valuable information on the kind of protocol being used for the download (ftp,http,tftp,...) as well as the malware distribution strategy. We can in fact distinguish:

- Phone-home strategies. The victim is forced to download the malware sample from the attacker through a download session initiated by the victim. This strategy maximizes the probability of success in downloading the sample also in presence of firewalls, on the victim side, and was used by many worms such as the previously cited Blaster [4]. It can fail if the attacking side is NATed and/or protected by a firewall.
- Central malware repositories. An attacker may distribute malware samples taking advantage of centralized repositories, eventually hidden behind fast-flux networks.
- Push-based strategies. The victim is forced to receive the malware sample opening a port and passively receiving the malware through a connection initiated by the attacker. This technique is used, for instance, by the Allaple [11] worm and has the opposite advantages and drawbacks than the phone-home strategy.

The data collected by the experimental deployment in the period between the 1st of November 2007 and the 1st of March 2008 shows some interesting insights with respect to the preferred download methods used by currently spreading malware. Out of a total of 1511 injections whose shellcode was successfully analyzed by the shellcode handler, 826 of them (54%) took advantage of push-based strategies. In 803 of these cases, the port used to upload the malware was port 9988. The remaining 685 code injections took advantage of phone-home strategies. Until now, we never observed the usage of central malware repositories to distribute binaries.

The interaction of the SGNET with the shellcode handler has already underlined some interesting facts related to the ability of the shellcode handler (whose implementation, being derived from Nepenthes, is signature based) to recognize the shellcodes collected by SGNET. We have been able to identify shellcodes not correctly recognized by the shellcode handler due to the lack of a proper signature in the Nepenthes shellcode engine [12]. We have been able to report these cases to the Nepenthes development team that refined the corresponding signature.

Malware information The ultimate result of the shellcode handler activity is the download of a malware sample. The malware dimension μ potentially provides extremely detailed information on the identity of the attacker (being in a way a sample of the attacker itself), and gives us a way to link the network activities observed within SGNET to other sources of information, such as the malware databases maintained by AV vendors. Each malware sample μ_i collected by SGNET is treated taking advantage of different tools and services. From a static analysis point of view, we perform very simple analyses such as looking at the size of the samples and at the file magic numbers to understand their nature. We take advantage of VirusTotal [31] to retrieve information about the signature given to the malware by 22 different AntiVirus vendors. This information is extremely valuable to evaluate, for instance, the relevance of a given malware sample: if a malware sample is detected by a few vendors, it is likely to be a new threat and as such deserves special attention.

We take advantage of the Anubis sandbox [3] to retrieve information about the dynamic behavior of the malware sample. Anubis provides a wide range of information, that we use to build a *behavioral profile* of the malware. This information ranges from the name of the processes, services and registry key being created by the execution of the sample to aggregate information on its network behavior (connections to IRC channels, scanning attempts on the different ports, ...).

In the previously considered period among the 1st of November 2007 and the 1st of March 2008 we collected 788 unique MD5s of malware samples. These malware samples corresponded to 48 different malware families according to the F-Secure AV solution, and only 14 according to Symantec Antivirus. 525 of these MD5s were downloaded a single time, and are thus likely to belong to families of malware taking advantage of some polymorphism to avoid detection (i.e. the Allaple worm). This shows the difficulty of the problem of the malware classification, and the need to elaborate better ways to classify malware samples taking advantage, for instance, of the behavioral information. Also, the information provided by the Anubis sandbox allows us to detect among the 788 malware samples 102 cases in which the downloaded sample was corrupted and could not be executed by the Windows host. The file corruption seem to be due to lack of reliability of some download protocols. Anubis information allows to filter out from the antivirus statistics those cases in which the behavior of an antivirus solution is not clearly defined. We saw that certain vendors seem to choose not to generate alerts when facing a corrupted binary, while others tend to generate an alert in any case. Since this is mainly due to a policy choice, it is impossible to compare the performance of different vendors in the case of corrupted binaries.

A case study In order to exemplify the kind of information provided by the SGNET deployment, let's consider a real-world example extracted from our current dataset. The 18th of December, at 21:16 UTC, one of our honeypots was exploited by a host located in Romania.

The exploit, targeting port 135, was known and thus autonomously handled by the ScriptGen knowledge. This exploit is composed of only two packets targeting port 135, and is used for the propagation of various malware samples, such as "Hupigon.gen83" and "W32/Pinfi.A" (F-Secure).

The injected payload is quite small, 573 bytes of which 161 are NOP instructions. According to the shellcode handler, the payload forces the victim to download a malware sample using the *link* protocol from the attacking host connecting on port 3165 (ie, push-back strategy). The SGNET honeypot successfully downloaded the malware sample.

The malware sample, of length 71175 bytes and MD5 hash caf208342b87013543333059c189a34d, was recognized by most antivirus vendors with different names, such as:

- "W32/Smalldoor.AWCI" (F-Secure),
- "Trojan Horse" (Symantec),
- "W32/Poebot.NN.worm" (Panda Antivirus),
- "Trojan.Agent-11146" (ClamAV).

Submitting the malware to Anubis, a number of malicious behaviors was detected by the sandbox. The malware started to scan random IPs trying to connect on port 135 (supposedly trying to propagate), connected to a IRC server and joined multiple channels, and also connected to a web server located in Canada and downloaded from there two additional executable files placed in the Windows system directory with filenames "shdcmg.exe" and "tnnhpmhj.exe".

This is an example of the vast amount of information provided by the SGNET deployment for a single code injection attack. The amount and the quality of the collected data is potentially extremely significant, but requires a widespread deployment of sensors along the IP space. We thus strongly invite any research or industrial entity to take part to the deployment and actively contribute to it and, by doing so, to benefit from the access to this rich dataset.

6. Conclusions

In this paper, we have presented in detail Leurré.com, a worldwide distributed system of honeypots running since 2003. We have extensively described its internal architecture that has been developed to collect meaningful data about some ongoing attack processes observed at various places on the Internet.
Several examples have been given throughout the text to illustrate the richness of our central data repository and the flexibility of its design, enabling a large diversity of analyses to be carried out on it. It is not the purpose of this paper to report on a specific analysis. Other publications have focused on some of these issues and some more work is ongoing. We have shown though by means of simple examples that this database helps in discovering trends in the attacks and in characterizing them. Next to this, we have also presented the important improvements we made to our infrastructure by deploying high-interaction ScriptGen sensors, which enable us to collect even more precise

and valuable information about malicious activities. In the light of some early promising results, we showed that this entire data collection infrastructure holds a great potential in augmenting our threats intelligence capability on the Internet. Being able to conduct in-depth analyses on this huge data collection, in a systematic way, will hopefully help us to make some advances towards the creation of early warning information systems.
So, it is our wish to share the data contained in this database with those interested in carrying some research on it. The authors can be reached by mail to get detailed information regarding how to join the project in order to gain access to the database.

References

[1] ALMODE Security. Home page of disco at at http://www.altmode.com/disco/.

[2] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.

[3] U. Bayer, C. Kruegel, and E. Kirda. *TTAnalyze: A Tool for Analyzing Malware*. PhD thesis, Master's Thesis, Technical University of Vienna, 2005.

[4] CERT. Advisory CA-2003-20 W32/Blaster worm, August 2003.

[5] E. Cooke, M. Bailey, Z. M. Mao, D. Watson, F. Jahanian, and D. McPherson. Toward understanding distributed blackhole placement. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 54–64, New York, NY, USA, 2004. ACM Press.

[6] J. Crandall, S. Wu, and F. Chong. Experiences using Minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities. *Proceedings of GI SIG SIDAR Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2005.

[7] M. Dacier, F. Pouget, and H. Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.

[8] M. Dacier, F. Pouget, and H. Debar. Honeypots, a practical mean to validate malicious fault assumptions. In *Proceedings of the 10th Pacific Ream Dependable Computing Conference (PRDC04)*, Tahiti, February 2004.

[9] M. Dacier, F. Pouget, and H. Debar. Leurre.com: On the advantages of deploying a large scale distributed honeypot platform. In *Proceedings of the E-Crime and Computer Conference 2005 (ECCE'05)*, Monaco, March 2005.

[10] DShield. Distributed Intrusion Detection System, www.dshield.org, 2007.

[11] F-Secure. Malware information pages: Allaple.a, http://www.f-secure.com/v-descs/allaplea.shtml, December 2006.

[12] C. Leita and M. Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.

[13] C. Leita and M. Dacier. SGNET: Implementation Insights. In *IEEE/IFIP Network Operations and Management Symposium*, April 2008.

[14] C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *RAID 2006, 9th International Symposium on Recent Advances in Intrusion Detection, September 20-22, 2006, Hamburg, Germany - Also published as Lecture Notes in Computer Science Volume 4219/2006*, Sep 2006.

[15] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.

[16] Maxmind. Ip geolocation and online fraud prevention, www.maxmind.com.

[17] Maxmind Product. Home page ot the maxmind company at http://www.maxmind.com.

[18] D. Moore, C. Shannon, G. Voelker, and S. Savage. Network telescopes: Technical report. *CAIDA, April*, 2004.

[19] S. Needleman and C. Wunsch. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. J Mol Biol. 48(3):443-53, 1970.

[20] Netgeo Product. Home page of the netgeo company at http://www.netgeo.com/.

[21] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. *Proc. ACM SIGOPS EUROSYS*, 2006.

[22] F. Pouget, M. Dacier, and V. H. Pham. Understanding threats: a prerequisite to enhance survivability of computing systems. In *IISW'04, International Infrastructure Survivability Workshop 2004, in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS 04) December 5-8, 2004 Lisbonne, Portugal*, Dec 2004.

[23] T. C. D. Project. http://www.cymru.com/darknet/.

[24] N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.

[25] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2006.

[26] E. Ramirez-Silva and M. Dacier. Empirical study of the impact of metasploit-related attacks in 4 years of attack traces. In *12th Annual Asian Computing Conference focusing on computer and network security (ASIAN07)*, December 2007.

[27] J. Riordan, D. Zamboni, and Y. Duponchel. Building and deploying billy goat, a worm detection system. In *Proceedings of the 18th Annual FIRST Conference*, 2006.

[28] I. M. Sensor. http://ims.eecs.umich.edu/.

[29] TCPDUMP Project. Home page of the tcpdump project at http://www.tcpdump.org/.

[30] The Metasploit Project. www.metasploit.org, 2007.

[31] VirusTotal. www.virustotal.com, 2007.

[32] T. Werner. Honeytrap. http://honeytrap.mwcollect.org/.

[33] M. Zalewski. Home page of p0f at http://lcamtuf.coredump.cx/p0f.shtml.

Appendix 1: Leurre.com Traffic Statistics

The tables here under reflect global statistics of the traffic gathered by Leurre.com sensors for each year, from 2004 until March 2008, and for the low-interaction honeypots only. The different fields in the colomns are as follows:

- Seq_Id and Description: the identificator of the port sequence created by the attackers on the honeypots and their description (e.g. |I |445T means ICMP followed by 445/TCP);
- Nr of Sources and %: the total number of sources and their relative weight in the global traffic observed that year;
- Top attacking countries: the main countries of origin where the sources seem to come from, according to Maxmind;
- Main platforms and Nr of subnets hit: the most targeted sensors for each port sequence (with a threshold of min. 7% of the sources observed on a sole sensor) and the number of different subnets (Class A) where those sensors belong to;
- Host distribution: the relative proportion of sources having hit respectively one of the three virtual hosts emulated by honeyd (H1 and H2 = windows, H3 = linux), and only for the most targeted sensors.

Id	Description	Nr Sources	%	Top Attacking Countries	Main Platforms Hit	Nr Subnets (class A)	Host Distribution [H1, H2, H3]
1	|445T	135,417	25	US: 16%, CS: 13%, DE: 8%	6, 20, 5, 10	4	[0.33; 0.33; 0.33]
3	|135T	107,901	20	TW: 18%, US: 13%, DE: 10%,	20, 6, 8, 10	4	[0.34; 0.32; 0.34]
4	|I	33,730	6	US: 11%, CN: 9%	5, 20, 23, 10, 6, 9	5	[0.37; 0.39; 0.23]
7	|1026U	26,883	5	US: 22%, GB: 11%, DE: 10%	4, 5, 9, 13	3	[0.33; 0.33; 0.33]
8	|1027U	25,097	4	US: 22%, GB: 11%, DE: 10%	4, 5, 13, 9	3	[0.33; 0.33; 0.33]
23	|I|445T	12,574	2	DE: 13%, JP: 11%, US: 9%	20, 5, 10, 23, 6	5	[0.37; 0.41; 0.22]
2	|137U	12,477	2	US: 24%, CN: 8%, BR: 6%	20, 4, 6, 5	4	[0.33; 0.33; 0.33]
27	|139T	11,634	2	US: 24%, KR: 15%, TW: 12%	10, 6, 21, 1, 5, 8	5	[0.18; 0.25; 0.56]
21	|80T	11,615	2	US: 23%, CN: 20%, JP: 11%	1, 20	2	[0.73; 0.13; 0.13]
22	|5554T|9898T	11,586	2	CN: 62%, CA: 14%, HK: 10%	25, 10	2	[0.33; 0.33; 0.33]
15	|1433T	11,253	2	US: 26%, DE: 12%, KR: 11%	1, 21, 14, 4, 5, 9, 2, 20	5	[0.30; 0.34; 0.35]
40	|1025T	10,561	1	CN: 37%, KR: 13%, US: 10%	10, 20, 25	2	[0.34; 0.30; 0.35]
6	|1434U	10,443	1	US: 27%, CN: 14%, JP: 9%	5, 14, 20, 23, 6, 10	6	[0.33; 0.33; 0.33]
19	|9898T	9,363	1	CN: 79%	25, 10	2	[0.33; 0.33; 0.33]
17	|5554T|1023T|9898T	8,739	1	KR: 76%, CN: 18%	5, 21, 9, 13, 20	2	[0.27; 0.36; 0.36]
14	|135T|4444T	6,970	1	US: 24%, DE: 12%, FR: 10%,	1, 5, 10, 14, 20	5	[0.50; 0.25; 0.25]
52	|5554T	5,398	1	CN: 58%, CA: 8%	25, 10	2	[0.33; 0.33; 0.33]

Table 5. Global Statistics of the Main Port Sequences for 2004

Id	Description	Nr Sources	%	Top Attacking Countries	Main Platforms Hit	Nr Subnets (class A)	Host Distribution [H1, H2, H3]
3	|135T	198,239	20	US: 15%, DE: 11%, CS: 8%	20, 6, 31, 8, 21, 27	3	[0.35; 0.33; 0.31]
1	|445T	191,902	19	CS: 24%, US: 11%, DE: 8%	6, 20, 31, 9	3	[0.31; 0.31; 0.37]
7	|1026U	97,368	10	US: 31%, JP: 8%	44, 31, 26, 23, 39, 9, 49	3	[0.33; 0.33; 0.33]
27	|139T	61,910	6	US: 12%, CS: 11%, KR: 8%	6, 20, 31, 40, 23, 9	3	[0.32; 0.31; 0.37]
15	|1433T	45,497	4	CN: 34%, US: 14%, KR: 8%	20, 25, 21, 27, 6, 32, 9	3	[0.34; 0.35; 0.30]
4	|I	35,776	4	US: 27%, CN: 10%, TW: 7%	9, 20, 31, 23, 6, 8, 27, 26	4	[0.46; 0.27; 0.26]
21	|80T	35,707	4	CN: 22%, US: 21%	25, 20, 8	1	[0.36; 0.33; 0.30]
8	|1027U	33,691	3	US: 37%, GB: 8%, DE: 7%	31, 26, 22, 13, 9, 4, 23, 6, 1, 21	4	[0.33; 0.33; 0.33]
6	|1434U	28,723	3	CN: 39%, US: 18%, JP: 9%	27, 14, 25, 6, 9, 21, 23, 31, 20, 13, 43	7	[0.33; 0.34; 0.32]
2	|137U	19,675	2	US: 21%, BR: 8%	20, 6, 31, 27, 9, 4	3	[0.35; 0.35; 0.29]
40	|1025T	18,281	2	CN: 40%, KR: 15%, US: 9%	20, 25, 8	1	[0.34; 0.39; 0.27]
176	|445T|135T|445T|135T	14,959	1	CS: 39%, DE: 8%	6, 31, 23	2	[0.48; 0.50; 0.01]
29	|4899T	12,423	1	KR: 30%, CN: 19%, US: 15%	6, 31, 9, 14, 8	2	[0.33; 0.33; 0.33]
22	|5554T|9898T	10,500	1	CN: 77%, CA: 9%	25	1	[0.33; 0.33; 0.33]

Table 6. Global Statistics of the Main Port Sequences for 2005

Id	Description	Nr Sources	%	Top Attacking Countries	Main Platforms Hit	Nr Subnets (class A)	Host Distribution [H1, H2, H3]
7	|1026U	274,711	21	US: 48%	57, 27, 34, 9, 54	4	[0.33; 0.33; 0.33]
4	|I	237,953	18	US: 31%, CN: 18%	50, 57	1	[0.37; 0.31; 0.31]
1	|445T	142,397	11	CS: 34%, US: 8%	6, 53, 71	2	[0.30; 0.28; 0.42]
3	|135T	89,542	7	US: 14%, CS: 11%, JP: 9%, DE: 8%	6, 20, 21, 27, 65, 56, 31	3	[0.34; 0.33; 0.33]
27	|139T	84,058	6	US: 10%, FR: 9%, KR: 8%	6, 55, 31, 49, 32, 20, 21, 27, 57, 54, 45, 9	2	[0.27; 0.34; 0.38]
15	|1433T	53,540	4	CN: 40%, US: 14%	25, 50, 9, 64, 45, 57	2	[0.34; 0.33; 0.33]
1478	|I|80T	41,756	3	US: 44%, CN: 21%, KR: 8%	50, 56	1	[0.44; 0.31; 0.25]
132	|5900T	32,275	2	US: 22%, CN: 12%, FR: 7%	64, 45, 25, 9, 41, 55, 57, 54, 6	2	[0.36; 0.37; 0.26]
21	|80T	30,401	2	US: 24%, CN: 11%, FR: 7%	41, 25, 50, 47	2	[0.33; 0.33; 0.33]
6	|1434U	30,197	2	CN: 44%, US: 15%, JP: 7%	almost all	10	[0.33; 0.33; 0.33]
427	|1026U|1027U|1028U	20,235	2	CA: 100%	57	1	[0.33; 0.33; 0.33]
2	|137U	18,078	1	US: 13%, BR: 10%	64, 31, 6	1	[0.33; 0.33; 0.33]
61	|22T	17,529	1	CN: 25%, US: 14%, KR: 10%	50, 60, 8, 54, 32, 56, 9, 27	1	[0.32; 0.32; 0.35]
29	|4899T	15,532	1	US: 19%, CN: 18%, KR: 16%	41, 31, 57, 9, 45, 6, 55, 54	2	[0.33; 0.33; 0.33]

Table 7. Global Statistics of the Main Port Sequences for 2006

Id	Description	Nr Sources	%	Top Attacking Countries	Main Platforms Hit	Nr Subnets (class A)	Host Distribution [H1, H2, H3]
4	|I	692,038	34	US: 19%, KR: 12%, CN: 10%	77	1	[0.36; 0.32; 0.32]
427	|1026U|1027U|1028U	192,440	9	CA: 100%	57, 84, 64	3	[0.34; 0.33; 0.33]
7	|1026U	175,890	9	US: 45%	34, 27, 9, 89	3	[0.33; 0.33; 0.33]
1	|445T	134,861	7	CS: 25%, RS: 23%	6	1	[0.30; 0.30; 0.39]
132484	|2967T	90,036	4	US: 21%, CN: 8%	25, 64, 57	2	[0.34; 0.33; 0.32]
3	|135T	88,723	4	JP: 13%, US: 11%, CN: 8%	6, 56, 71	3	[0.33; 0.33; 0.33]
132	|5900T	68,249	3	US: 22%, FR: 7%, KR: 6%	25, 64, 57, 41	3	[0.33; 0.33; 0.33]
15	|1433T	60,829	3	CN: 32%, US: 15%	25, 64, 77	2	[0.34; 0.33; 0.32]
1478	|I|80T	46,112	2	US: 56%, KR: 11%, CA: 7%	50, 77, 56	1	[0.37; 0.40; 0.23]
10337	|I|139T|445T	45,794	2	US: 15%, PL: 9%, TW: 7%	9, 21, 58, 45	3	[0.02; 0.02; 0.95]
160	|I|139T	44,601	2	KR: 48%, US: 9%	9, 21, 45	2	[0.34; 0.35; 0.31]
27	|139T	35,628	2	US: 15%, RS: 7%, KR: 7%	6, 64, 57	3	[0.27; 0.27; 0.45]
61	|22T	25,985	1	CN: 23%, US: 14%, KR: 8%	77, 50, 56	1	[0.33; 0.32; 0.34]
6	|1434U	25,801	1	CN: 41%, US: 15%, JP: 7%	almost all	12	[0.33; 0.33; 0.33]

Table 8. Global Statistics of the Main Port Sequences for 2007

Id	Description	Nr Sources	%	Top Attacking Countries	Main Platforms Hit	Nr Subnets (class A)	Host Distribution [H1, H2, H3]
4	|I	83,593	24	US: 25%, KR: 12%	77	1	[0.36; 0.31; 0.32]
427	|1026U|1027U|1028U	67,486	19	CA: 100%	57, 84, 64	3	[0.35; 0.32; 0.32]
7	|1026U	19,074	5	US: 43%, CA: 8%	89, 27, 57	3	[0.33; 0.33; 0.33]
433	|1027U|1026U|1028U	15,798	4	CA: 100%	84, 57, 64	3	[0.37; 0.31; 0.31]
3	|135T	15,324	4	JP: 16%, US: 13%, CN: 10%	56, 64, 32, 21, 87, 49, 27, 6	5	[0.33; 0.33; 0.33]
1	|445T	12,962	4	RS: 28%, US: 7%, DE: 7%	6, 64, 53	3	[0.32; 0.31; 0.37]
132484	|2967T	12,401	4	US: 22%, DE: 14%, PL: 7%, CN: 7%	25, 53, 64, 57, 41	4	[0.34; 0.33; 0.33]
132	|5900T	10,230	3	KR: 13%, US: 12%, CA: 7%	57, 80	2	[0.32; 0.38; 0.30]
21765	|1028U|1026U|1027U	9,841	3	CA: 100%	57, 64	2	[0.33; 0.33; 0.33]
15	|1433T	9,401	3	CN: 40%, PK: 11%, US: 9%	77, 87, 25, 64	3	[0.33; 0.33; 0.33]
3107	|1027U|1028U|1026U	8,294	2	CA: 100%	64, 84, 57	3	[0.33; 0.33; 0.33]
61	|22T	6,893	2	CN: 25%, US: 11%, KR: 7%	77, 56, 87	2	[0.33; 0.33; 0.33]
10337	|I |139T|445T	6,605	2	US: 21%, JP: 8%	59, 87, 21, 9, 58	4	[0.03; 0.03; 0.94]
1478	|I|80T	5,819	2	US: 54%, KR: 14%, CA: 7%	77, 56	1	[0.53; 0.30; 0.16]
466	|1028U	5,416	2	CA: 100%	84, 64	2	[0.10; 0.43; 0.46]
160	|I|139T	5,203	1	KR: 37%, US: 12%	9, 21, 58, 59, 87	4	[0.35; 0.34; 0.30]
6	|1434U	5,027	1	CN: 44%, US: 12%, JP: 7%	almost all	13	[0.33; 0.33; 0.33]
428	|1026U|1028U|1027U	4,483	1	CA: 100%	64, 57	2	[0.33; 0.33; 0.33]
2	|137U	4,300	1	US: 19%, BR: 8%, FR: 7%	64, 78	2	[0.33; 0.33; 0.33]

Table 9. Global Statistics of the Main Port Sequences for 2008 (as of March)

Appendix 2: Illustrative Queries

Query 1: Temporal evolution (by day) of attack cluster 17718

```
SELECT count(Source.Source_Id) as nbs, floor((cast(Large_Session.Begin_At as date)
 - to_date('01/12/06'))/1) AS Date_
FROM Source, Large_Session, Environment
WHERE Large_Session.Begin_At> to_date('01/12/06')
AND Large_Session.Begin_At<to_date('01/03/07')
AND Large_Session.Tool_Id =  '17718'
AND Large_Session.Source_Id=Source.Source_Id
AND Large_Session.Environment_Id = Environment.Environment_Id
GROUP BY floor((cast(Large_Session.Begin_At as date) - to_date('01/12/06'))/1)
ORDER BY floor((cast(Large_Session.Begin_At as date) - to_date('01/12/06'))/1), nbs
DESC
```

Query 2: Geographical location of the attackers

```
SELECT  Country AS Country, count(Large_Session.Source_Id) as nbs
FROM Large_Session, Environment , Info_Source_Maxmind
WHERE Large_Session.Begin_At > to_date('01/12/06', 'DD/MM/YY')
AND Large_Session.Begin_At < to_date('01/03/07', 'DD/MM/YY')
AND (Large_Session.Source_Id=Info_Source_Maxmind.Source_Id)
AND Large_Session.Tool_Id = '17718'
GROUP BY Info_Source_Maxmind.Country
ORDER BY nbs DESC
```

Query 3: Attackers Domain names

```
SELECT REGEXP_SUBSTR(dom_name, '\.[^.]*$') as Domain, count(Source.Source_Id) NBS
FROM Source, Domain, Large_Session
WHERE Large_Session.Begin_At > to_date('01/12/06', 'DD/MM/YY')
AND Large_Session.Begin_At < to_date('01/03/07', 'DD/MM/YY')
AND Source.Source_Id = Large_Session.Source_Id
AND Source.Domain_Id= Domain.dom_id AND Large_Session.Tool_Id = '17718'
AND REGEXP_LIKE(dom_name, '\.[^.]*$')
GROUP BY REGEXP_SUBSTR(dom_name, '\.[^.]*$')
ORDER BY NBS DESC
```

Query 4: Attackers Subnets Information

```
SELECT REGEXP_SUBSTR(INET_NTOA(Source.Ip_Address),'^\d{1,3}') as ClassA,
 count(Source.Source_Id) as NBS
FROM Source, Large_Session
WHERE Large_Session.Begin_At > to_date('01/12/06', 'DD/MM/YY')
AND Large_Session.Begin_At < to_date('01/03/07', 'DD/MM/YY')
AND Source.Source_Id = Large_Session.Source_Id
AND Large_Session.Tool_Id = '17718'
AND REGEXP_LIKE(INET_NTOA(Source.Ip_Address),'^\d{1,3}\..*')
GROUP BY REGEXP_SUBSTR(INET_NTOA(Source.Ip_Address),'^\d{1,3}')
ORDER BY NBS DESC
```