Institut Eurécom

Corporate Communications Department

2229, route des Crêtes

B.P. 193

06904 Sophia Antipolis

FRANCE

Research Report RR-04-104

White Paper: Honeypot Platform: Analyses and Results[1]

October 30, 2004

Fabien Pouget, Marc Dacier

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

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

White Paper:
"Honeypot Platform: Analyses and Results[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

This report is part of the Eurecom research contract with France Telecom R&D. With respect to the deliverables characteristics, we present in this paper the first concrete results extracted from the honeypot platform. We take this opportunity to present the clustering approach we are using as part of our analysis. Based on it, we show that interesting information can be learnt from the attack processes. This information is of great value for getting a better understanding of current threats. We intend in a near future to integrate this value into correlation tools.

1. Introduction

This report is part of the Eurecom research contract with France Telecom R&D. In some previous presentations and deliverables, a honeypot architecture and its installation processes have been presented. A dedicated database has been built to efficiently store the data and to refine information by means of external utilities.

This database is now accessible from a web server hosted at Eurecom. Its URL is http://riviera.eurecom.fr. From this interface, users can directly access the SQL database or send requests through the GUI. A few dozens of interesting queries are also proposed to help the user

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

querying efficiently the interface and finding interesting information on the observed attacks. We invite the interested readers to contact the authors for more information on it. A specific account has been opened for France Telecom R&D.

In this report, we present the results that have been obtained so far. We show that the installed honeypot platform is targeted by a few numbers of tools. We also show that some results presented in our earlier publications ([DaPD04, DPDe04]) are partially true, as we have made use of partially correct tools. Finally, the existence of attackers and scanners communities is highlightened with simple experiments. As a global conclusion, these first results clearly show the interest of developing such a platform. It provides meaningful information that can be hardly obtained with other existing techniques. This work is still in-progress and other analysis methods will be developed in near future.

The report is built as follows: In Section 2, we provide a short reminder of the architecture and we give a big picture of the whole database. In Section 3, a clustering technique to analyze data is detailed. In Section 4, we conduct an in-depth analysis thanks to this last approach, and resulting conclusions are drawn. Section 5 concludes this report.

2. Setting & First Results

2.1 The Observation Platform

Our environmental setup consists in a virtual network built on top of VMWare [VMware] to which three virtual machines, or guests in the VMWare terminology, are connected (a Windows 98 workstation, a Window NT server and a Linux RedHat 7.3 server). It is very important to understand that the setup is such that these machines can only be partially compromised. They can accept certain connections on open ports but they can not, because of a firewall, initiate connections to the outside world. Furthermore, as they are built on non-persistent disks [VMware], changes are lost when virtual machines are powered off or reset.

We do reboot the machines very frequently, almost on a daily basis, to clean them from any infection that they could have had. The three machines are attached to a virtual Ethernet switch. ARP spoofing is implemented so that they can be reached from the outside world. A fourth virtual machine is created to collect data in the virtual network. It is also attached to the virtual

switch (what VMWare calls a switch is in fact a hub) and tcpdump is used as a packet collector [TCPdump]. In addition, a dedicated firewall controls the access to all machines (iptables [IPtables]) and tripwire regularly checks integrity of the host files. More detailed information about the setup can be found in [DaPD03].

2.2 Data Collection and Storage

In the following, we will make use of the expressions *attack source* and *ports sequence* which we define as follows:

- *Attack Source:* an IP address that targets our environment within one day. The same IP address seen in two different days counts for two different *attack sources* (see [DPDe03] for more on this).

- *Ports Sequence:* an ordered list of ports targeted by an attack source. For instance, if source A sends requests on port 80 (HTTP), and then on ports 8080 (HTTP Alternate) and 1080 (SOCKS), the associated *ports sequence* will be {80;8080;1080}.

All network packets are stored in a database and enriched with geographical information of the sources as well as their OS fingerprints. The database is made of several tables that are optimized to efficiently handle complicated queries. In other words, the computing cost of querying the database is marginally influenced by the number of logs in it. Of course, the price to pay is increased disk and memory space to handle redundant meta information (keys, counters, etc.) in several tables. We report the interested reader to [DPDe03] for more information on the design of the database.

Data is sent every day through a secure connection from the honeypot environment to a data server and then, is used to feed a specific database. This data server was carefully designed in a preliminary step. Its entity-relationship scheme is quite complex, and we give in figure 1 a simplified overview of its structure.

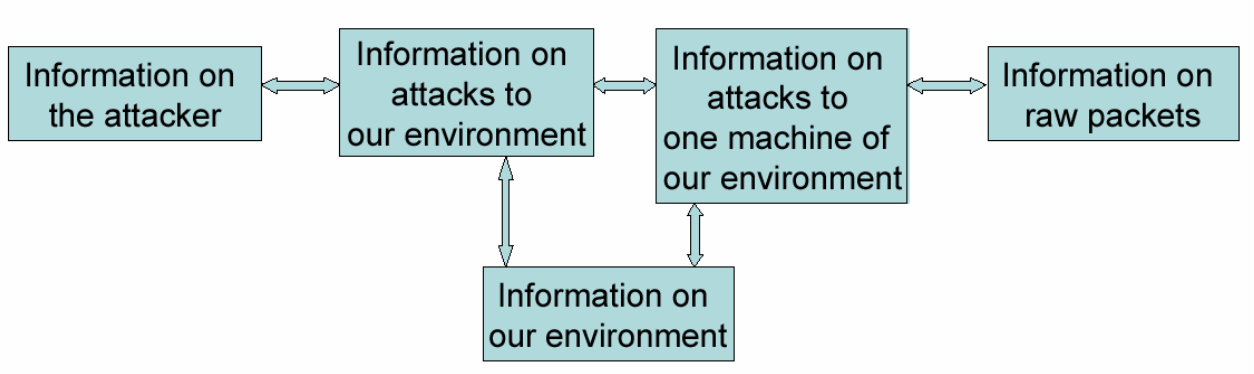

Figure 1: Simple database structure scheme

Information is grouped into five main categories, and collected data is enriched by additional information such as:

- The IP geographical localization of packets' source addresses, obtained with tools such as Netgeo [Netgeo]
- Passive OS fingerprinting on tcpdump logs, thanks to tools such as p0f [Pof] and Disco [Disco]
- Well-known blacklists of IP addresses
- An enriched precision on attack date (correlation with working days, working hours, holidays, etc)
- A honeypot environment description
- An analysis of observed sources' ports
- Other attack features: For instance, are honeypot machines attacked in parallel or in sequence?

The interested reader can directly contact the authors for more details on the database. An access to its interface has been specifically opened to France Telecom R&D.

2.3 Results: The big picture

In our past publications, we have shown and discussed early results obtained by querying our database and applying the clustering algorithm on a database containing 10 months of data. With 80% more data, we are still able to confirm all results obtained so far. This apparent stability over a long period of time is extremely encouraging for those who wish to obtain macroscopic analytical models of the Internet threats. Of course, as we will see here after, things are much

more complicated and less stable when one tries to dig into the data to better understand the root causes of these stable processes.

To highlight this regularity, we provide a few key figures obtained over this 18 months period and we refer the reader to our previous work ([DaPD04,DPDe04]) to verify by himself how regular these numbers are:

- 4102635 packets from/to our virtual machines have been stored in the database.
- 45360 attack sources have been observed.
- Only 205 different ports have been probed.
- Only 604 different ports sequences have been observed
- All attack sources, but a few very rare exceptions, have sent requests during less than 30 seconds.
- In 71% of the cases, attacks sources have sent requests to the 3 honeypots, always in the same order.
- In 6%, attack sources have contacted only two out of the three machines.
- In 23% they have focused on only one of the three honeypots.
- 99.5% of the IP addresses have been seen in only one day.

The database we have developed offers a large panel of information types. We report the interested reader to [DaPD04] for some results we have obtained by means of simple queries against the database. One major observation we have made was the following: very few ports were targeted over the experiment period. From February 2003 to February 2004, only 195 different ports have been probed. Each attacking machine probes one or more targeted ports following a given *Ports Sequence*. The number of different observed sequences is limited to 485 distinct sequences. Furthermore, we have observed two important points:

- Each sequence is often limited to one port.
- A given set of ports almost always uniquely identifies a *ports sequence*, as we have very rarely observed two sequences differing only by the order of ports being probed.

Table 1 taken from [DaPD04] represents, per month, the top 8 *ports sequences* performed by the attack sources observed during one month. For instance, one can see that in March, 45.5% of the attack sources have sent packets to the sole port 445, while 2.7% have scanned sequentially ports

80, 57 and 21. These 8 sequences characterize the activity of about 75% of the attacks every month.

March 2003	April 2003	May 2003	June 2003
445 (45.5%)	445 (43.5%)	445 (40.1%)	445 (30.6%)
80 (15.9%)	80 (15.4%)	80 (15%)	80 (15%)
1433 (8.8%)	1433 (7.6%)	1433 (8.8%)	139 (9.9%)
139 (5.8%)	139 (7.5%)	139 (7.1%)	1433 (8.2%)
21 (3.5%)	21 (3.3%)	21 (3.3%)	445,139 (4.4%)
135 (2.9%)	135 (2.2%)	135 (3.2%)	139, 445 (3.7%)
80, 57, 21 (2.7%)	80, 57, 21 (1.4%)	80, 57, 21 (2.2%)	21 (3.3%)
443 (2.1%)	443 (1.4%)	139, 445 (2%)	135 (3.2%)
Others (12.8%)	Others (17.7%)	Others (18.3%)	Others (21.7%)
July 2003	**August 2003**	**September 2003**	**October 2003**
445 (29.5%)	445 (23.6%)	80 (15.8%)	80 (15.6%)
80 (20.5%)	80 (17%)	1433 (12.6%)	1433 (11.9%)
1433 (9.6%)	139 (11%)	445 (12.6%)	139 (10.6%)
139 (8.1%)	1433 (9.5%)	139 (11.6%)	135 (9.8%)
21 (3.1%)	135 (5.8%)	135 (7.8%)	445 (8.4%)
139, 445 (2.9%)	139, 445 (4.1%)	554 (5.1%)	27374 (6%)
135 (2.5%)	17300 (3.4%)	139,445 (4.2%)	139,445 (5.2%)
443 (1.6%)	21 (3.2%)	21 (4.2%)	135, 4444 (5.2%)
Others (22.2%)	Others (22.4%)	135, 4444 (3.8%)	21 (4.4%)
		Others (22.3%)	Others (22.9%)
November 2003	**December 2003**	**January 2004**	
135 (28.9%)	135 (28.4%)	135 (23.5%)	Others ≈
135, 4444 (13.6%)	135, 4444 (17.3%)	6129 (14.6%)	185 distinct sequences each month
80 (9.2%)	80 (9.4%)	135, 4444 (14.5%)	
1433 (8.7%)	1433 (8.5%)	1433 (8.4%)	
139 (8%)	139 (8.4%)	139 (6.3%)	
445 (6.4%)	445 (5.4%)	445 (5.7%)	
21 (3.5%)	139, 445 (2.6%)	80 (4.3%)	
554 (2.9%)	21 (2.1%)	139, 445 (2.5%)	
Others (20.8%)	Others (17.9%)	Others (20.2%)	

Table 1: Percentage of ports sequences per month

3. A New Analysis Method

3.1 Introduction

These results show that only a few *ports sequences* are observed each month and that they represent a large volume of traffic. However, it is important to determine if they have the same *root causes* [Para88]. Indeed, they may cover up more subtle and rare attacks because of the volume of data they represent in the honeypots logs.

We call a *root cause* the most basic cause that can be reasonably identified as the origin of the attack on a given *ports sequence*. A *root cause* can be reasonably associated to one *attack tool*, or at least to one of its configuration.

Thus, we propose in this Section 3 to identify the most basic factors that characterize these *root cause*s. Julisch et al. have called such an approach a *Root Cause Analysis* (RCA) in [Juli03].

The idea here is the same but the context is different: their goal was to eliminate semi automatically clusters of false alarms while we are interested in identifying attack tools.

Furthermore, they were looking at intrusion detection alarms while we consider raw network packets sent against honeypots. Last but not least, the clustering algorithms used differ: they used hierarchical tree classifications while we choose association rules [Juli03, Agra93].

The *root causes* we get are grouped into clusters: one cluster is supposed to characterize one *root cause.* Section 3.3 shows that this statement is not totally exact. As a consequence, we propose a refinement of the algorithm to get a better *root cause* identification.

Finally, if such *root causes* are efficiently identified, we can imagine new techniques to enrich their identification and to ease correlation of alerts issued by intrusion detection systems in production networks. Indeed, if frequent attacks are correctly identified, we can eliminate the ambient noise they produce in log files to put all efforts in rarer phenomena which require special attention.

This paper aims at showing that *ports sequences* do not describe one and only one *attack tool* in a univocal way. In addition, we propose a simple technique that efficiently identifies such *attack tools* (or *root causes*).

3.2. Root Causes Identification

3.2.1 Accurate parameters

We have observed many repetitive attacks in our honeypot logs. As explained in 3.1, we propose to identify them by means of simple features. Our database offers many kinds of information as presented in 2.2. We have made several tests to find the most relevant ones.

For the sake of concision, we only present in the following three of them. They give condensed and interesting results, in regards of the experiments we made with other parameters. They are listed below:

- T: the number of machines in our environment targeted by one source
- n_i: the number of packets sent by one source to machine i.
- N: the total number of packets sent by one source to the whole environment. In other words, $N = \sum n_i$, with $i \in \{1,2,3\}$.
- Duration of the source observation
- Average time between requests sent by the attack source.

Some of these parameters are generalized by means of a hierarchy algorithm.

As said in Section 3.1, we want to determine the *root causes* of frequent processes observed in our honeypot environment. Many techniques exist to do *Root Cause Analysis* [Lati02, Liv01, Para88]. At this stage, the algorithm choice is not as important as the results we expect to get from it. Therefore, we first propose to apply a simple one. If results are encouraging, more complex techniques will be applied to refine them, if needed.
The method we apply is described in the following subsection and results are given in subsection 3.2.3.

3.2.2 Association Rules

- AR Motivations

We determine T, n_i and N for each attack source associated to a given *ports sequence*. They are then inserted in a new database table. To make a long story short, there is one table per *ports sequenc*e, with each column being one parameter {T, n_i, N} and each line being information on one *attack source*. The idea consists in searching for interesting relationships between the items contained in each table. One simple but widespread solution consists in applying association rule (AR) mining.

- AR Applications

Association rules are powerful exploratory techniques which have a wide range of applications in many areas of business practice and research - from the analysis of consumer preferences or human resource management, to the history of language [Agra93]. For instance, these techniques enable analysts and researchers to uncover hidden patterns in large data sets for so-called *market basket analysi*s, which aims at finding regularities in the shopping behavior of customers of

supermarkets. With the induction of association rules one tries to find sets of products that are frequently bought together, so that from the presence of certain products in a shopping cart one can infer (with a high probability) that certain other products are present.

The usefulness of this technique to address unique data mining problems is best illustrated in a simple example: “If a French customer buys wine and bread, he often buys cheese, too”. It expresses an association between (set of) *item*s. This association rule states that if we pick a customer at random and find out that he selected certain items, we can be confident, quantified by a percentage, that he also selected certain other items. The standard measures to assess association rules are the *support* and the *confidence* of a rule, both of which are computed from the *support* of certain item sets. Some additional rule evaluation measures are sometimes considered, but they are out of the scope of this work. Moreover Han et al. have classified association rules in various ways, based on some criteria (type of values handled in the rule, dimensions of data involved in the rule, levels of abstractions involved in the rule set, etc) [Han01]. We report the interested reader to the seminal paper [Agra93] for more information on AR evaluation methods. The *support* and *confidence* indices are presented below. Many algorithms currently exist to restrict the search space and check only a subset of rules, but if possible, without missing important rules. Generally speaking, association rules mining is a two-step process:

- Find all frequent itemsets: by definition, each of these itemsets will occur at least as frequently as a pre-determined minimum support count.
- Generate strong association rules from the frequent itemsets: By definition, these rules must satisfy minimum support and minimum confidence.

We use the algorithm presented by Agrawal et al. in [Agra94], which is called *the Apriori algorithm*.

Thus, for a given rule R: A and B -> C, we consider two parameters:

- *Support(R)* = the support is simply the number of transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

$$Support(R) = \frac{\#transactions(containing A, B, C)}{\#transactions}$$

- *Confidence(R)* = is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

$$Confidence(R) = \frac{\#transactions(containingA, B, C)}{\#transactions(containingA, B)}$$

- AR Interesting Outputs

Association rule mining consists in finding frequent itemsets (set of items, satisfying the *minimum support threshol*d), from which strong association rules are generated. These rules also satisfy a *minimum confidence threshold*.

The *Apriori* algorithm is applied to each *ports sequence* table in our database. Table 2 gives some resulting frequent itemsets with their respective support. We only give some of them for conciseness concerns. We provide for 11 sequences some frequent itemsets as well as their corresponding support. These *ports sequences* are the most frequent ones, as observed in [DaPD04]. The 'Total Number of clusters' columns provides the number of obtained clusters, with support higher than 1%.

For instance, we find 5 important clusters for *ports sequence* {445}. Three of them represent attacks on the sole port 445, having the following common features:

- *Cluster Nb* 1: Three virtual machines are targeted. One packet is sent to the first machine and three to the two others.
- *Cluster Nb* 2: One machine only is targeted and it receives strictly one packet.
- *Cluster Nb* 3: Three machines are targeted and the attack source sends a total of eight packets

These three clusters stem for almost 90% of attacks with *ports sequence* equal to {445}.

Furthermore they are mutually exclusive (an attack cannot belong to two different clusters).

Ports Sequences	Some frequent itemsets	Support (%)	Total number of clusters (support> 1%)
{445}	Cluster 1: T = 3 & N = 9 (n1=3 & n2=3 & n3 = 3) Cluster 2: T= 1 & N = 1 Cluster 3: T = 3 & N = 8	65.5% 18.4% 6.1%	5
{135,4444}	Cluster 4: T = 3 & N = 13 (n1 = 3 & n2 = 8 & n3 = 2) Cluster 5: T = 3 & N = 26 (n1 = 6 & n2 = 16 & n3 = 4)	66.3% 33.6%	3
{80}	Cluster 6: T = 1 & N = 3 Cluster 7: T = 3 & N = 11 (n1 = 3, n2 = 5, n3 = 3) Cluster 8: T =1 & N = 1	55.8% 18.7% 5.6%	6
{1433}	Cluster 9: T = 3 & N = 9 (n1=3 & n2=3 & n3 = 3) Cluster 10: T = 3 & N = 3 (n1=1 & n2=1 & n3 = 1) Cluster 11: T = 3 & N = 12 (n1=4 & n2=4 & n3 = 4)	53.9% 18.7% 18.1%	5
{139, 445}	Cluster 12: T = 3 & N = 38 (n1 = 18 & n2 = 18 & n3 = 2) Cluster 13: T =3 & N = 145 Cluster 14: T = 1 & N = 74	41.1% 7.6% 6.8%	18
{135}	Cluster 15: T = 3 & N = 13 (n1 = 3 & n2 = 7 & n3 = 3) Cluster 16: T = 3 & N = 21 (n1 = 3 & n2 = 15 & n3 = 3)	50.1% 15.3%	4
{17300}	Cluster 17: T = 3 & N = 3 (n1 = 1 & n2 = 1 & n3 = 1) Cluster 18: T = 3 & N = 6 (n1 = 2 & n2 = 2 & n3 = 2) Cluster 19: T = 3 & N = 9 (n1 = 3 & n2 = 3 & n3 = 3)	39.8% 34.6% 25.5%	9
{21}	Cluster 20: T = 3 & N = 16 (n1 = 1 & n2 = 1 & n3 = 1) Cluster 21: T = 3 & N = 11 (n1 = 1 & n2 = 1 & n3 = 1) Cluster 22: T = 3 & N = 7 (n1 = 1 & n2 = 1 & n3 = 1) Cluster 23: T = 1 & N = 11 (n1 = 1 & n2 = 1 & n3 = 1)	23.4% 13.7% 10.6% 9.1%	25
{80, 57, 21}	Cluster 24: T = 3 & N = 40	65.1%	5
{554}	Cluster 25: T = 3 & N = 8 Cluster 26: T = 1 & N = 1 Cluster 27: T = 3 & N = 9	50.8% 29.8% 7.4%	6
{139}	Cluster 28: T = 3 & N = 9 (n1=3 & n2=3 & n3 = 3) Cluster 29: T = 2 & N = 8 Cluster 30: T = 1 & N = 4	49.2% 9.5% 6.3%	6

Table 2: Some frequent itemsets for 11 ports sequences

By choosing a support higher than 1%, we have built 92 clusters for the presented *ports sequence*s. They represent 87 % of the attacks we have observed during the one year experiment.

We have applied this method for all ports sequences, and we have selected relevant rules to create exclusive clusters.

A first major observation is that there can be several clusters associated to a single *ports sequence*. Thus, *ports sequences* cannot be directly used to identify *root cause*s.

Furthermore, this result clearly validates that most of the collected data is redundant and predictable insofar as a few clusters can be associated with a large number of observed attacks.

There are some other phenomena that must be clarified. First, for one given attack, we sometimes observe that virtual machines receive the same number of packets from the attack source. This is due to blind and simple scans on IP ranges. Secondly, there is a clear correlation between packets numbers and OSs/services running on our honeypot machines.

Obviously enough, attacks on open ports correspond to more important packets traffic (see *ports sequence* {135} for instance). Finally, some *ports sequences* such as {139, 445} have a larger number of associated clusters. The reason is that such attacks often correspond to high packets traffic: they target Windows machines and try to obtain a lot of information from these talkative netbios services. There might be some tcp retransmissions that make parameters value somehow different in terms of packets numbers. A clustering refinement would be possible in this case.

The previous clusters are obtained based on very simple parameters. They first show that basic *ports sequences* do not identify *root causes* in a unique way. Secondly, each cluster is now supposed to represent one *root* cause, or *attack too*l. We show in the following section this is partially correct by checking clusters' coherency. We then introduce another membership property to refine clusters.

3.2.3 Levenshtein distances

- Introduction

We have obtained interesting clusters in Section 3.2.2. They cover a large percentage of attacks, but we cannot be sure if they are *good* clusters or not. One can argue that there might be two different attacks generating the same amount of packets against the same *ports* sequences.

Clusters are built on very simple parameters, and their consistencies must be validated.

In order to validate clusters consistency, we consider packet data contents. The payloads of all packets sent from the same source are concatenated to form a simple text phrase, thanks to the

tethereal utility [Tethe04]). Tethereal is the command line version of the popular network traffic analyzer tool ethereal. It allows examining data from a capture file and browsing detailed information for each packet in a text format. Thus, we consider each phrase as a tethereal line, with '||' separators. Figure 2 gives a short phrase of an ftp attack for illustration.

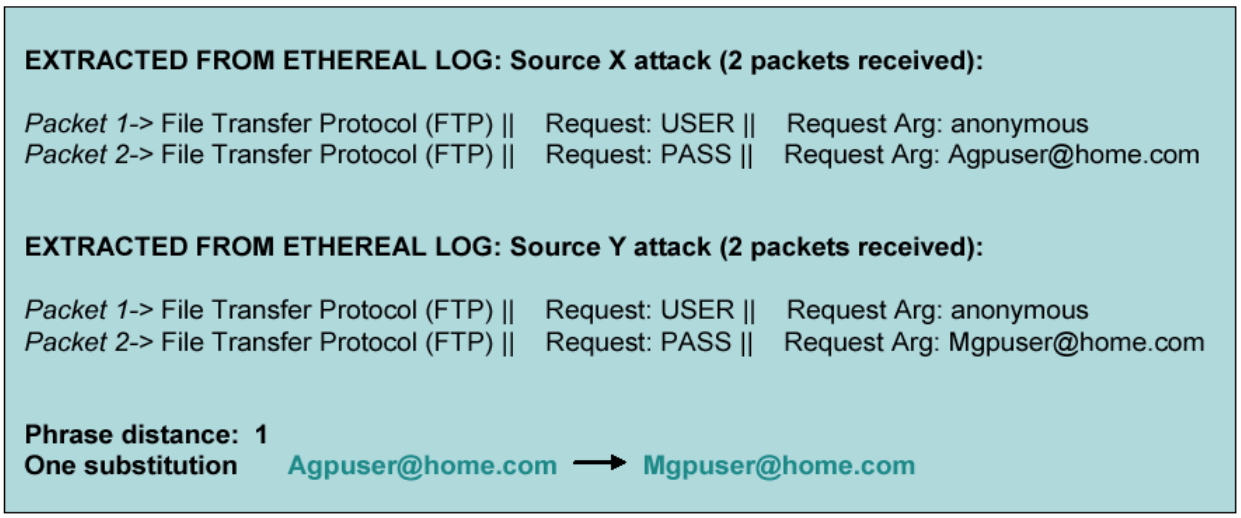

Figure 2: examples of phrases from an ftp attack

Each cluster gathers all *attack sources* that are assumed to be due to a single *root cause*, i.e. to the use of the same attack tool against our honeypots. We define for each *attack source* its associated *'attack phrase'*. Then, we compare for each attack of one given cluster the phrase distances to all others phrases of the same cluster. This technique is based on the Levenshtein edit distance which is explained below.

- Phrase distance approach

The Levenshtein distance (LD) algorithm has been used in many domains, such as spell checking, speech recognition, DNA analysis or plagiarism detection. It is a measure of the similarity between two strings, which we will refer to as the source string *(*s) and the target string *(*t) [Borg03]. The distance (sometimes called edit distance) is the number of deletions, insertions, or substitutions required to transform *s* into t. For example,

- If *s* is "Agpuser@home.com" and *t* is "Agpuser@home.com", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.

- If *s* is "Agpuser@home.com" and *t* is "Mgpuser@home.com", then LD(s,t) = 1, because one substitution (change "A" to "M") is sufficient to transform *s* into t.

In general the two components of the phrase distance (i.e. the string distance and the positional distance) can have a different cost from the default (that is 1 for both) to give another type of phrase distance. There is a third component: a cost that weights on the phrases that have less exact matches. It is described in details by Roger et al. in [Borg03]. This third component is disabled by default (i.e. it has a 0 cost), but it can be enabled with custom cost.

The method we apply sums the phrase distance from the words from the set (i.e. formed by the defined set of characters) and the phrase distance calculated from the "words" belonging to the complementary set. Moreover, the algorithm used to find the distance is the "Stable marriage problem" one [Gus89, Ksm01, Mathw] This is a matching algorithm, used to harmonize the elements of two sets on the ground of the preference relationships (in this case the string distance of single "words" plus the positional distance plus eventually the exact match weight.

- Results

The algorithm has been applied to clusters built in Section 2.2.3. For a given cluster, we compare all *attack phrases* and calculate the average distance. More concretely, let C be the set of *attack phrase*s, with |C| = n. Let D(a,b) be the phrase distance of a and b and i any *attack phras*e. Then, the average distance from i to other elements of C is:

$$d_i = \sum_{j \in C} \frac{D(i,j)}{n-1}$$

Thus, the average distance is defined as:

$$D_C = \sum_{i \in C} \frac{d_i}{n} = \sum_{\substack{i,j \in C \\ i<j}} \frac{2.D(i,j)}{(n-1).n}$$

Some results are presented in Table 3. The 'phrase distance' column gives the average distance D_C after having computed all intermediate values.

The standard deviation (square root of the *variance*) is a frequent measure to describe variability in data distributions. It is a measure of the average amount by which observations deviate on either side of the mean. Unlike the variance, the standard deviation describes variability in the original units of measurement. Its value is given in the 'Standard deviation' column.

Cluster	Ports sequence	Phrase distance	Standard deviation
Cluster 1	{445}	0	0
Cluster 2	{445}	0	0
Cluster 3	{445}	0	0
Cluster 4	{135, 4444}	0	0
Cluster 5	{135, 4444}	22	84
Cluster 9	{1433}	39	311
Cluster 10	{1433}	0	0
Cluster 11	{1433}	641	1213
Cluster 20	{21}	88	258
Cluster 21	{21}	0	0
Cluster 22	{21}	0	0

Table 3: Cluster coherency with Levenshtein distance

We observe from Table 3 that seven out of eleven clusters have a null phrase distance. They contain very similar *attack phrase*s. We say that they are coherent and correspond to one *root*

cause, i.e. to a single attack tool. The null value can be explained by the fact that these attacks are limited to simple and repetitive scans.

A deeper analysis of other clusters is required to understand the phrase distance value. It suffices to extract different patterns to obtain some information. There are two options: Either we observe very strong similarities and non zero phrase distances are simply due to packets random fields. In this case, we consider the cluster corresponds to one *root cause,* and the random fields are *attack tools* features. Or we find different patterns that have nothing in common. In this situation, we build new sub-clusters and reevaluate the new distance phrase.

In our examples, we have:

- Cluster 20 of *ports sequence* {21}: there are two anomalous attacks which bias the average phrase distance value. They are totally different from all the others. Once eliminated, we get a new average phrase distance of two. This is due to the random login name used by the attack tool on port 21 (see figure 2).
- Cluster 5 of *ports sequence* {135, 4444} is not null because of data payload differences. This *ports sequence* is associated to MBlaster attacks and some variants contain data payload in the first packets [Sym03].
- Cluster 9 and 11 of *ports sequence* {1433}: both can be split into two sub-clusters: those which contains data payload of 8 bytes (all zeros) and of 24 bytes (all zeros). These sub-clusters have a null phrase distance. The initial cluster was not totally correct.

We get similar results with other clusters. Very few of them have required a small refinement (sub-clustering). This validates that with simple parameters, we have built interesting clusters that characterize main attack types our honeypot environment is facing.

- Conclusion

We have presented a method to find root causes of frequent traffic met in the honeypots logs.

The *root cause analysis (RCA)* steps are:

- Find frequent *ports sequences* (table 1 in our example)
- Build clusters on simple parameters (such as those used in our example on Section 3.1)
- compute the average distance phrase and its standard deviation to validate the clustering phase

- Split the cluster in sub-clusters, if necessary, by recursively applying the phrase distance approach (see Cluster 9 and 11 of ports sequence {1433})
- Get one cluster per *root cause*. The *root cause* corresponds to a specific configuration of an *attack tool*.

This method validates that an in-depth analysis is a prerequisite to a better understanding of malicious activity collected by our honeypots. The apparent regularity we observed in [DAPD04, DPDE04] is *de facto* due to various *root causes* which are precisely identified by our method.

Moreover, the obtained results can be exploited in different ways. First, they can permit to filter honeypot logs and detect newly observed *root causes*. Secondly, they allow us to track *root causes* evolution. Finally, they may characterize some specific *attack tools* and help finding some new signatures of them. Three applications are presented in the following Section 3.2.4 and in Section 4.

3.2.4 Clusters applications

- Approximate number of attacks

We have obtained data reduction with simple clusters. Each of them groups attacks which can be associated to one *attack tool*. If we now create all possible clusters (same operation than in 3.2.3 but with min(Support)=0%) for the observed *ports sequences*, we obtain the – pessimistic- number of *attack tools* (tools fingerprints) we have observed during the experiment year.

attack tools = ∑#clusters(i), for each *ports sequence i* = 753.

This number is very conservative. Indeed, we consider all observed attacks. However, many *ports sequences* are not exactly due to attack tools on our machines. They can come from other phenomena such as backscatters [MoVS01].

Table 4 presents the quantitative number of obtained clusters. The first column takes into account the Association Rules (AR) clustering only, while the second column presents the two steps of our algorithm (Association Rules –AR- and phrase distances –LD-).

Support value	AR	AR + LD
> 0%	491 clusters	753 clusters
> 1%	92 clusters	152 clusters

Table 4: Quantitative number of clusters

Finally, this result is conservative insofar as we did not consider cases of packet losses and retransmissions. Clusters are built on received packet numbers, so attacks can be placed in two distinct clusters if losses occur despite their very strong similarity.

- Tools fingerprinting

By construction, each cluster can be potentially associated to one *attack too*l. This requires either a good knowledge of blackhat tools, or a short lookup in security advisories pages or incidents mailing lists [ISC03, Secu04]. Moreover, clusters must be analyzed in depth thanks to parameters described in Section 1.2.3. From the clusters presented above, we get the following information:
- Three clusters of *ports sequence* {135, 4444}: three variants of MBlaster worm.
Others seem to exist (see [Sym03]), but we did not observe them in our environment.
- Two clusters of *ports sequence* {554}: They correspond to two different configurations of an RTSP scanner [Rtsp]
- Cluster 1 of *ports sequence* {21}: it corresponds to a famous ftp scanner, named
Grim's Ping [GrimP] .
- Cluster 2 of *ports sequence* {21}: it can be associated to another ftp scanner running on Windows machines, named Roadkil's FTP Probe [Road04]
- Cluster 2 of ports sequence {139, 445}: it corresponds to an enumeration and penetration-testing scanner called the Leviathan Auditor [Lev04].
- Cluster 1 of ports sequence {1433}: this is due to another worm named SQLSnake – or any of its variants. It scans for open SQL Server 2000 [Snake].
- Cluster 2 of ports sequence {1433}: this corresponds to another scanner, named Sfind.exe [Sfind].
It suffices to test these tools in our environment to find their 'signatures'. If one attack has been observed from this tool, a cluster with similar 'signature' exists.
This method is all the more efficient that main attackers do not even take time to modify such tool 'signatures' (for instance, the default logins & passwords of ftp scanners).
Finding that attacks have characteristic signatures is not a breakpoint by itself. Rule-based intrusion detection systems are built on such observations. However, a honeypot environment permits to identify frequent observed signatures. Such attacks trigger lots of 'true positive alerts' and, when identified, are not of real interest for the busy administrator. It would be interesting to

extract such alerts, so that more specific alerts remain. This is even more interesting that administrators are often overwhelmed with alerts from many security systems.
In addition, a similar work must be performed on the honeypots data. After having extracted well-known observed attacks, a deeper look at rare and strange ones is compulsory to understand what happened. Are they due to a new tool? Are they specific to the honeypot environment? Analysis of data collected by honeypots is then reduced to the observation of a few phenomena. In practice, this makes honeypots more interesting to administrators that already have a large amount of data to deal with.
In the following Section, we detail some interesting results we have obtained so far. Most of them were obtained thanks to this approach detailed in the whole Section 3.

4. Applications and New Results

4.1 Geographical Information

Figure 3 shows the amount of attack sources observed per country and per month, from February 2003 to August 2004. The important thing to note here is the uniformity of the distribution. A few countries systematically compose the top 10 of the attacking countries: China, the USA, Japan, Taiwan, France, Germany, South Korea. Though, it is important to stress that we do note two important phenomena:
- A coordinated decrease of activities from the 7 countries in July 2003 that will be discussed in Section 4.3.
- A dramatic increase of activities from all countries observed from January to July 2004.

This cannot be easily explained by the appearance of a single worm. Indeed, the impressive peak from France and Germany in April and May 2004 is mainly due to Sasser activity. This worm attempts to exploit a bu_er over ow vulnerability in the Windows Local Security Authority Service Server (LSASS). It propagates by scanning random IP addresses on port 445/tcp. However, a deeper analysis of the impressive increase of USA activity reveals that a large number of these attacks target only one of our three honeypots, and on different ports. This is a new pattern of attack that we had not observed before. On one hand, this seems to contradict our

claim that regular patterns exist and that analytical models could therefore be proposed for the existing threats. On the other hand, we have been able to identify this new type of attack pattern because it did not fall within the regular pattern. Thus, despite this new unstable activity, we maintain that data sets obtained by means of simple honeypots are amenable for analytical modeling of the attack processes present on the Internet.

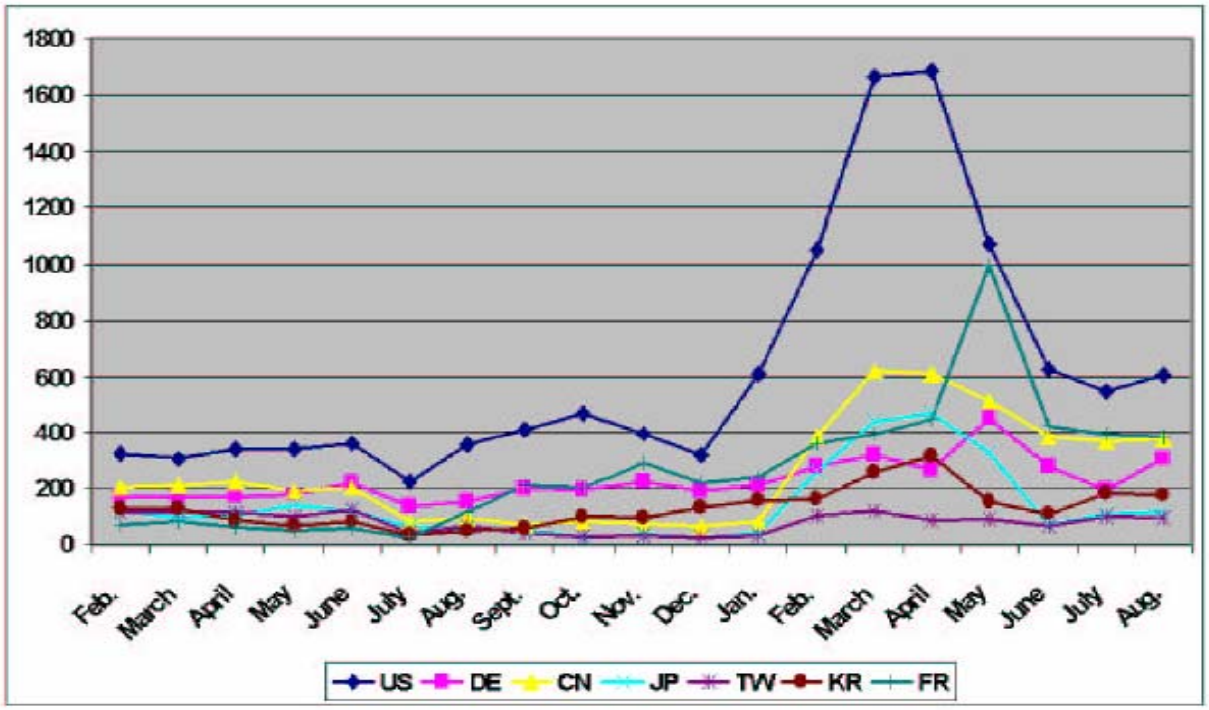

Figure 3: # of attack sources per country and per month

- NetGeo vs. MaxMind

We discussed in [DaPD04] some information on attacking machines, and more specifically on their geographical location. In that paper, we have presented results obtained thanks to the NetGeo utility, developed in the context of the CAIDA project [Caida]. NetGeo is made of a database and a collection of perl scripts that map IP addresses and AS numbers to geographical locations. This utility is open-source and has been applied in several research papers among which [Ng02, Ros03, Zeit03, Yoo02]. In our case, running NetGeo scripts on our data base indicated that 70% of the attacks originated from only three countries: Australia (27.7%), the USA (21.8%) and the Netherlands (21.1%). This was quite surprising but very consistent month after month. As a result of these first publications and discussions with our peers, we have decided to use another system that also provide geographical location, namely MaxMind

[Maxmind], to double check the results obtained with NetGeo[3]. Results are presented in Figure 4 for the top 4 attacking countries according to NetGeo: Australia, Netherlands, USA and 'unknown'. For all IP addresses that NetGeo indicated to us as belonging to one of these countries we have checked the results returned by MaxMind. The Figure must be read as follows: In the upper left pie chart, one can see that 29% of the IP addresses that NetGeo has located in Australia are located in China by MaxMind.

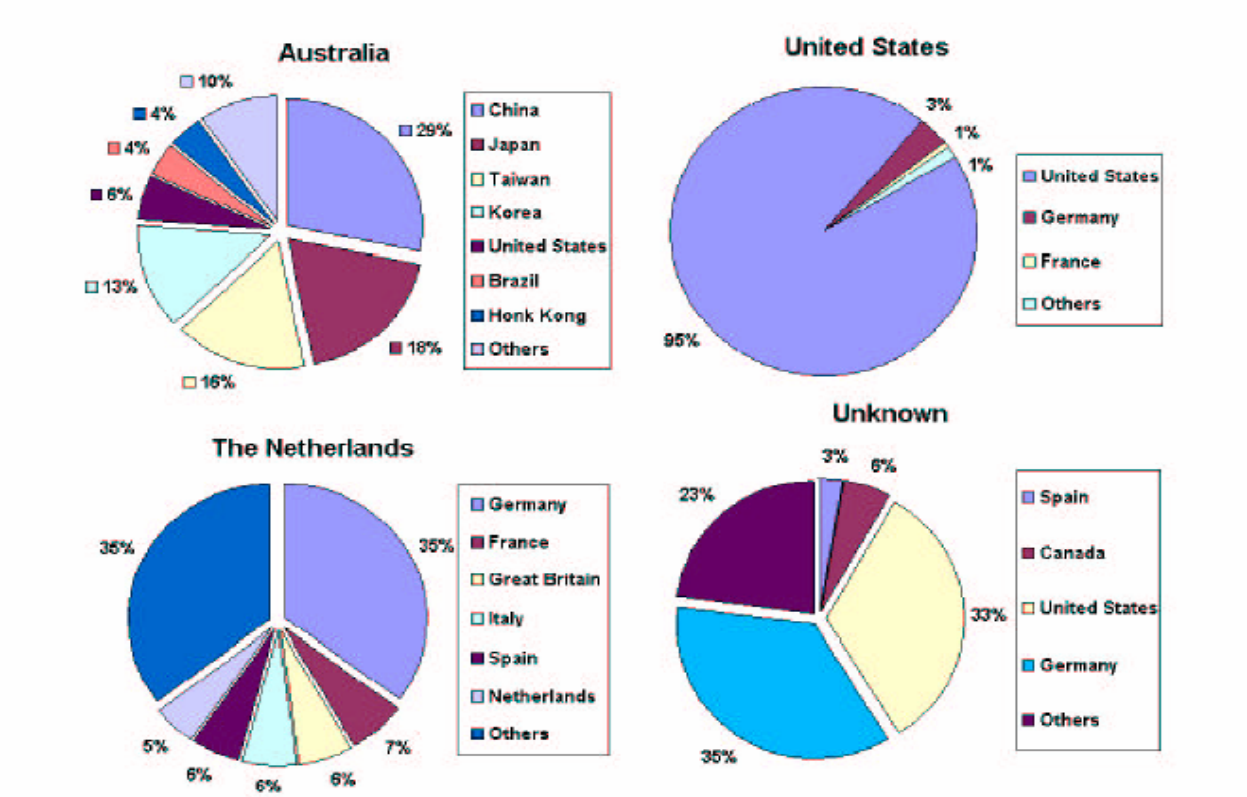

Figure 4: top 4 attacking countries according to NetGeo compared to MaxMind results

If we consider the results of MaxMind instead of those of NetGeo, our top 4 countries are now replaced by a group of 7 countries, namely: USA, Germany, China, France, Japan, Taiwan and Korea (see table 4). This seems to better fit with the expectations of the security community. For Australia and the Netherlands, MaxMind gives completely di_erent results than NetGeo. According to MaxMind, almost no attacks are originating from Australia anymore but they, instead, seem to come from four other countries: China, Japan, Taiwan and South Korea. Moreover, the Netherlands are replaced by two main countries: Germany and France. It is also

[3] figure 3 has been obtained thanks to MaxMind. The corresponding curve from NetGeo can be found in [DaPD04].

worth pointing out that, in our data set, both tools disagree for 61% of all IP addresses. Also, even if they agree, for the total amount of IP addresses located in the USA, 21.8% vs. 22.3%, there are important differences regarding the specific state within the USA where they locate the IP addresses. Finally, the evaluation of two other commercial utilities, namely IP2Location and ActiveTarget, validates MaxMind results over NetGeo as they have returned same results for the very same IP entries [IP2loc,ActTarg].

The explanation of these differences lies in the precise understanding of the information returned by both tools. The structure of Internet is based on inter-connected autonomous systems (ASs), where each AS is administrated by a single authority with its own choice of routing protocols, configuration, and policies. For a few large, geographically dispersed ASs, NetGeo returns the geographical location of the authority of the AS to which belongs the IP instead of the real location of the IP itself. This is what happens for RIPE (Netherlands) and APNIC (Australia). However, this is not what NetGeo is supposed to return, as described in [Caida]. At this time of writing, this apparent erroneous behavior is not clear to us and might lead to misleading information[4]. The NetGeo web page states that NetGeo has not been actively maintained for several years and this will probably not change in the foreseeable future.(...) Please be warned that NetGeo may give widely incorrect results". Unfortunately, many research papers still refer to this tools for IP geographical analysis [Ng02, Ros03, Zeit03, Yoo02]. On the other hand, according to one of the MaxMind representatives, their system uses user-entered location data aggregated with 'whois' data to better estimate the locations of the end users. These data are obtained thanks to data exchange agreements with some of their customers.

The most important consequence of using the information returned by MaxMind instead of NetGeo is that Australia, which was, by far, our top attacking country now disappears from the map.

[4] 06/28/2004, Bugtraq mailing list: information on the Scob Trojan indicates that IP addresses of infected machines are mostly located in the USA and Australia [34]. The author indicates that the information is based on APNIC; it is quite likely that here too Australia is blamed for no good reason ...

Countries	NetGeo	MaxMind
Netherlands	21.1	1.1
US	21.8	22.3
Australia	27.7	0.6
China	1.5	10.1
France	4.1	6.1
Germany	1.9	12.0
Japan	0.2	5.7
Taiwan	0.7	5.5
South Korea	0.7	4.8
Italia	1.1	2.7
Great Britain	0.0	2.5
Spain	0.5	2.0
Others	18.7	24.6

Table 4: NetGeo vs. MaxMind against our honeypot data (% of observed attack sources per country)

4.2 Deloder worm and aftermath

One of the questions left unanswered in [DPDe04] was the apparent decrease of attacks coming from Australia around July 2003. This is represented in Figure 5. Figure 6 represents the same curves, based on MaxMind data, for all addresses identified as Australian ones by NetGeo.

We notice that attacks are coming from four Asian countries, respectively China, Japan, Taiwan and South Korea. We were expecting to find the explanation of the 'Australian' decrease in a change due to one of these countries, as a consequence, for instance to some increased control of trafic owing out of the country. However, Figure 6 shows that the decrease exists for all countries in mid-2003. In addition, we have applied the clustering algorithm presented in Section 3 which shows that the decrease is mainly due to a few specific clusters, many of them involving ports sequence {445}. The tool associated to these clusters has been identified as being the Deloder worm [Delo1, Delo2]. In Figure 7, we represent, per month and per country, the amount of attack sources compromised by the Deloder worm that have tried to propagate to our honeypots.

Each represented cluster can be linked to one of its variants. This worm, which spreads over Windows 2K/XP machines, attempts to copy and execute itself on remote systems, via accessible

network shares. It tries to connect to the IPC$ share[5] and uses specific passwords. According to antivirus websites like [Delo1, Delo2], this worm, which was initially detected in March 2003, originates from Asia, and more especially China. This is consistent with our curves.

The surprising fact comes from the rapid decrease of its propagation around July 2003. [Moo02] mentions that the shutdown of CodeRedII was pre-programmed for October 1, 2001. [Sym04] mentions that Welchia worm will self terminate on June 1st, 2004, or after running 120 days. A similar mechanism could have been used for Deloder but, as far as we know, no one has ever made mention of it publicly.

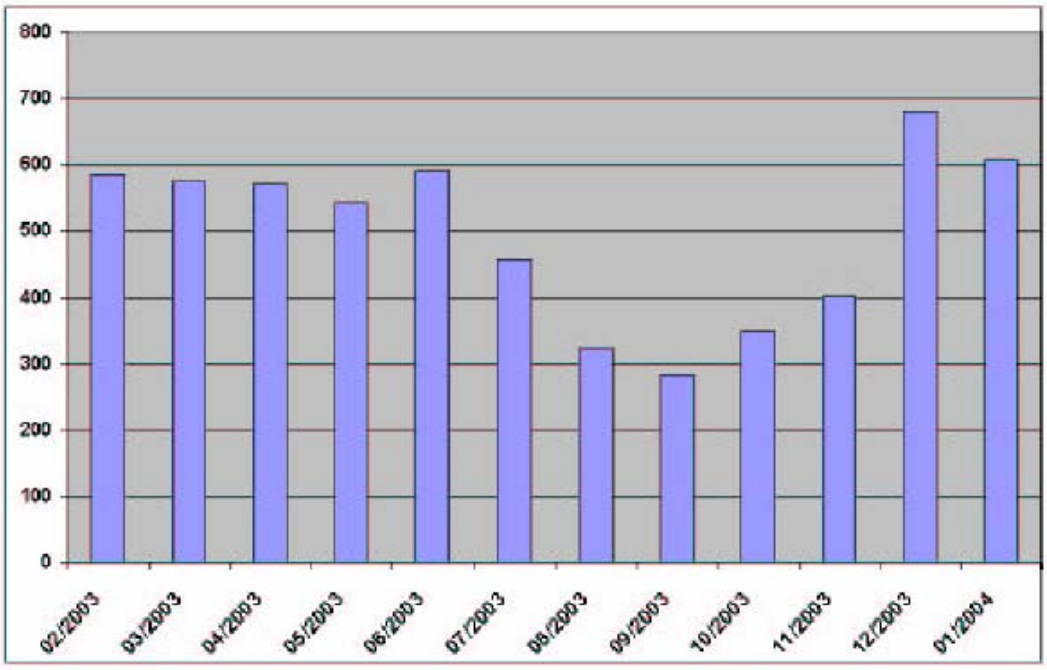

Figure 5: Australian attacks observed each month, based on the NetGeo utility

[5] or ADMIN$, C$, E$ shares depending on the Deloder variants

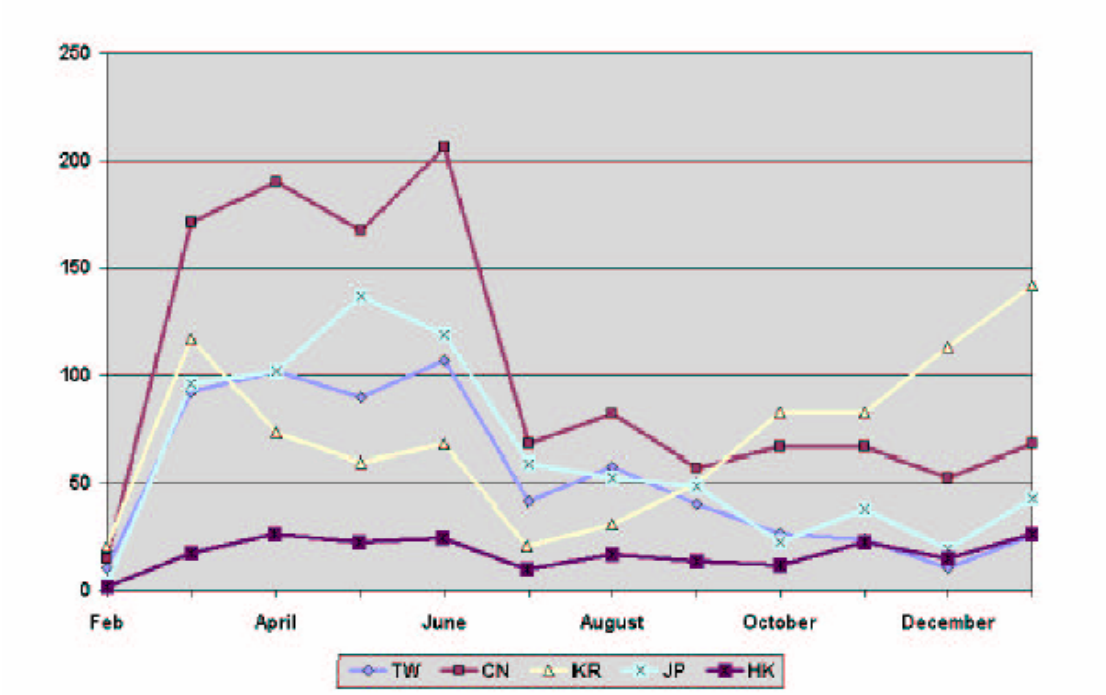

Figure 6: Addresses identified as Australian by Netgeo: Repartition by country over months

In the absence of such a mechanism, it is worth trying to imagine the reasons for such a sudden death. We have come with the following scenarii:

1. Deloder is still active but our virtual machines are not scanned anymore, for some unknown reasons. Statistically speaking, this seems unlikely and should be validated by means of the other similar platforms we are deploying.
2. All machines have been patched or the access to port 445 has been closed/blocked. Deloder has been eradicated. This is another unlikely scenario since Deloder targets a large number of platforms, many of them being personal computers which will probably never be patched. Newer successful worms targeting the same port (eg Sasser, Welchia, the Korgo family, etc.) tend to confrm this.
3. Deloder bots are listening on IRC channels for commands to run attacks. One of these commands might have told them to stop the propagation process. In this case, the Deloder worm is not visible anymore but its botnet remains as dangerous as before.

We consider the third option as the most plausible one. One of the most complete study of Deloder worm can be found in [Stone03]. V. Stone explains that the variant of Deloder

discovered on March 2003 contains an Internet Relay Chat bot Trojan, called *IRC-Pitchfork*. The bot connects compromised computers to Internet Relay Chat servers, which attackers commonly use to execute commands on the remote machines. "13 servers are tried for the IRC connection. However, there is not much public information about the variant discovered on April 2003. (...) Therefore the fate of the IRC component is indeterminate for this version." How many servers are defined in other Deloder variants? Are they all closed, and if so, who shut them down? There are clearly many questions that remain publicly unanswered. As to confirm our observation, V. Stone claims in his paper that: "At least one of these networks is an army of more than 10000 machines and there are indications that these networks are being used fornetwork administration. Whether the army is used today or held in reserve to be used at some future data for an attack does not really matter. The potential is there for them to cause serious long term damage".

This would imply that worms writers have developed a new strategy. Instead of continuously trying to compromise more machines, they have decided to enter into a silent mode when the size of their botnets is suffcient. By doing so, they dramatically reduce the likelihood of seeing an in-depth study of their worm being done as *invisible worms* are definitely less interesting to the security community than virulent ones. The bottom line of our findings is that such an in-depth analysis of that worm is probably worse being done if it has not been done yet. Sleeping worms might actually be more sophisticated and nefarious than active ones.

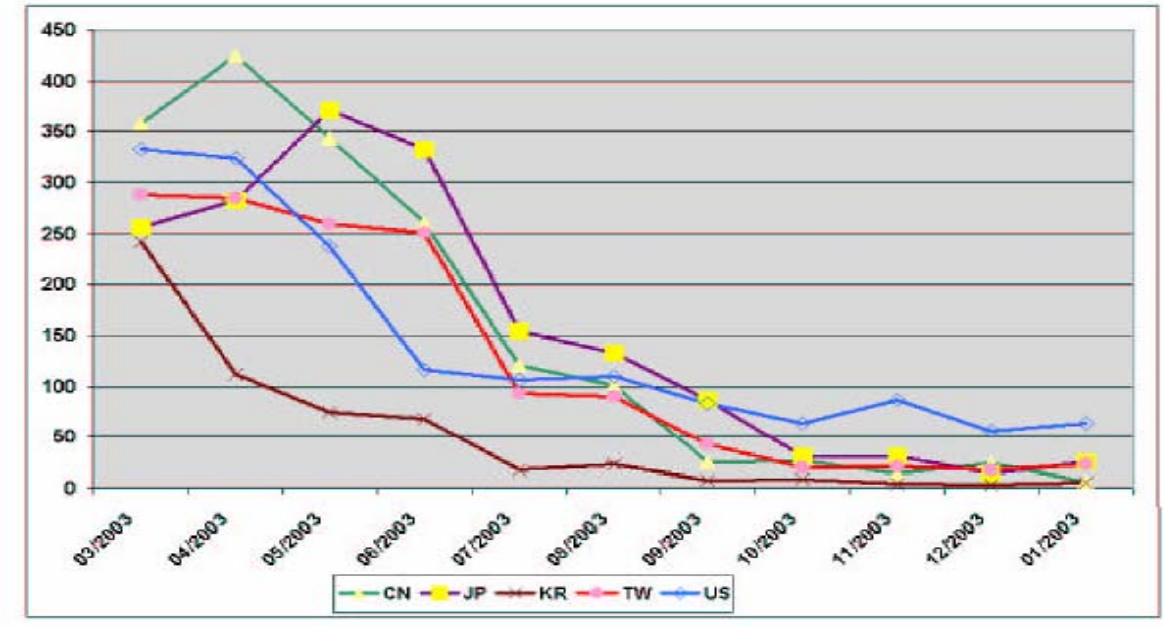

Figure 7: Deloder Activity (# asociated attack sources) observed over months

Lastly, the attentive reader will notice in figure5 that the coordinated decrease in July does not correspond to the decrease of Deloder activity for the USA (the lowest curve on figure 5). A deeper analysis reveals that another emerging cluster has become more and more important within the same period and for the USA only. This cluster, which has not clearly been identified, targets port 139[6]. As for an illustration, we show in figure 8 that the decrease of the USA in July 2003 mainly result from another cluster which has a distinct behavior. Even if this cluster is currently not identified, we show with this example that macroscopic phenomena (the apparently synchronized decrease of activities in July 2004 for the 7 countries) can hide interesting microscopic activities. These activities must be clearly analyzed in order to improve the survivalility of connected systems.

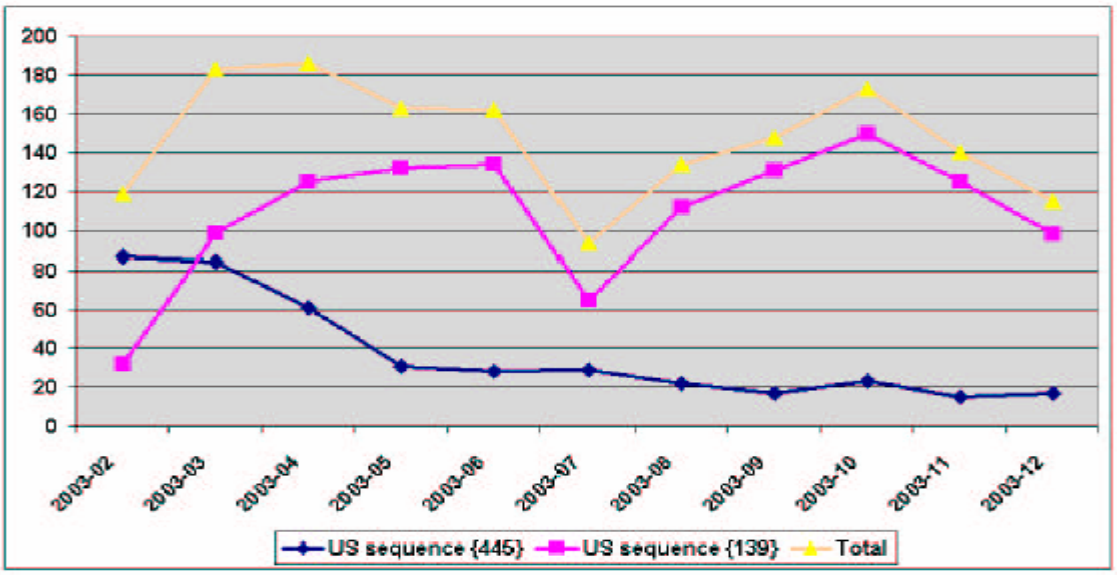

Figure 8: USA activities for ports sequences {139 } and {445 }

4.3 Scanners and Attackers

4.3.1 Preliminary Note

In [DaPD04], we have shown that approximately 70% of the observed IP addresses have sent requests to our 3 honeypots while 25% have sent packets to only one honeypot with an unexpected success rate. Apart from some identified exceptions (backscatters essentially), all IPs belonging to this category have sent packets to ports that were actually open! Statistically

[6] A similar observation has been reported in a few incidents mailing lists [Incid]

speaking, it is very unlikely to see this phenomenon. As a consequence, we have postulated that among the 70% of machines talking to our three honeypots, the role of some of them was restricted to scan our machines without trying any kind of attacks against them.

The result of this information gathering process was then later used by machines that we had never seen before, enabling them to hit systematically our machines on open ports. In order to validate that claim, to have a better understanding of the ratio of machines involved in the scanning-only process and to try to figure out how many populations of scanners we are facing, we have designed an experiment the results of which are presented here below.

4.4.2 Experiment

Starting mid October 2003, we have opened port 1433 on our linux virtual machine[7], which is the traditional port for the MS SQL service on Windows machines. Before that day, we had never observed a machine talking to that sole linux box sending packets to that port. We were interested in verifying that the situation would change once the 'hypothetical' scanners would have figured out that this port was now open.

The results we have obtained are given in Figure 9. It represents the number of sources that targetedthe sole port 1433 on that unique machine. We note that they are all new IPs. None had been observed previously. Point 1 in the Figure corresponds to the date we opened this port. Point 2 shows the first explicit attack observed on that sole machine. It reveals that it takes around 15 days for such a precise attack to happen. Also, it is worth pointing out that port 1433 has been opened on a Linux machine while this port is normally used by a Microsoft service. Thus, this indicates that scanning machines provide some basic information regarding the opened ports but fail in fingerprinting the OS of the machine they have just probed. This might be too costly in regards of the probability of having a Windows port opened on a Linux box. Moreover, the increasing number of specific attacks show that we are facing different attackers communities. Otherwise, they would not try to attack the same machine without success again and again. Clearly, these attackers did not share the experience of their failures.

At the end of December 2003, the number of attacks still increases but less abruptly. Thus, we have decided to close this port on january 12th 2004. This is indicated by the point 3 in Figure 9.

[7] We started a daemon listening on that port. There is no service attached to it

We note then a very abrupt decrease of such observed attacks. They almost totally disappear in February and there is none of them in March. This simply means that some scanners have updated the *shared information*, and it takes less than two weeks for the attackers' community to update their information and to stop the attacks.

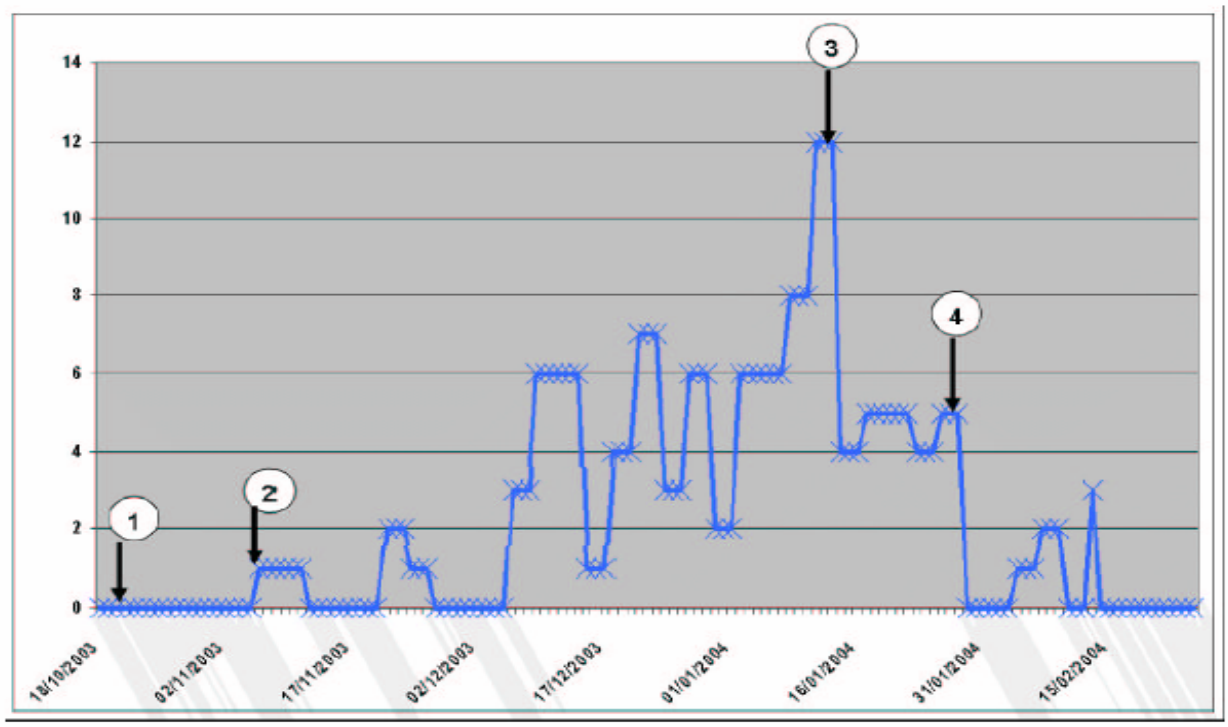

Figure 9: # attack sources having targeted port 1433 only on the sole Linux virtual machine

In summary, this experiment allows deducing four major results:

1. The claim about scanners-only machines is validated.
2. The *shared information* is not as good as it could be. Many tools are now implementing OS fingerprinting techniques (actively [Pof] or passively [Disco, Honeyd]): either scanners having targeted our machines are not using them, or the attackers communities do not check this information before launching their attacks. The second hypothesis seems to be the more correct. Indeed, we observe many scans on port 1433 following other Windows ports like 139, 445 or 135. As a consequence, such scanners know whether the machine is a Windows station or not.
3. The previous point justifies the usage of simpler honeypots than VMWare ones to replicate our environment. Indeed, if attackers do not bother distinguishing between

Windows and Linux boxes, it is quite likely that are not more interested in detecting honeypots. We had initially build our environment on a VMWare platform to avoid introducing any bias in the data collection process. Now armed with these results, we have good reasons to shift to a cheaper and simpler environment, despite its limitations with respect to fingerprinting. We will, though, maintain a few VMWare machines to verify that observations obtained in both environments keep being consistent.

4. Once we open the port, it takes 15 days for the first attack to be observed. However, the elapsed time is a lot shorter for the attacks to get updates on the port state: As we close the port, attacks dramatically stop within less than two days. This means such attacks share the same information and that attackers are very reactive to attack changes. Furthermore, we notive another important decrease of attacks at point 4 and then a few attempts in February. This leads to the following remark: We are facing a very few numbers of attackers, or at least communities of attackers. We distinguish from figure 9 three of them: those who immediately stopped after point 3, those who stopped after a few days at point 4, and the minority who still keep on performing unsuccessful attacks on February.

Having validated the four previous points, we can analyze a little bit further this phenomenon by using our clustering algorithm (see Section 3) in order to determine potential scanners that have made this *shared information* available. We can reasonably assume that the information about open ports is maintained up to date by a single tool. In other words, the same scanning tool is responsible for having identified the ports as open and, later for having changed the information regarding that port when it found it closed again. With this assumption in mind, we observe the following facts:

1. From the date we opened the port (point 1 in Figure 9, October 20, 2003) to the date we observed the first explicit attack (point 2 in Figure 9 November 6th, 2003), the clustering algorithm shows that five different sets of machines have targeted port 1433 among others on our 3 honeypots.
2. We observe only two of these five clusters between the 'closing date' of the port (point 3) and the decrease of the attacks (point 4).
3. The scanning tool which shares information with attack comunities is quite likely one of these two clusters (if not both). Figure 10 gives the observation of these two clusters from

October 2003 to March 2004. Scans grouped in the first cluster (Cluster 1) are observed frequently and regularly. Those in the second cluster (Cluster 2) appear, at the contrary, rather sporadically.

4. Instances of Cluster 1 are seen every day. Thus, if Cluster 1 is the one that contains the scans that have led to the following direct attacks, it is diffcult to understand why it took two weeks to see the first attack launched. On the other hand, the first attack is observed less than 5 days after the first scan belonging to Cluster 2. Moreover, the port has been closed on January 12th (point 3) and the first Cluster 2 scan has been observed a dozen days later. The decrease in the number of attacks was noticeable a few days after that specific scan, at the end of January, as can be seen on Figure 9.

Cluster 2 seems thus to be a good candidate. This is confirmed by looking at packets corresponding to both clusters. Some packets associated to Cluster 1 contain a 42-byte data payload sent to our honeypot. In other words, these machines not only look for open ports but also try to send some data to them. On the other hand, packets associated to Cluster 2 are simple TCP SYN, half open scans. Similar experiments on different platforms would enable us to determine more precisely the modus operandi of these scan-only machines but, so far, this refined analysis leads to the fifth and final result of this experiment:

5. Information gathered by a given scan is shared between several communities of attackers. Indeed, we do not see a one-to-one relationship between a scan-only IP and an attacker. On the contrary, we have more attacks than scans. 76 different but precise attacks have been performed from October 20th, 2003 to January 12th, 2004 (points 1 and 3 on the Figures respectively). Scans belonging to cluster 2 have only been observed five times for the same period. This tends to indicate that more than one community is using the information provided by a scan-only machine. Here to, we need more machines to get a better insight on the interactions between scanner-only and attackers. This is part of our ongoing work.

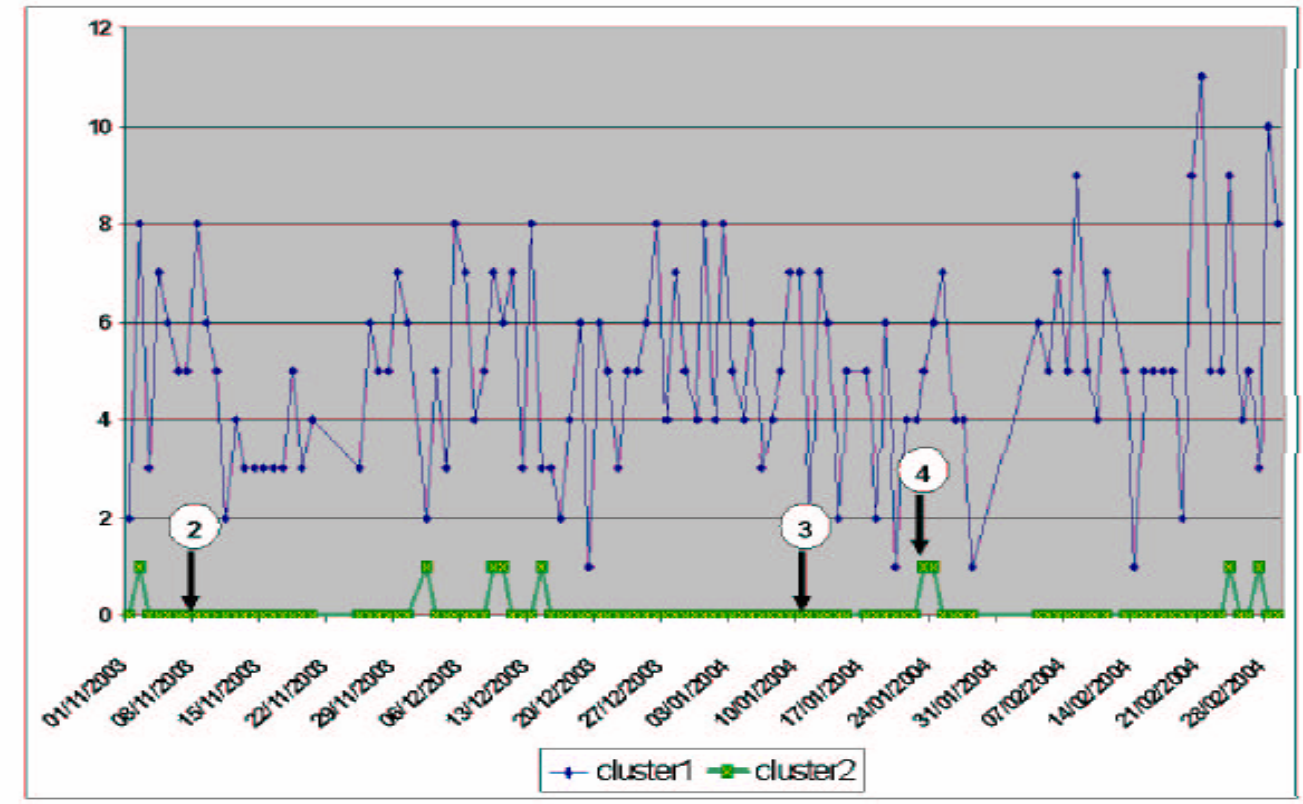

Figure 10: Clusters activity (#attack sources) per day

5. Conclusion

Honeypots are very interesting observation environments that enable us to easily collect malicious data. However, very few efforts are currently made to take advantage of these rich information sources. Much of the effort is devoted to honeypots design.

We show in this report that simple clustering techniques can be applied to obtain more in-depth information on observed attacks. We have proposed one solution based on association rules and phrases distance. Results are very promising and confirm that such an analysis is meaningful. Indeed, we obtain clusters representing *root causes* of attacks.

We have shown with this approach that a large amount of malicious traffic corresponds to a few number of *root causes*. Such *root causes* are important information on the *attack tools* in use.

In addition, we have managed in this report to explain some weird phenomena, such as the Australian activity which was dominant at a first glance. We have also validated that machines

are facing multiple independent communities of attackers, and that it is possible to estimate their number, as well as the kind of information they might exchange together. Finally, we have justified the usage of honeypots as an interesting platform to collect data in oder to build analytical models of Internet threats. Some other questions arise and require more expertise. They will be detailed along with the other deliverables.

6. Bibliography

[ActTarg] ActiveTarget. Component to lookup country name by ip address. demo URL: http://www.activetarget.com/livedemo.asp.

[Agra93] R. Agrawal, T. Imielinski, A. Swami. "Mining Association Rules between Sets of Items in Large Databases". *In Proc. of the 1993 ACM SIGMOD International Conference on Management of Dat*a.

[Agra94] R. Agrawal, R. Srikant. "Fast Algorithms for Mining Association Rules". *IBM Almaden Research Cente*r, 1994.

[Bell92] S. Bellovin, "There Be Dragons", *Proc. of the Third Usenix Security Symposiu*m, Baltimore MD. Sept. 1992.

[Bell93] S. M. Bellovin, "Packets Found on an Internet", *Computer Communications Review* 23:3, pp. 26-31, July 1993.

[Borg03] C. Borgelt, "Finding Association Rules/Hyperedges with the Apriori Algorithm". *Dpt of Knowledge Processing and Language Engineering,* Germany, 2003.

[Caida] CAIDA Project. Netgeo utility - the internet geographical database. URL:http://www.caida.org/tools/utilities/netgeo/.

[Cohe99] *Deception Tool Ki*t, DTK, Fred Cohen & Associates. http://www.all.net/dtk/dtk.html

[CLPB01] F. Cohen, D. Lambert, C. Preston, N. Berry, C. Stewart and E. Thomas, *"A Framework forDeceptio*n", Tech. Report, July 2001, http://all.net/journal/deception/Framework/Framework.html

[DaPD04] M. Dacier, F. Pouget, H. Debar, "Attack Processes found on the Internet". *NATO Symposium IST-041/RSY-01*3, Toulouse, France, April 2004.

[Delo1] McAFee Security Antivirus. Virus pro_le: W32/deloder worm. URL:http://us.mcafee.com/virusInfo/.

[Delo2] F-Secure Corporation. Deloder worm analysis. URL:http://www.f-secure.com.

[DPDe04] M. Dacier, F. Pouget, H. Debar, "Honeypots, a Practical Mean to Validate Malicious FaultAssumptions". *Proc. Of the 10 th Pacific Ream Dependable Computing Conference (PRDC04*), February 2004.

[Disco] *The Disco tool* home page: http://www.altmode.com/disco/

[GenH03] *"Know Your Enemy: GenII Honeynets Easier to deploy, harder to detect, safer to maintai*n", by the honeynet Project members, June 2003. Available on line: http://project.honeynet.org/papers/gen2/

[Grah03] R. Graham, *"MS-RPC/DCOM/Blaster case study*", ISS presentation available on line at: http://csiannual.com/classes/v12.pdf

[GrimP] Grim's Ping home page: http://grimsping.cjb.net/

[Han01] J. Han, M. Kamber, *"Data Mining: Concepts and Techniques*", Morgan Kaufman Publishers, 2001.

[HoIe03] Irish Honeynet Alliance members. Collected data available on line: www.honeynet.ie/results.htm

[HonAll] Honeynet Alliance home page: http://www.honeynet.org.

[Honeyd] Honeyd Virtual Honeypot from N. Provos. URL:http://www.honeyd.org.

[HonNo] Norvegian Honeynet Alliance members: Collected data available on line: www.honeynet.no/reports/

[Incid] Insecure.Org Mailing List. Thread: port 139 syn-_n scans, 2003. URL:http://seclists.org/lists/incidents/2003/Apr/0098.html.

[IP2loc] IP2Location. Bringing geography to the internet. demo URL: http://www.ip2location.com/free.asp.
[IPtables] IPTables Net_lter Project. URL:http://www.net_lter.org.
[ISC03] *Internet Storm Cente*r, home page: http://isc.incidents.org/
[Juli03] K. Julisch, "Using Root Cause Analysis to Handle Intrusion Detection Alarms". *PhD dissertatio*n, IBM Research laboratory of Zurich, Switzerland, 2003.
[Kraw04] N. Krawtez, "Anti-Honeypot Technology", *IEEE Security and Privacy.* Vol 2, Nb 1, p. 76-78, 2004.
[Ksm01] K. Smani lectures available at: http://www.csee.wvu.edu/~ksmani/courses/fa01/random/lecnotes/lecture5.pdf
[Labrea] Labrea Tarpit Project, http://labrea.sourceforge.net/
[Lati02] R.J. Latino, K. latino, "Root Cause Analysis: Improving Performance for Bottom Line Results". *CRC Press, LLC,* 2002.
[Lev04] Leviathan Auditor open source project, home page: http://leviathan.sourceforge.net/
[Liv01] A.D. Livingston, G. Jackson, K. Priestley, "Root Causes Analysis: Literature Review". *HSE Book*s, 2001. Available at: http://www.hsebooks.co.uk
[Mathw] The Stable Marriage Problem: http://mathworld.wolfram.com/StableMarriageProblem.html
[Maxmind] MaxMind GeoIP Country Database Commercial Product. URL:http://www.maxmind.com/app/products.
[Moo02] D. Moore, C. Shannon, G.M. Voelker, and S. Savage. Core-red a case study on the spread andvictims of an internet worm. In *ACM/USENIX Internet Measurement Workshop*, November 2002.

[MoVS01] D. Moore, G. Voelker et S. Savage. *"Inferring Internet Denial-of-Service Activit*y", 2001 USENIX Sec. Symp. www.caida.org/outreach/papers/2001/BackScatter/usenixsecurity01.pdf
[Netgeo] *Netgeo Utilit*y, available online at http://netgeo.caida.org/perl/netgeo.cgi
[Nevi03] A. Neville, *"IDS Logs in Forensics Investigations: An Analysis of a Compromised Honeypo*t", March 2003. Available on line: http://www.securityfocus.com/infocus/1676
[Ng02] T.S. Eugene Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM 2002*, 2002. URL:http://www-2.cs.cmu.edu/ eugeneng/papers/INFOCOM02.pdf.
[Nmap] NMap utility from insecure.org. URL:http://www.insecure.org/nmap.
[Para88] M. Paradies, D. Busch. "Root Cause Analysis at Savannah River Plant". *In Proc. of the IEEE Conference on Human Factors and Power Plants,* page 479-483, 1988.
[PDD03] F. Pouget, M. Dacier, H. Debar: "Honeypot: a comparative survey". *Eurecom Report, RR-03-8*1, September 2003.
[P0f] p0f passive fingerprinting tool home page: http://lcamtuf.coredump.cx/p0f-beta.tgz
[Pow03] "Conceptual Model and Architecture of Maftia", D. Powell, R. Stroud (editors), MAFTIA Project (IST-1999-11583), Deliverable D21, January 2003; available on line at http://www.maftia.org
[Road04] Roadkil's FTP Probe home page: http://homepages.ihug.com.au/~roadkil/ftpprobe.htm
[Ros03] A. Rosin. Measuring availability in peer-to-peer networks. September 2003. URL:http://www.wiwi.hu-berlin.de/ _s/p2pe/paper A Rosin.pdf.
[Rtsp] RTSP Scanner available at: http://iperl.homelinux.org/haxor/scanner.pl
[Secu04] Honeypots mailing list archives from SecurityFocus: http://www.securityfocus.com/popups/forums/honeypots/faq.shtml
[Sfind] Sfind scanner home page: http://www.force5web.com/articles/sql_scan.htm
[Snake] SQLSnake presentation available at: www.giac.org/practical/Christopher_Short_GCIH.doc
[SoMS01] D. Song, R. Malan and R. Stone, *"a global snapshot of internet worm activit*y", November 2001. Technical Report, http://research.arbor.net/downloads/snapshot_worm_activity.pdf.
[Stol88] C. Stoll, "Stalking the Wiley Hacker", *Communications of the AC*M, Vol. 31 No 5. May 1988.

[Stone03] V. Stone. W32 deloder worm: The building of an army. Technical report, 2003. URL:http://www.sans.org.

[Spit02] L. Spitzner, *"Honeypots: Tracking Hacker*s", Addislon-Wesley, ISBN from-321-10895-7, 2002.

[Sym03] Symantec MBlaster advisories available at: http://securityresponse.symantec.com/avcenter/venc/data/

[Sym04] Symantec. Symantec security response w32.welchia.worm, 2004. URL: http://response.symantec.com/avcentr/venc/data/w32.welchia.b.worm.html.

[TCPdump] TCPDump utility. URL:http://www.tcpdump.org.

[Tethe04] tethereal utility home page: http://www.ethereal.com/tethereal/

[THP03] *Tiny Honeypot* home page: http://www.alpinista.org/thp/

[VMware] *VMWAR*E, User's manual. Version 3.1, home page: http://www.vmware.com

[Yoo02] S.H. Yook, H. Jeong, and A.L. Barabasi. Modeling the internet's large scale topology. In PNAS -vol.99,October 2002. URL:http://www.nd.edu/ networks/PDF/Modeling.

[Zeit03] A. Zeitoun, C.N. Chuah, S. Bhattacharyya, and C. Diot. An as-level study of internet path delay characteristics. Technical report, 2003. URL:http://ipmon.sprint.com/pubs trs/trs/RR03-ATL-051699-AS-delay.pdf.