available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

A framework for attack patterns' discovery in honeynet data

Olivier Thonnard[a,*], *Marc Dacier*[b]

[a]*Royal Military Academy, Polytechnic Faculty, Brussels, Belgium*
[b]*Institut Eurecom, 2229 Route des Cretes, Sophia Antipolis, France*

Keywords:
Honeypot forensics
Traffic analysis
Attack patterns
Security data mining
Knowledge discovery

ABSTRACT

Collecting data related to Internet threats has now become a relatively common task for security researchers and network operators. However, the huge amount of raw data can rapidly overwhelm people in charge of analyzing such data sets. Systematic analysis procedures are thus needed to extract useful information from large traffic data sets in order to assist the analyst's investigations. This work describes an analysis framework specifically developed to gain insights into honeynet data. Our forensics procedure aims at finding, within an attack data set, groups of network traces sharing various kinds of similar patterns. In our exploratory data analysis, we seek to design a flexible clustering tool that can be applied in a systematic way on different feature vectors characterizing the attacks. In this paper, we illustrate the application of our method by analyzing one specific aspect of the honeynet data, i.e. the time series of the attacks. We show that clustering attack patterns with an appropriate similarity measure provides very good candidates for further in-depth investigation, which can help us to discover the plausible root causes of the underlying phenomena. The results of our clustering on time series analysis enable us to identify the activities of several worms and botnets in the collected traffic.

1. Introduction

1.1. *Collecting malicious traffic*

The experimental study of attacks on the Internet is a very active research domain, and it has gained a lot of attention in recent years. Many valuable initiatives exist for capturing or monitoring malicious activities. Broadly speaking there are three approaches used to monitor unsolicited traffic. The first approach uses low- or high-interaction honeypots (Leurre.com project; Riordan et al., 2006; Provos, 2004; Baecher et al., 2006; Werner); in the second approach the so-called Internet telescopes, or darknets (Moore et al., 2004; Internet Motion Sensor; Team Cymru), are used in order to monitor all traffic directed to unused subnets within a locally allocated address space. The last class of techniques for gathering information about computer attacks aims at collecting and sharing firewall and IDS logs collected from a very large number of heterogeneous sources (DShield).

Collecting data related to Internet threats has thus become a relatively common task for security researchers or network operators. However, effectively analyzing the vast amounts of data collected using these above-mentioned sensors can prove challenging. The volume and the diversity of the raw data can rapidly overwhelm people in charge of analyzing those data sets. As a result, deciding which samples or which attack patterns should be investigated first is a difficult task.

1.2. *Towards a systematic analysis framework*

This research develops a systematic analysis framework for exploring the malicious Internet traffic obtained from a distributed

* *Corresponding author.*
E-mail addresses: olivier.thonnard@rma.ac.be (O. Thonnard), marc.dacier@eurecom.fr (M. Dacier).
1742-2876/$ – see front matter © 2008 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2008.05.012

set of honeypot responders. The framework relies on a quality-based clustering technique specifically designed to identify all groups of highly similar attack patterns in a "one-pass procedure". Although other dimensions are discussed here below, the primary clustering feature is "time signature". The contribution is being able to discover attack patterns via attack trace similarity, rather than via a rigid signature. More specifically, the contributions are: (i) an ability to detect attacks as a function of a series of (potentially stealthier) connections, rather than detecting a single nefarious connection; (ii) detecting zero-day or polymorphic attacks based on similarity to other attacks; and (iii) a systematic approach for obtaining knowledge from honeynet data that can be leveraged in intrusion detection efforts for real-world systems or networks.

1.3. *Identifying relevant attack features for clustering*

Experience has shown that certain *attack features* may provide valuable information about the root causes of attack phenomena. First, the geographical location of the attackers may be useful for identifying attacks coming only from specific countries, hence revealing something about the spatial aspects of the attack root causes. The IP subnet characteristics of the attackers are also an interesting feature for data exploration. Attackers' subnets can provide a good indication of the spatial "uncleanliness" of certain networks, i.e. the tendency for compromised hosts to stay clustered within unclean networks, especially for zombie machines belonging to botnets, as demonstrated in Collins et al. (2007). This information can thus help us to determine appropriate counter-measures to defend ourselves against this kind of attack phenomena. Other studies have demonstrated that certain worms show a clear bias in their propagation scheme, such as a tendency for scanning machines of the same (or nearby) network so as to optimize their propagation (Chen et al., 2003). So, if we analyze the collected attacks by looking at the distance between the attackers' IP addresses and the ones of the victims (i.e. our sensors), then experience has shown that we can recognize the propagation mode of certain malware families.

Also, time series analysis can provide useful information about the underlying attack phenomena and might thus also be used as clustering feature. In this specific case, the choice of a good time granularity will depend on the kind of attack phenomena we want to investigate: for short high-intensity attacks, like botnet probes or flash worms, it might be useful to represent attacks with smaller time intervals (e.g. 1 h), while other types of worms will use a stealthier propagation scheme that will only show up on a larger time scale (e.g. by day).

Yet other features can prove relevant *at one given time* for learning about the attack processes and the modus operandi of attackers, such as the name resolution of the attackers (TLDs, ISPs or hostnames), or the malware characteristics when available (executable name, MD5 hash, name given by AV-vendors).

As detailed above, performing network-based forensics on real-world attack traces can be challenging, as attack features may vary over time or are not always relevant for all phenomena. In other words, attack phenomena have not only a very large dimensional space, but also a highly dynamic and changing behavior. This has motivated us to introduce some flexibility in our analysis framework, in order to allow the analyst to plug in different feature vectors and appropriate similarity metrics to be used in the clustering step. This flexibility helps the analyst to effectively investigate the collected malicious traffic by taking different views on the data set. In this work, we illustrate the application of our framework by exploiting one specific aspect of the attacks, namely the time series of the attacks, as explained here after.

1.4. *Exploring attack time signatures*

We use here the *time signatures* of a large set of attacks as input features of our framework, with the objective of discovering similarities among them. By "time signature", we mean a time series representing the aggregated source count for a given type of attack. Fig. 1 gives a concrete example of two different attacks observed on two different sensors. We represent on the x-axis, the time (by day), and on the y-axis, the number of IP sources belonging to a given attack. We consider that two sources realized the same attack when they have created a very similar network trace on a honeypot with respect to the following characteristics: the ordered sequence of ports that have been targeted on the honeypot, the number of packets and bytes exchanged in the complete set of connections, the duration of the whole attack and the packets' payload. Note that this preliminary classification of the traffic is explained in more detail in Section 3.1. Each type of attack is then identified by a unique number. We note that the two attacks represented in Fig. 1 present strong temporal similarities on both sensors, even though they target completely different ports (5900/TCP and 1433/TCP).

So, there are several reasons for interest in discovering similarity patterns between the temporal evolutions of the attacks. First, very similar temporal patterns can bring some evidence of synchronized activities on different sensors, revealing a certain spatial diversity for similar attack activities, which originate most probably from the same root phenomenon. Then, the type of pattern (i.e. the "shape" of the time series) may also reveal valuable information about the observed activity. For instance, a recurrent diurnal/nocturnal activity pattern may indicate a botnet propagation scheme, as suggested by the diurnal propagation model of Dagon et al. (2006). The authors observed dozens of botnets representing millions of victims. They noted diurnal properties in botnet activity, which might, for example, occur because victims are clustered in different time zones and turn their computers off at night. Finally, when we find very similar patterns shared by completely different attacks, say for instance probes on completely different TCP ports, this may also help to discover so-called "multi-headed" attack tools as identified in Pouget et al. (2006b), which combine different types of attack techniques (i.e. different exploits) to propagate.

The remainder of the paper is organized as follows. Section 2 presents a general graph-based clustering method we designed to discover similarity patterns from a large data set, which we have implemented in our framework. In Section 3 we illustrate step-by-step the application of our method to analyze time series. Then we present the results of the experiments we

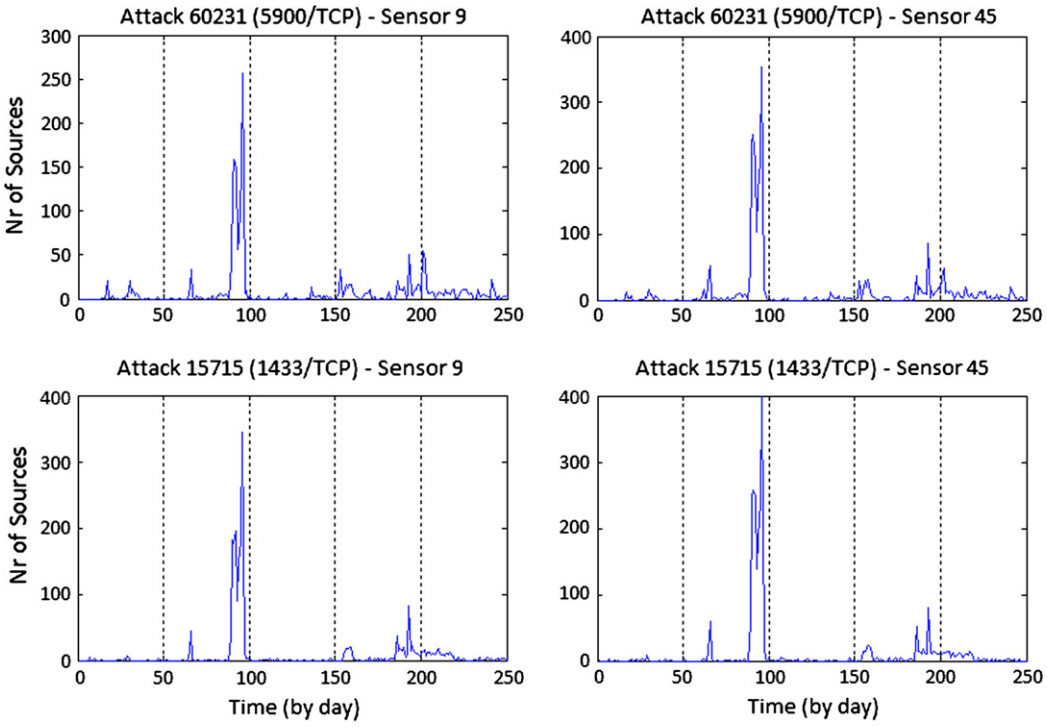

Fig. 1 – Two different attacks (identified by no. 60231 and no. 15715) observed on two different sensors (no. 9 and no. 45). While targeting completely different ports (5900/TCP and 1433/TCP), these attacks present strong temporal similarities.

performed on a large set of real-world attack traces collected with 44 worldwide distributed honeypots. We illustrate the efficacy of our method, and we underline its good potential in assisting investigations about attack phenomena. Finally, we conclude with a discussion and we outline some future work.

2. Graph-based clustering

2.1. *Method overview*

The clustering method of our analysis framework involves the following steps, which are commonly used in many typical data mining tasks (Jain and Dubes, 1988):

1. feature selection and/or extraction, and pattern representation;
2. definition of a pattern proximity measure appropriate to the data domain; and
3. grouping similar patterns.

In the very first step of any clustering task, we select or extract a certain feature characterizing a relevant aspect of the data. Then we represent the aggregate data series with an adequate means, usually with an array of values (for instance, a times series for representing the temporal evolution of an attribute). Sometimes, the term ''feature vectors'' is used to denote such data series.

Once created, we need an effective means to measure how similar two data series are with respect to a certain distance. For that purpose, we can find many valuable contributions which thoroughly study various types of similarity distances (e.g. Mahalanobis, Minkowski, Pearson or Spearman correlations, jackknife correlation, etc.). Obviously, the similarity metric must be carefully determined in consideration of the original data series and the expected properties of the clusters (e.g. cluster size, quality, consistency, etc.), but this discussion lies outside the scope of this work. In Section 3.2, we will briefly present one efficient similarity measure called SAX, which we adapted specifically to our honeypot time series to obtain the desired properties when measuring the similarities.

2.2. *Grouping patterns – a graph-based approach*

The last step consists of grouping all patterns that look very similar. There exists a plethora of clustering algorithms for doing this. We outline and compare some well-known techniques in the last subsection. In the design of our method, we used a graph-based approach to formulate the problem, which is for us a convenient representation in this case.

A graph is a structure used to model pairwise relations between objects from the same domain. It comprises a set of vertices (or nodes) connected by links called edges, which can be directed or undirected. We can define $G = (V, E, w)$ as an undirected edge-weighted graph with no self-loops where $V = \{1, 2, \ldots, N\}$ is the vertex set, $E \subseteq V \times V$ is the edge set and represents the relationships between each pair of vertices, and $w : E \rightarrow \mathcal{R}^+$ is a positive weight function associated with each edge of E.

We can represent the graph G with its corresponding weighted *adjacency matrix* (aka the similarity matrix), which is the $n \times n$ symmetric matrix $A(i, j) = w(i, j), \forall (i, j) \in E$; and 0 otherwise. The weight of each relationship $w(i, j)$ is computed with the similarity metric defined at step 2. A *clique* is defined as an induced subgraph of a (un)directed graph in which the vertices are fully connected. A clique is maximal if it is not contained within any other clique.

Hence, finding the largest group of similar elements in a data set can be transformed into the problem of searching for complete subgraphs where the links express the similarity relationships between the vertices. This is a classical NP-complete problem studied in graph-theory, also known as the *maximal clique problem* (MCP) (Bomze et al., 1999). The MCP is a combinatorial optimization problem that involves finding the largest subset of fully connected nodes in a graph. It has many important practical applications, especially in information retrieval, classification theory and cluster analysis, economics, scheduling, VLSI design and computer vision (Pardalos and Xue, 1994). In fact, the maximal complete subgraph was considered the strictest definition of a cluster in Gary Augustson and Minker (1970). Because of its NP-hard complexity, many approximate algorithms for solving the MCP have been developed, like the dominant sets' approach (based on replicator equations; Pavan and Pelillo, 2003), local search heuristics, Hopfield network, ant colony optimization, and the heuristic-based genetic algorithm, among others (Xinshun et al., 2007). Those methods are often elaborate and have provable bounds on how well they can approximate the MCP.

In the early stages of our framework design, we used the dominant sets' approach of Pavan and Pelillo (2003), which proved to be an effective method for finding maximal cliques. Although the results were satisfactory in terms of solution quality, we were still confronted with the problem of a very high computational cost, mostly due to the fact that MCP algorithms require the calculation of the complete adjacency matrix (i.e. $N(N-1)/2$ elements of the symmetric matrix $A(i, j)$). This is too computationally expensive for our application due to two constraints: (i) the size of the data set; and (ii) the function used to compute the pairwise similarities between complex time series like the ones depicted in Figs. 1 and 2.

2.3. *Fast quality-based clustering*

2.3.1. *Design choices*

For the reasons outlined above, we needed to define a cheaper clustering algorithm to find cliques in our data set. The desired properties of the algorithm are (i) to identify fully connected components with a ''one-pass run''; (ii) to focus on finding cliques that have a (high) quality guarantee; and (iii) to reduce the total processing time by eliminating the link calculations that are not really necessary to obtain the expected results. The last two properties can be achieved by defining a good similarity function (i.e. characterized by a low false positive rate) and by setting a quality threshold on it. This must ensure a transitivity between any triplet of vertices in a clique. That is, by setting a high quality threshold, any two vertices of a clique must have a similarity value that is close to that threshold

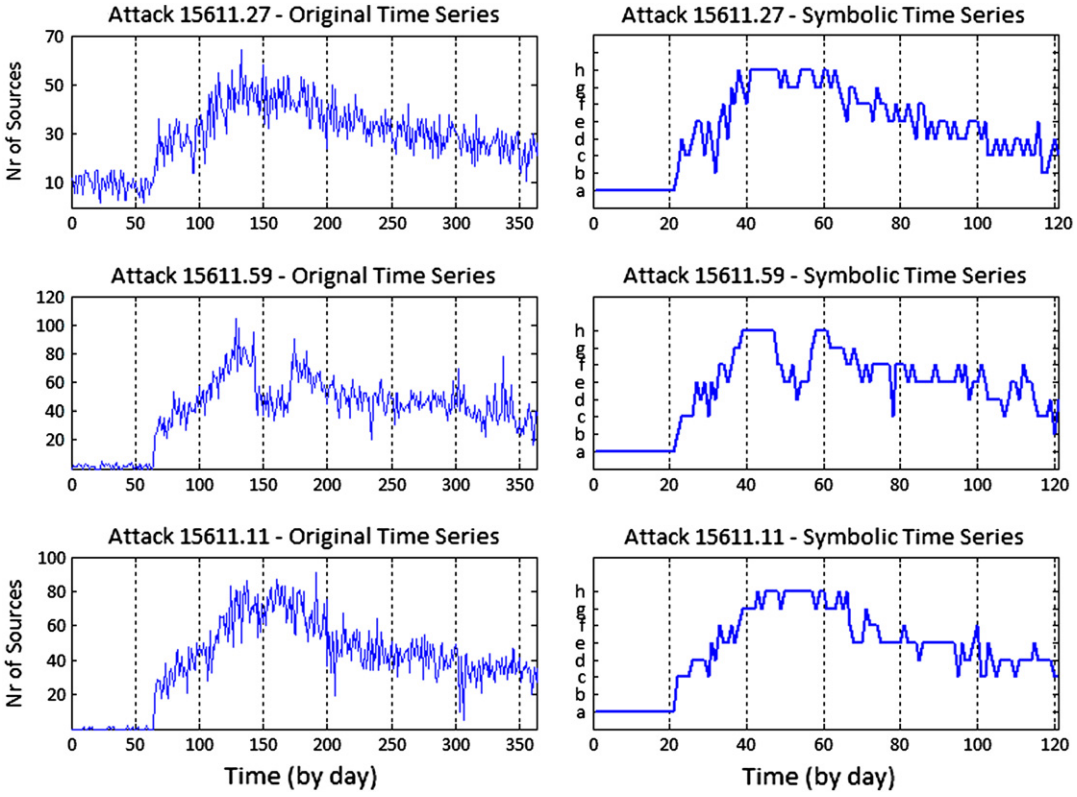

Fig. 2 – (Left) Three time series of attack 15611 as observed on three different sensors. (Right) Their respective symbolic representations obtained with SAX, which are far easier to compare. This attack is related to ICMP traffic generated by the W32/Allaple.B worm.

or higher. In other words, the algorithm focuses on the consistency of the solutions by setting a certain "diameter" on the clusters, so that only very close neighbors are clustered together to form a clique. Since all members of our clusters will have by design a strong relationship with every other node, we keep the *clique* denomination in analogy with the concept used in graph-theory (even if the cliques we find may not be maximal). We are aware that this algorithm no longer addresses the initial MCP introduced in the general formulation of the problem. These design choices are motivated by a trade-off between the expected properties of the results and the processing time. We want to quickly discover the general attack patterns in a large data set. Once the patterns are revealed, we may analyze specific patterns of interest in more detail.

2.3.2. *Quality-based clustering algorithm*

The algorithm works as follows: a candidate clique is formed by starting with the first pattern. The algorithm computes the links between the considered pattern and all the remaining ones. All the patterns that have a similarity of at least q with this pattern are merged into the current clique. If the clique size is at least 2 (or another threshold value defined by the user), then the algorithm extracts the clique members from the data set and saves them for later visual inspection. The procedure continues with the next available pattern in the reduced data set. The pseudo-code for the algorithm is given below. The procedure takes as input parameters a set of patterns V and a quality threshold q.

```
Algorithm 2.1 : FindCliques(V,q)
pattern ← V[0]
V ← V\V[0]
sim[0] ← 1
i ← 0
for each p ∈ V
  do { i ← i + 1
       sim[i] ← ComputeSim(pattern, p)
clique ← {j || sim[j] < q  ∀j ∈ (1, …, |sim|)}
S ← S ∪ clique
S ← FindCliques(V\S,q)
return (S )
```

2.3.3. *Strengths and limitations*

The quality-based algorithm has several advantages over other clustering techniques: firstly, neither the total number of cliques nor the clique size are needed prior to running the algorithm; at the end, all cliques will have the expected quality level; and finally, this straightforward algorithm has a low computational complexity and thus a fast running time.

On the other hand, a first limitation of the algorithm is the requirement for the definition of a quality threshold to decide if two patterns must be grouped or not. Although the threshold value may change the number and size of the final cliques, this is not a serious problem in our case. Each clique will have a guaranteed quality as specified by the user, and we note that the precise threshold value is not so important. At this point of our analysis, we want to discover the general patterns present in the cliques, so they can guide further investigations. We may accept that our cliques are not maximal ones.

A second limitation is that the algorithm output can be influenced by the order in which the cliques are formed. This could be avoided by forming, at each step, candidate cliques for every remaining pattern and by selecting the largest one. Doing so would minimize the impact of local decisions on the final clique results, but again this would induce a higher computational cost.

Another improvement (which we leave as future work) consists of adding a second pass to the algorithm in order to assess the quality of the results obtained with a preliminary quality threshold, and to derive an optimal diameter for each clique (adaptive approach; De Smet et al., 2002). This would eliminate clique members that are not significantly similar to the others, and it would also free the user from finding an optimal value for the quality threshold.

2.3.4. *Comparison with classical clustering techniques*

The quality-based clustering has some resemblance to an agglomerative hierarchical clustering technique, more specifically the complete linkage procedure, which tends to find small, compact clusters with all patterns in a cluster within a given diameter threshold. The threshold value is determined by the level at which the hierarchical tree is cut. For large data sets, the tree can become extremely complex, so the choice of where to cut the tree can be difficult to define. Also, the complete linkage algorithm decides to join two elements based only on the distance between those elements (local decision-making scheme), which can induce mistakes in the clustering. Two other classical methods include k-means clustering and self-organizing maps. Although both algorithms have some interesting features, one of their main problems is that the number of clusters (k) must be specified in advance, and thus the final clustering depends heavily on the choice of k. In our application, k is unknown prior to the execution of the clustering. Furthermore, clusters formed by those methods do not guarantee a certain quality level.

3. Application to honeypot time series

3.1. *Experimental environment*

For the sake of completeness, we briefly introduce the experimental environment used to collect honeypot data. The reader familiar with previous publications in the context of the Leurre.com Project may jump directly to Section 3.2 where we describe our experiments.

3.1.1. *Leurre.com*

Our analysis draws upon extensive honeynet data obtained through the *Leurre.com Project* (Leurre.com project). The main objective of this global distributed honeynet is to get a more realistic picture of certain classes of threats occurring on the Internet, by using unbiased quantitative data over a long-term perspective. Since 2003, a distributed set of identical honeypots based on honeyd (Provos, 2004) have been deployed in many different countries, and on various academic and

industrial IP subnets. Each platform runs three virtual honeypots that emulate different operating systems (two Windows and one Linux machine) with various common services (e.g. the ports 21, 23, 80, 137, 139, and 445 are open on Windows honeypots, and ports 21, 22, 25, 80, 111, 514, 515 and 8080 are open on Linux responders). Each of the three virtual hosts has its own public IP address, so each platform maintains three public IP addresses. The collected traffic, including the payloads of the packets, is automatically stored in an Oracle database. The traces are also enriched with contextual information, such as (i) the geographical localization of the attackers (obtained with MaxMind); (ii) the passive OS fingerprinting (obtained with P0f); and (iii) the DNS reverse lookups for domain names and hostnames.

3.1.2. *Attack fingerprints*

Previous studies within this project focused on techniques for identifying the attack tools behind the observed network traffic by using clustering algorithms, and on the use of packet inter-arrival times (IATs) to characterize anomalous traffic, malicious or otherwise (Pouget et al., 2006a). A clustering engine (presented in Pouget and Dacier, 2004) classifies all IP sources based upon their network behavior. An *attack source* is defined as an IP address that targets a honeypot platform on a given day, with a certain port sequence. By *port sequence* we mean the ordered list of ports targeted by the source on a honeypot, so it can involve multiple protocols. For instance, if a source tries to connect to port 80 (HTTP), and then to ports 8080 (HTTP Alternate) and 1080 (SOCKS), the associated port sequence of the attack session[1] will be |80T|8080T|1080T. The clustering technique aims to find a good trade-off between specificity and generalization of the observed activities, by generating *attack fingerprints* based on the following network characteristics: (i) the number of virtual machines targeted on a platform, (ii) the number of packets sent to each virtual machine, (iii) the total number of packets sent to the platform, (iv) the duration of the attack session, (v) the average inter-arrival time between packets, and (vi) the associated port sequence. A last refinement step is the payload validation, based on the Levenshtein distance, for the attack sessions that contain data. If the algorithm identifies within a cluster multiple groups of attack sessions sharing similar payloads, it further refines the cluster according to these groups.

In the rest of the text, we will simply use the term *attack* to refer to the set of IP sources having the same attack fingerprint on a honeypot sensor.

3.2. *Experiments*

We analyzed the attacks collected by 44 honeypot sensors that are located in various geographical places and belong to different IP subnets. These sensors have been selected because they have been up and running during the whole analysis period, which spans 486 days starting from September 1st, 2006, until January 1st, 2008. Some characteristics of our data set[2] are given in Table 1.

Table 1 – Data set characteristics

Total number of sources	Data set size	Protocol distribution of the sources
1,738,565	~27 GB	TCP only: 26.5% UDP only: 32.3% ICMP only: 35.8% ICMP + TCP: 5.4%

3.2.1. *Data aggregation*

For each attack on a given sensor, we generate an aggregate source count that we represent with a time series of 486 elements (one source count per day). The choice of 1 day for the time scale is quite arbitrary and depends on the type of phenomena we want to discover, as outlined in Section 1. To take advantage of the distributed aspect of the sensors, we create the time series of the attacks by considering them separately by platform. In total we considered a set of 1268 time series which had at least one peak of activity with a minimum of 10 sources for a given day. Fig. 1 illustrates this first step by showing two different attacks observed on two different sensors. Although they target completely different ports (5900/TCP and 1433/TCP), we observe strong temporal similarities between these attacks (mostly around day 100), which is quite likely due to a common root cause. This has been identified as "multi-headed attack tool" in Pouget et al. (2006b).

3.2.2. *Similarity distance for time series*

To compute the similarity between time series, we need to use a technique that is able to reduce the data dimensionality, so as to wipe out small details or irrelevant variations due to additional noise. We can find numerous appropriate techniques for this purpose in the scientific literature, such as singular value decomposition, piecewise aggregate approximation, discrete Fourier transform, wavelets, etc. The application of one or another technique depends on the intrinsic characteristics of the data, like periodicity, regularity or statistical distribution. The method we chose in this paper, for illustration purposes, is called SAX (symbolic aggregate approximation) and falls in the category of PAA techniques which tend to approximate time series by segmenting them into time intervals of equal size and summarizing each of these intervals by its mean value. An additional step of SAX regarding the PAA technique is to use predetermined breakpoints during the quantization. The quantized time series are then interpreted as a string of symbols since every quantization level is mapped to a given symbol of a finite alphabet. SAX provides a lower-bounding distance measure that is easy to interpret, and which can be easily used in our method to decide if two data series are similar or not. More details about the SAX technique can be found in Lin et al. (2003).

Note that any other distance that the analyst considers as relevant can be used transparently in our framework, as long as it has the transitive property between any triplet of vertices.

[1] By definition, we consider an attack session to be finished if no more packets are sent by the source for more than 25 h after the last received packet.

[2] Note that we do not consider the backscatter traffic for this study.

Some care must be taken in setting the appropriate thresholds for the similarity measurements, but this can be easily done by inspecting some typical data series. In general, by setting a very high similarity degree on the measurement given by SAX (e.g. above 90%), we obtain fairly good results. For data series with an extremely ''bursty'' pattern (i.e. only a few high peaks, and almost no activity elsewhere), some additional care must be taken when applying SAX. That is, we also define a local similarity degree between two time series in order to measure how similar they are w.r.t. their peak values situated in the upper 0.975-quantile. We refer the interested reader to Appendix for a more detailed description of this SAX adaptation.

Fig. 2 illustrates two attack time series that are summarized with SAX, using a compression ratio of 3 days (on the *x*-axis) and an alphabet of eight symbols (on the *y*-axis). Those two similar time series correspond to the network activity of the W32/Allaple.B worm as observed on two different sensors; according to our measurements, the outbreak of this worm commenced around November 5th, 2006. Allaple.B is a polymorphic worm that spreads to other computers by exploiting common buffer overflow vulnerabilities, including: SRVSVC (MS06-040), RPC-DCOM (MS04-012), PnP (MS05-039) and ASN.1 (MS04-007) (Sophos Threat Analysis). To conclude this illustration, we obtain with SAX a global similarity measure of 92.3% between the two time series, which seems to be quite consistent.

3.2.3. *Grouping time series*

On a commodity desktop, Algorithm 2.1 took about 209 s to perform the clustering of the time series with a quality threshold of 85%. A total of 164 cliques were found, with less than 3% inconsistencies, i.e. the ratio of time series showing a pattern that is significantly different from the others of the same clique. In total, 755 time series have been classified into cliques. After a visual inspection of each clique, we note that the quality of the results is high. We also assess the consistency of the results by computing for each clique the minimum similarity degree between a pair of time series which we term *clique quality*. We find that 92 cliques have a quality above 90% and 68 cliques have a quality between 80% and 90%. Only four cliques have a quality under 80% (e.g. around 77%). We now turn to the analysis of the results for which we give a detailed description in the next subsection.

3.3. *Results' analysis*

Table 2 gives an overview of our clique-based classification made on 1268 attack time series. In total, we obtained 164 cliques that represent 52% of the whole traffic in terms of attacking sources. We made an in-depth analysis of each group of time series. We observe that each clique of attack patterns can be seen as the expression of an attack phenomenon, or at least a part of its root causes. Due to space limitations, we cannot reproduce all details of our clique analysis. So, we give below only the essential part of our findings to illustrate the feasibility and the usefulness of this approach. Surprisingly, we observe only three broad classes of activities:

(i) *Continuous activity*: 19 cliques, containing 58 time series account for more than a third of the traffic volume (in terms of sources). The involved attacks show a continuous activity pattern, typically several months, and they are correlated on many different sensors. After investigation, we found out that this voluminous traffic was mainly due to Messenger Spammers (on UDP ports) and some classical network worms. Fig. 3(left) illustrates one of these cliques, which has grouped attacks related to the Allaple worm (Sophos Threat Analysis). This kind

Table 2 – Cliques results' overview

Classes of activities	No. of cliques	No. of time series	No. of sources	Main port sequences	Plausible root causes
Continuous	19	58 (4.6%)	581,136 (33.4%)	|1026U|1027U|1028U |I|139T|445T |1434U |135T |I	Scam on Messenger Svc Classical worms (Allaple.B, Slammer) Continuous scan
Sustained bursts	24	107 (8.4%)	204,336 (11.8%)	|I|445T|139T |5900T and |1433T |2967T, |2968T |445T	Large botnet activity Multi-headed worm Sustained scan activities
Ephemeral spikes (epiphenomena)	109	554 (43.7%)	98,610 (5.7%)	|6644T, |17838T, |6769T |5168T, |53842T, |12293T |6211T, |50286T, |9661T |135T, |139T, |445T |2967T, |2968T |1025T, |80T, |1433T |5900T, |5901T |4662T, |4672T	Ephemeral probes on unusual high TCP ports Targeted scans on common Windows ports (NetBios, Symantec, RPC, VNC, etc.) Misconfigurations (P2P)
Inconsistencies or misclassifications	12	36 (2.8%)	25,716 (1.5%)	|135T, |139T, |445T |1433T	Background noise on common services

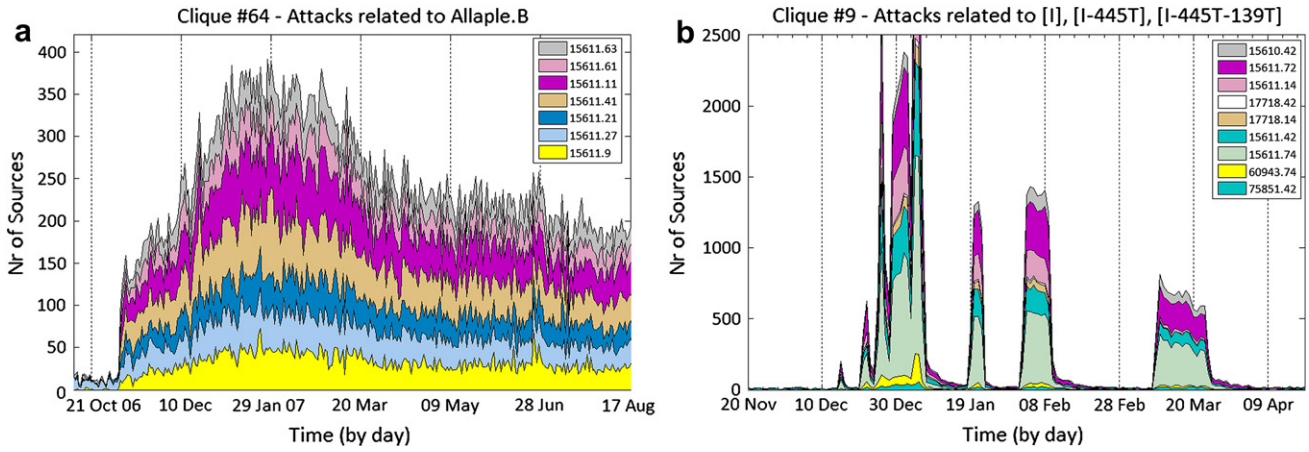

Fig. 3 – (Left) Example of *continuous activity*: those time series belong to a clique of attacks related W32/Allaple.B on seven different sensors (port sequences: |I, |I|139T and |I|139T|445T). (Right) Example of *sustained bursts*: these time series belong to a clique of attacks related to attack activities on three different sensors, targeting |I, |I|445T, |I|445T|139T and |I|445T|80T.

of attack was observed on 12 different sensors located in seven different IP subnets (class A) and most of those attacks are still active at the end of our analysis timeframe. The quality of this clique (i.e. the minimum similarity value between a pair of time series) comes to 84.9%.

(ii) *Sustained bursts*: 24 cliques containing 107 time series representing about 11% of the traffic. The attacks involved have a typical *wave-like* activity pattern; they maintain their activity on a group of sensors for several days and then disappear for a while. Fig. 3(right) illustrates one of these cliques, for which the quality level is close to 86%. According to our in-depth analysis, likely root causes for such phenomena can be attributed to large botnet attack waves, multi-headed worms (Pouget et al., 2006b) or general scan activity. Regarding botnets, we observe typical diurnal activity patterns when analyzing some of those sustained bursts with a time scale of 1 h, which probably indicates that we face botnet propagation attempts coming from zombie machines (see Dagon et al., 2006). This is illustrated in Fig. 4 where we show a detailed view of the first attack wave of the attacks represented in Fig. 3(right).

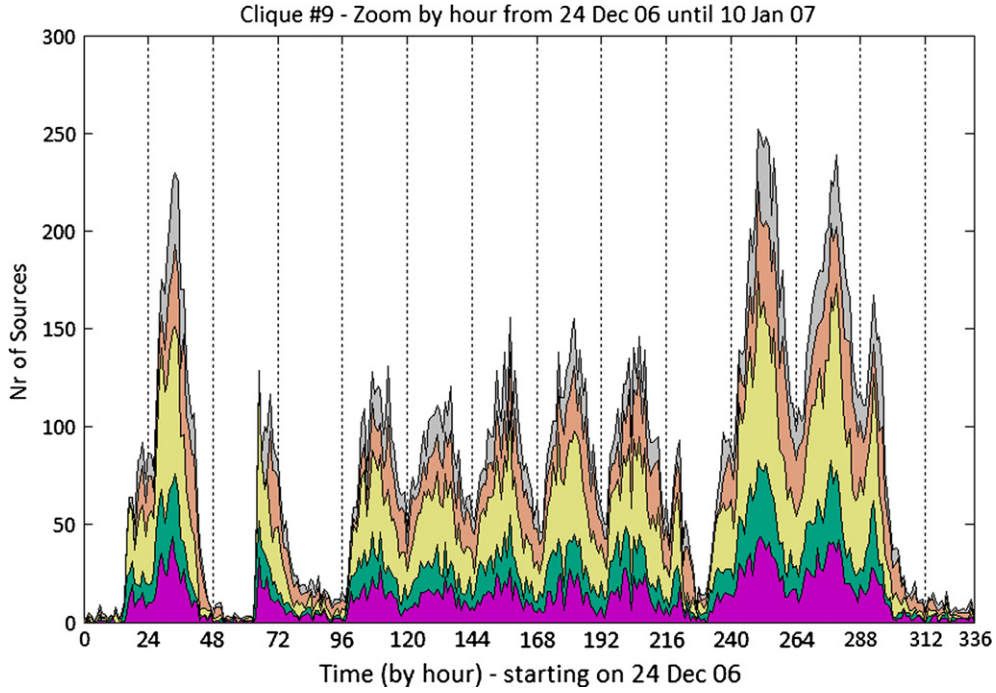

Fig. 4 – A zoom by hour of the first attack wave of the clique represented in Fig. 3(right). We observe a diurnal activity pattern with a period of 24 h, which is likely due to a botnet propagation attempt.

The three other attack waves of that figure have very similar diurnal patterns. So, even if the attacks differ by their port sequences and by the IP addresses of the sources, we conjecture that all these attacks originate from the same large botnet (or different botnets controlled by the same entity).

(iii) *Ephemeral spikes*: a large majority of the cliques are related to very ephemeral attacks targeting one or a few sensors on the same day (Fig. 5). Although numerous, they represent a small traffic volume. Many of these probes were observed on quite unusual high TCP ports. In light of our detailed analysis, we hypothesize this can be due either to small botnet probes, targeted scan activities or misconfigurations (e.g. P2P traffic).

We have not yet set ourselves the goal of understanding completely the phenomena that may hide behind the cliques. We note, however, that clustering attack patterns with respect to different features provides very good candidates for further investigation.

4. Related works

Some facets of our work are related to the works presented in Li et al. (2005) and Chen et al. (2006), where the authors develop correlation and indexing techniques that enable the evaluation of coordinated scanning along several dimensions, mainly spatial and temporal dispersions. Another closely related work, which inspired us in some ways, is the seminal work of Yegneswaran and colleagues on ''Internet situational awareness'' (Yegneswaran et al., 2005). In that work, they explore ways to integrate honeypot data into daily network security monitoring, with the purpose of effectively classifying and summarizing the data to provide ongoing *situational awareness*. A specific emphasis was also given to the problem of accurately classifying large-scale events related to botnet sweeps and worm outbreaks.

Regarding network traffic classification, other interesting approaches have been developed for classifying traffic flows according to the tools or the applications that generate them. For instance in Karagiannis et al. (2005), the authors design a multi-level approach called *BLINC* that aims to classify traffic ''in the dark'', i.e. without knowledge about the source applications of the network flows. It is based on identifying the behavioral patterns of hosts at three increasing levels of details: the social, functional and application levels. This approach is mostly designed for analyzing traffic at a border router or a core switch; hence it is not well-suited for a honeynet's limited view. In McHugh (2004) the author proposes a framework based on sets of IP addresses for monitoring and analyzing high speed networks and very large data sets containing netflows (while we use full packet traces gathered with honeypots). Finally, Kannan et al. (2006) have proposed a framework for aggregating the network connections to application sessions by using a Poisson-based statistical technique, where a session is defined as a set of connections between a pair of hosts with a specific purpose (legitimate or malicious) and eventually involving multiple protocols. Although similar in certain aspects, our approach is very different both in terms of techniques and final objectives. Kannan et al. address the problem of effectively modeling source behavior at the application level, so as to characterize network traffic and to detect behavioral anomalies.

5. Conclusions

We have presented a systematic analysis framework for discovering groups of attack traces having similar patterns. Our framework includes a certain flexibility that allows analysts to plug in different feature vectors and appropriate similarity metrics to be used in the clustering step, depending on the attack features they might find relevant to investigate. The contribution is being able to draw knowledge out of honeynet data by discovering attack patterns via attack trace similarity, rather than via a rigid signature. To illustrate the

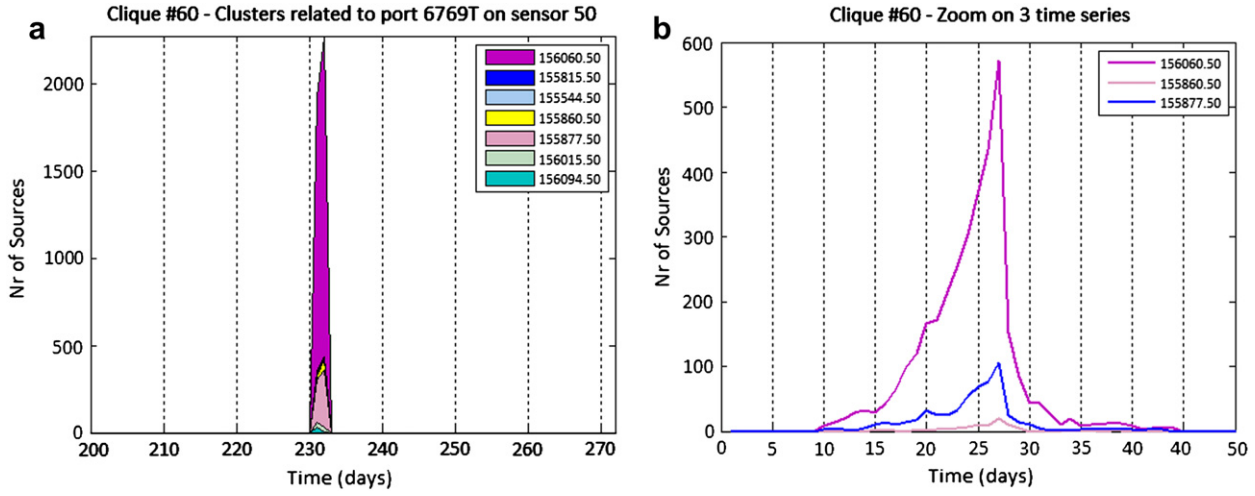

Fig. 5 – (Left) Example of *ephemeral spike*, where the superimposed peaks belong to a clique of similar attacks targeting a single sensor on port |6769T. These attacks had slightly different network fingerprints, but now appear to be related to the same root phenomenon. (Right) A zoom by hour for three of those peaks.

framework, we emphasized the temporal correlations between attacks. Through the experimental results, we showed how clique-based analysis can assist honeypot forensics by highlighting the striking correlation patterns between similar or even completely different attacks. The results of our clustering applied to time series analysis enabled us to identify activities of several worms and botnets in the traffic collected by the honeypots.

The ultimate goal is to help the analyst figuring out if a new imminent attack can be attributed to the very same group of attacks, by looking at different correlation patterns along multiple dimensions (geographical, temporal, IP subnets, etc.). We are considering ways to improve our framework so that it can be applied or used in real-world data sets to detect imminent or latent intrusions. To achieve this goal, we will adapt the framework so it can systematically cross-correlate the network traces with respect to different attack features.

Acknowledgements

This work has been partially supported by the European Commission through project FP7-ICT-216026-WOMBAT funded by the seventh framework program and by the Resist Network of Excellence (contract number 026764). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

Appendix. A robust similarity metric for time series

To measure the similarity between time series, we have adapted SAX (symbolic aggregate approximation; Lin et al., 2003), a PAA[3] technique which tends to approximate time series by segmenting them into intervals of equal size and summarizing each of these intervals by its mean value. An additional step of SAX is to use predetermined breakpoints during the quantization. The quantized time series are then interpreted as a string of symbols since every quantization level is mapped to a given symbol of a finite alphabet. An SAX representation of a time series T of length N can be denoted by $W_T(N, w, \alpha)$, where N is the number of elements in T, w is the number of segments in the SAX representation of T, and α is the alphabet size (number of quantization levels). The ratio $r = N/w$ is called the compression ratio. A value of $r = 10$ means that 10 elements of T are mapped to a single symbol in W_T. Prior to the quantization, the time series are standardized w.r.t. their mean and standard deviation. SAX provides a distance measure that lower bounds the original distance [e.g. Euclidean, see Lin et al. (2003) for the proof]. It estimates the distance between two time series T_1 and T_2 as the distance between their SAX representations W_{T_1} and W_{T_2}. For the sake of efficiency, inter-symbol distances can be pre-computed based on a normal distribution of the time series and loaded in a lookup table. Let T_1 and T_2 be two time series of the same length N, then the minimum *distance* given by SAX can be calculated as follows:

$$\mathrm{SAX}(W_{T_1}, W_{T_2}) = \sqrt{\frac{N}{w}}\sqrt{\left(\sum_{i=1}^{w} \mathrm{dist}(W_{T_1}(i), W_{T_2}(i))\right)^2}$$

The *dist*() function returns the inter-symbol distance and is implemented using a table lookup for better computational efficiency. Fig. 6 gives an example of a time series from our data set that has been "symbolized" with SAX, using a compression ratio of 2 days and an alphabet of six symbols. This time series corresponds to probes on port 2968/TCP (Symantec).

Global similarity measure. We define a *global similarity* index between a pair of symbolic time series by using the formula below. The denominator represents the largest lower-bounding distance that we can obtain theoretically with SAX, as calculated for two symbolic series W_T and $\tilde{W}_T$ that have the largest difference between every pair of symbols for all points.

$$\mathrm{SIM}_G(W_{T_1}, W_{T_2}) = 1 - \frac{\mathrm{SAX}(W_{T_1}, W_{T_2})}{\mathrm{SAX}\left(W_T, \tilde{W}_T\right)}$$

Local similarity measure. SAX has by design an important limitation. The predetermined breakpoints used during the quantization are defined assuming that time series have a *Gaussian distribution*. Many attacks though exhibit a temporal pattern with only a few spikes and zeros or very small values elsewhere. This leads to a skewed distribution and a median value close to zero. In that case, the high similarity degree given by SAX is obviously biased, due to the large amount of very small or zero-values. We can illustrate this with a typical example of two time series that have a few high peaks (Fig. 7). The global SAX similarity between those two signals is about 90%, which is definitively inconsistent with the shapes of the time series.

To circumvent this problem, we define also a *local similarity measure* which compares only the relevant variations of the two signals, on a point-by-point basis. By "relevant" we mean the values that are situated in the upper 0.975-quantile of the time series. Thus, for time series with a few high peaks and zeros elsewhere, this measure considers only the similarities between those peaks, since they are located in the upper 0.975-quantile of the distribution. Let X be the set of points that are situated in the upper quantile UQ for either the first symbolic series W_{T_1} or the second symbolic series W_{T_2}. More formally

$$X_{\mathrm{UQ}} = \bigcup_{k=1}^{2} \left(x_i \middle| W_{T_k}(x_i) > \mathrm{UQ}, \quad \forall i \in \left\{1, \ldots, \left|W_{T_k}\right|\right\}\right)$$

Now we can define the local similarity between two time series as

$$\mathrm{SIM}_L(W_{T_1}, W_{T_2}) = 1 - \frac{\sum_{j=1}^{|X_{\mathrm{UQ}}|} \mathrm{SAX}(W_{T_1}(x_j), W_{T_2}(x_j))}{\sum_{j=1}^{|X_{\mathrm{UQ}}|} \mathrm{SAX}\left(W_T(x_j), \tilde{W}_T(x_j)\right)}$$

Here the symbolic series W_T and $\tilde{W}_T$ are generated so as to obtain the largest difference between every pair of symbols, hence we compute for the local points X_{UQ} the highest lower-bounding distance that we can obtain with SAX. By applying the formula given by SIM_L, we obtain a local similarity of 0% between the two time series represented in Fig. 7, which is much

[3] Piecewise aggregate approximation.

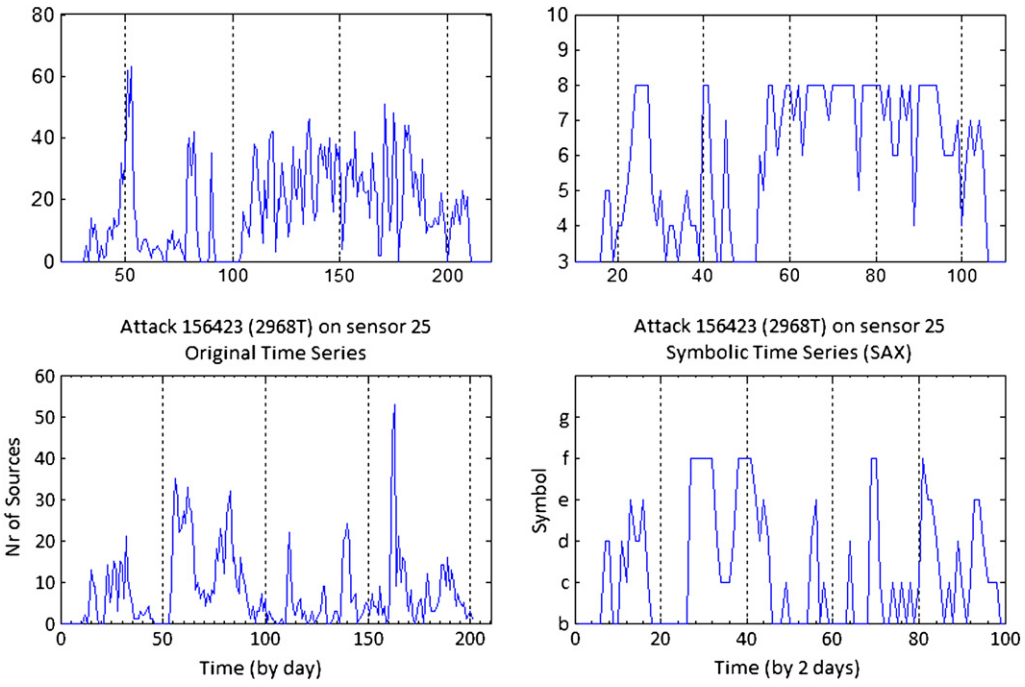

Fig. 6 – An example of an attack time series (left) and its symbolic representation (right) obtained with SAX (compression ratio of 2 days).

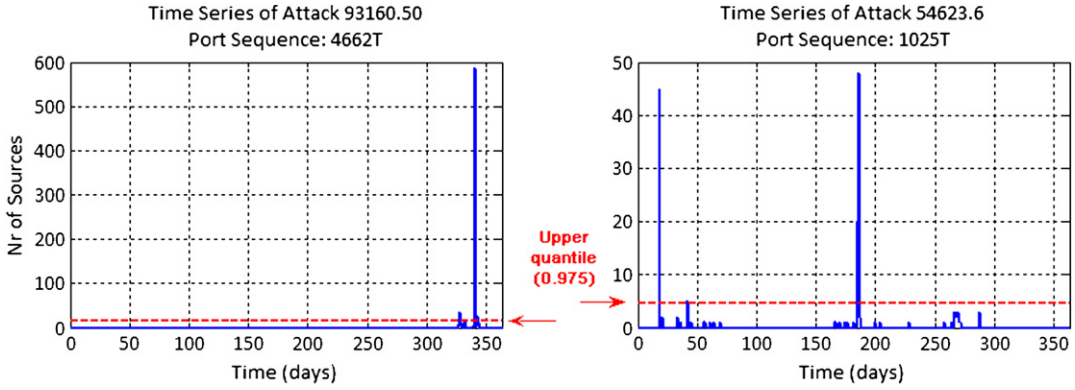

Fig. 7 – The inconsistent global similarity degree between those time series (about 90%), calculated with SAX, is biased due to the large amount of small and zero-values, leading to a median close to zero. The definition of a local similarity measure on the values in the upper quantile allows the two dissimilar signals to be discriminated.

more consistent than the global similarity value of 90%. So, by combining both similitudes (i.e. global and local), we obtain now an effective similarity measure for comparing our time series.

REFERENCES

Baecher P, Koetter M, Holz T, Dornseif M, Freiling F. The nepenthes platform: an efficient approach to collect malware. In: Proceedings of the 9th international symposium on recent advances in intrusion detection (RAID); September 2006.

Bomze I, Budinich M, Pardalos P, Pelillo M. The maximum clique problem. In: Handbook of combinatorial optimization, vol. 4. Boston, MA: Kluwer Academic Publishers; 1999.

Chen B, Yegneswaran V, Barford P, Ramakrishnan R. Toward a query language for network attack data. In: Second IEEE workshop on networks meets databases (NetDB '06); April 2006.

Chen Z, Gao L, Kwiat K. Modeling the spread of active worms; 2003.

Collins MP, Shimeall TJ, Faber S, Janies J, Weaver R, De Shon M, et al. Using uncleanliness to predict future botnet addresses. In: IMC '07: Proceedings of the seventh ACM SIGCOMM conference on Internet measurement. New York, NY, USA: ACM; 2007. p. 93–104.

Dagon D, Zou C, Lee W. Modeling botnet propagation using time zones. In: Proceedings of the 13th annual network and distributed system security symposium (NDSS'06); February 2006.

De Smet F, Mathys J, Marchal K, Thijs G, De Moor B, Moreau Y. Adaptive quality-based clustering of gene expression profiles. Journal of Bioinformatics 2002;18:735–46.
DShield. Available from: <http://www.dshield.org>.
Internet motion sensor. Available from: <http://ims.eecs.umich.edu/>.
Gary Augustson J, Minker Jack. An analysis of some graph theoretical cluster techniques. Journal of the ACM 1970;17(4):571–88.
Jain AK, Dubes RC. Algorithms for clustering data. Prentice-Hall advanced reference series; 1988.
Kannan J, Jung J, Paxson V, Koksal C. Semi-automated discovery of application session structure. In: IMC '06: Proceedings of the sixth ACM SIGCOMM conference on Internet measurement. ACM; 2006. p. 119–32.
Karagiannis T, Papagiannaki K, Faloutsos M. Blinc: multilevel traffic classification in the dark. SIGCOMM Computer Communication Review 2005;35(4):229–40.
Li X, Bian F, Zhang H, Diot C, Govindan R, Hong W, et al. Advanced indexing techniques for wide-area network monitoring. In: First IEEE international workshop on networking meets databases (NetDB); 2005.
Lin J, Keogh E, Lonardi S, Chiu B. A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the eighth ACM SIGMOD workshop on research issues in data mining and knowledge discovery, California, USA; 2003.
McHugh J. Sets, bags, and rock and roll: analyzing large data sets of network data. In: ESORICS; 2004. p. 407–22.
Moore D, Shannon C, Voelker GM, Savage S. Network telescopes: technical report. CAIDA; April 2004.
Pardalos PM, Xue J. The maximum clique problem. Journal of Global Optimization 1994;4:301–28.
Pavan M, Pelillo M. A new graph-theoretic approach to clustering and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2003.
Provos N. A virtual honeypot framework. In: Proceedings of the 13th USENIX security symposium; 2004.
Pouget F, Dacier M. Honeypot-based forensics. In: AusCERT2004, Brisbane, Australia; 23rd–27th May 2004.
Pouget F, Dacier M, Zimmerman J, Clark A, Mohay G. Internet attack knowledge discovery via clusters and cliques of attack traces. Journal of Information Assurance and Security March 2006a;1(1).
Pouget F, Urvoy Keller G, Dacier M. Time signatures to detect multi-headed stealthy attack tools. In: 18th annual FIRST conference, June 25–30, 2006, Baltimore, USA; June 2006b.
Riordan J, Zamboni D, Duponchel Y. Building and deploying billy goat, a worm-detection system. In: Proceedings of the 18th annual FIRST conference; 2006.
Sophos threat analysis. W32/allaple-b. Available from: <http://www.sophos.com/>.
Team Cymru. Darknet project. Available from: <http://www.cymru.com/darknet/>.
The Leurre.com project. Available from: <http://www.leurrecom.org>.
Werner T. Honeytrap. Available from: <http://honeytrap.mwcollect.org/>.
Xinshun X, Jun M, Jingsheng L. An improved ant colony optimization for the maximum clique problem. In: Third international conference on natural computation (ICNC 2007), vol. IV; 2007. p. 766–70.
Yegneswaran V, Barford P, Paxson V. Using honeynets for internet situational awareness. In: Fourth ACM SIGCOMM workshop on hot topics in networking (Hotnets IV); 2005.

Olivier Thonnard graduated as an Engineer in Telecommunications from the Royal Military Academy, in Belgium. He also holds a Master in Applied Computer Science from the Vrije Universiteit Brussel. He is currently completing a PhD, focused on honeypot traffic analysis, at EURECOM (France).

As an Officer, he is teaching at the Polytechnic Faculty of the Royal Military Academy, where he is involved in several courses related to computer and network security. His research interests include intrusion detection, honeypot technology, and network traffic analysis. His current research activities are closely related to the analysis of Internet threats. Most of his work concentrates on the identification of attack patterns and trends in network traces, by means of classification and correlation techniques applied to large sets of real-world attack traces. For that purpose, he is developing new specific data mining techniques based on clustering, correlation methods, and clique algorithms.

Marc Dacier has joined Symantec as the director of Symantec Research Labs Europe in April 2008. From 2002 until 2008, he was a professor at EURECOM, France (www.eurecom.fr). He also was an associate professor at the University of Liège, in Belgium.

From 1996 until 2002, he worked at IBM Research as the manager of the Global Security Analysis Lab. In 1998, he co-founded with K. Jackson the "Recent Advances on Intrusion Detection" Symposium (RAID). He is now chairing its steering committee. He is or has been involved in security-related European projects for more than 15 years (PDCS, PDCS-2, Cabernet, MAFTIA, Resist, WOMBAT, FORWARD). He serves on the program committees of major security and dependability conferences and is a member of the steering committee of the "European Symposium on Research for Computer Security" (ESORICS). He was a member of the editorial board of the following journals: IEEE TDSC, ACM TISSEC and JIAS. His research interests include computer and network security, intrusion detection, network and system management. He is the author of numerous international publications and several patents.