Time Signatures to detect multi-headed stealthy attack tools

Abstract.
In this paper, we present a method to detect the existence of sophisticated attack tools in the Internet that combine, in a misleading way, several exploits. These tools apply various attack strategies, resulting into several different attack fingerprints. A few of these sophisticated tools have already been identified, e.g. Welchia. However, devising a method to automatically detect them is very challenging since their different fingerprints are apparently unrelated. We propose a technique to automatically detect their existence through their *time signatures*. We exemplify the interest of the technique on a large set of real world attack traces and discover a handful of those new sophisticated tools.

1. Introduction

Empirical study of attacks in the Internet is an emerging research field. Early identification of new threats is a key element to implement counter-measures in a timely manner. Previous work focused on the propagation models of worms in general [11,32,38] or the analysis of specific worms [15,25,26,31]. A few initiatives aimed at monitoring worms and attacks propagation in the Internet such as the so-called telescopes (a.k.a blackholes/darknets) or the DShield web site [4,14,39,44]. Those approaches have highlighted the existence and dangers of certain types of malware (see for instance the recent Witty worm analysis in [5]).

In this paper, we are interested in the identification of a specific class of tools that we call multi-headed stealthy attack tools. They combine several known exploits within a single piece of software. This is not a new technique, as the very first worm, the Internet worm, did already contain several infection techniques [31]. However, the specificity of this class of attacks is that only one of the available exploits will be used to launch an attack against a given target. In other words, machines targeted by those multi-headed tools see different attacks originating from different sources that can easily be interpreted as different tools. As a consequence, the fact that several exploits have been combined within a single piece of code remains invisible to the victims as long as the malware is not captured and analyzed. If the spreading of the malware is not too aggressive, its existence may remain unknown for a while. A few of these sophisticated tools have already been identified, e.g. Welchia. However, this is the result of their malicious activities on users' machines and there is a high probability that some other similar, but more stealthy, tools of this type are currently active in the Internet. The identification of these tools remains a great challenge.

The main contribution of this paper is a method to detect their existence. The key idea is to identify the similarities between the *time signatures* of (apparently) different attack tools. By time signatures, we mean the time series of the number of different sources using a given attack per day. Assume for instance that we build a multi-headed tool that combines two attack tools X and Y and that we can record the number of attacks launched by X or Y each day in the Internet. If we make the assumption that each execution of the tool will randomly lead to launch either attack X or attack Y, and if we also assume that targeted machines are equally vulnerable to X and Y, then, on average, we should observe approximately the same number of attacks of type X or Y. This equality should hold for every single day. In other words, the time signatures of X and Y should look similar: they will exhibit an increase (resp. decrease) if more machines get compromised by (resp. cured from) the considered multi-headed tool. In the case where one attack variant is more successful than the other, time correlations (ups and downs) should still exist although the amplitudes of the curves should be different. On the other side, if one considers two independent tools, there is no reason for their time signatures to be similar over a long period of time[1]. Thus, by detecting similarities between the time signatures of (apparently) different attack tools, it is possible to find out that

[1] We provide a formal proof of this claim in Section 4.

these tools are combined within a single malware code. This, in fact, is the only existing technique we can use to identify these malware apart from fastidious reverse engineering techniques.

The structure of the paper is as follows. In Section 2, we present the experimental setup used to identify attack tools based on a set of characteristics (sequence of targeted ports, attack duration, number of exchanged packets, etc) excluding their time signature. In Section 3, we detail our method to systematically identify similarities between time signatures of attack tools and thus identify multi-headed tools. Section 4 presents the results of applying the method to our dataset. Conclusions and future work are presented in Section 6.

2. Experimental Environment

2.1. A Unique Dataset

Our dataset has been obtained with a honeypot platform (see [29] for details). It emulates three virtual machines running different operating systems (Windows 98, Windows NT Server, Linux RedHat 7.3) with various services. Logs are centralized in a database that contains, for each attack, a large variety of information, such as:

- Raw packets (entire frames including the payloads are captured with tcpdump);
- IP geographical localization obtained with NetGeo, MaxMind and IP2location;
- Passive Operating System fingerprinting obtained with Disco, p0f and ettercap;
- TCP level statistics using TCPstat;
- DNS reverse lookup, whois queries, etc…

The initial motivations behind the platform architecture were to gather statistical data in a long-term perspective. The platform has been active for 20 months (600 days). During this period, approximately 80,000 different IP addresses have been observed, i.e. about 135 distinct IP addresses, on average, per day. Those addresses originate from 91 different countries. Note also that only a negligible number of those addresses have been observed twice, that is on two different days.

We have also started to deploy similar platforms in other locations. Currently, our distributed honeypot environment consists of 30 entities located in 20 different countries, covering the 5 continents. However, data obtained with those new platforms do cover a much shorter period of time and are thus not suitable for the type of time analysis carried here. Therefore, we will focus in this work only on data provided by our initial platform.

2.2. Attacking tools clustering engine

We have presented in [12, 13] a method to identify the tools behind the observed attacks thanks to a clustering algorithm detailed in [28]. It is based on the following notions:

- *Attack Source*: One IP address that targets a honeypot platform on a given day. Thus, the same IP address seen in two different days corresponds to two different *attack sources* (see [13] for more details).
- *Ports Sequence*: An ordered list of ports targeted by an *attack source* on a virtual machine. For instance, if source A sends requests to port 80 (HTTP), and then to ports 8080 (HTTP Alternate) and 1080 (SOCKS), the associated *ports sequence* will be {80;8080;1080}.

The clustering algorithm relies on a small number of input parameters. Among them, there is:

- The number of virtual machines targeted by one source in the honeypot environment;
- The number of packets sent by one source to each machine;
- The total number of packets sent by one source to the whole environment (the three virtual machines);
- The duration of the observation, from the first packet sent by the source to the last one;

- The average inter-arrival time between packets sent by the source;
- The associated ports sequence.

Some parameters are generalized by means of hierarchy trees. This reduces the risk of errors induced by packet losses. Additionally, an association-rule based algorithm is applied to group all attacks sharing common values for these 8 parameters. In a second step, we refine the initial grouping. This is achieved with the Levenshtein distance (see [28] for details) that allows comparing strings obtained from the concatenation of the packet payloads. If the Levenshtein distances are not uniform within one cluster, we split it into smaller and more homogeneous clusters. As a result, the clustering technique detailed in [28] groups attacks into clusters defined as:

Definition: A **cluster** is a set of IP Sources having the same attack fingerprint (values of the parameters) on a honeypot platform.

Our algorithm, when applied to our dataset, generates around 2000 clusters. However, half of them contain only a single attacking source. Those clusters represent a negligible fraction of the total number of attacks (about 2%). On the other hand, as can be seen from Table 1, the 965 largest clusters represent 98% of the attacks and the largest 137 clusters count for 76% of all the attacks. Table 1 further indicates the average number of attack sources per cluster.

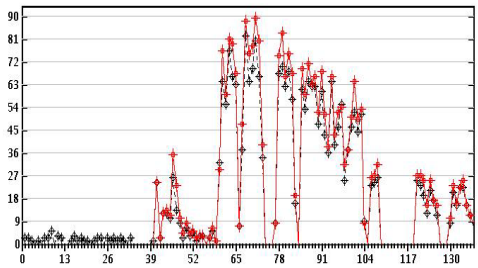
# of clusters C_k	% of the observed attacks	Average # of sources per cluster
965	98%	52
654	80%	63
137	76%	283
46	63%	704

Table 1: Information about the clusters generated for the 600 days dataset

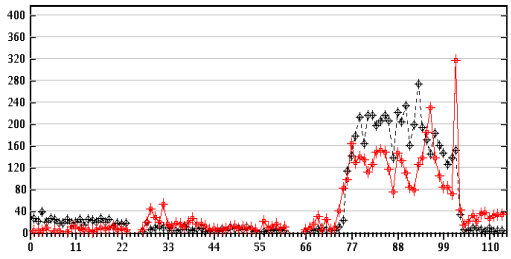
3. Time Analysis of Attack Processes

3.1. First Observations

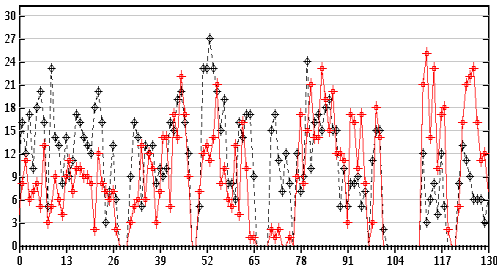
We have shown in [12,13] that the clusters obtained by our clustering algorithm are coherent in terms of their contents and reveal worth-investigating attack features (like the geographical locations of the attacks, the attack ordering, the raw profile of attacking machines, etc). We have been able to name of few of those clusters by comparing the fingerprints of some known tools on our honeypot, obtained in a control environment, to the fingerprints obtained in the wild. However, this task is tedious, and only a few dozens of tools have been clearly identified so far.
To further complement our clustering method, we looked at the time behavior of the clusters. Indeed, as illustrated by Figures 1(a-d), where the y-axis represents the number of IP addresses associated to each cluster, as a function of time (with a granularity of 3 days on the x-axis), some clusters clearly exhibit a similar time evolution. It is however striking that those similar (w.r.t. time) clusters correspond to very different attack fingerprints. What is more, Figures 2a) and b) further highlight that the global activities against some of those ports (by summing the activities of all the clusters targeting those ports) are completely uncorrelated. Without going into the details, intervals between brackets show periods where no evident time correlation is noticeable. An important conclusion from those examples is that some temporal correlations exist between attack fingerprints that seem otherwise unrelated. This result clearly deserves further investigation. The next sections of this paper aim at defining a methodology to find out those similarities and apply this methodology to our dataset.



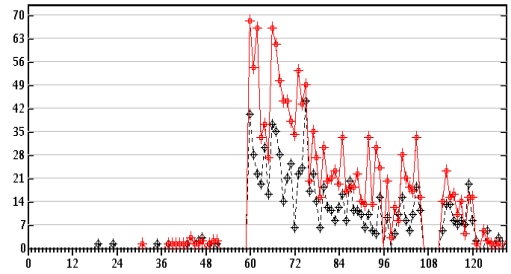
a) 2 clusters = 2 attacks targeting port {135} and ports {135,4444} resp.



b) 2 clusters = 2 attacks targeting port {80} and port {135} resp.

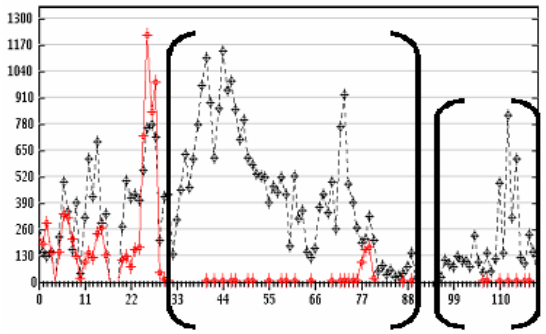


c) 2 clusters = 2 attacks targeting port {1433} and port {139} resp.

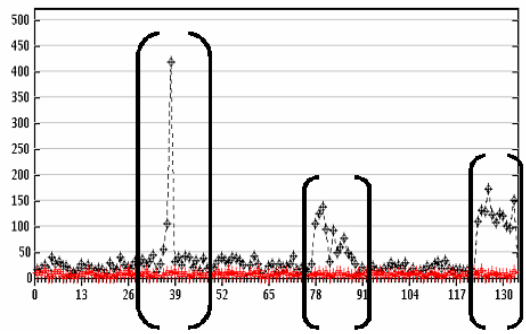


d) 2 clusters = 2 attacks targeting port {445} and ports {5554,1023,9898} resp.

Figure 1: Examples of time correlation between clusters



a) Number of attacks having targeted port 80 or attacks having targeted port 135



b) Number of attacks having targeted port 139 or attacks having targeted port 1433

Figure 2: Observed activities on some targeted ports

3.2. Methodology

Time series analysis can provide valuable information, e.g. trends [16], to administrators in charge of detecting anomalous behaviors or intrusions in their network [44]. In the present work, we use a time series analysis to discover similarities in the time behavior of attack processes identified by our clustering algorithm (see Section 2.2). By similar time behavior, we mean that the amount of attacks with respect to time of two similar processes should follow the same trends (ups and downs), but we do not impose that the actual values of the two time series are similar. In practice, this means that before testing the similarity of two time series, we first normalize them. We impose two additional constraints on the method that we use:

- First, the method should use an adequate technique to summarize data and thus wiping out details on the variability of the time series so as not be too conservative;
- Second, the distance returned by the method should be easy to interpret, leading to a clear decision on whether the time series are similar or not. This is of utmost necessity as the number of time series that we collect continuously increases with time.

The first step of the method will thus consist in summarizing data. Several techniques have been proposed to do this. They can be classified into data-adaptive techniques and non data-adaptive techniques, depending on whether the basis on which data is projected is derived from the data itself or not [30]. In the first category, we find, for instance, the Singular Value Decomposition (SVD), the Piecewise Aggregate Approximation (PAA) or the random projection (sketch) techniques. Discrete Wavelet Transform (DWT) or Discrete Fourier Transform (DFT) [3,9,10,21] are examples of techniques from the second category. The decision to use one type of method or the other depends on some intrinsic characteristics of the data, like periodicity or regularity [30].

We have decided to use SAX (Symbolic ApproXimate Aggregate), a recently proposed PAA technique, that transforms time series into strings and provides an easy-to-interpret distance between the resulting strings [18,20,24]. We have not used a Fourier analysis because of the clear lack of periodicity of our time series and we did not consider a wavelet approach due to the wide range of observed behaviors of our time series that hinder a clear choice for a common wavelet basis (e.g. Haar or Daubechies) for all of them. SAX is a recent and popular method with interesting proven properties.

In the remaining of this section, we present SAX and the method used to select its parameters for the need of our application.

3.3. Symbol Aggregate Approximation

Symbolic Aggregate Approximation (SAX) is a Piecewise Aggregate Approximation (PAA) technique [18,20,24]. PAA methods stem from the observation that most time series datasets can be approximated by segmenting the time series into intervals of equal size and summarizing each of these intervals by its mean value [17]. SAX adds one step to the PAA technique as the PAA representation of a given time series is further quantized using predetermined breakpoints (quantization levels, see [24]). The quantized time series is interpreted as a string of characters as each quantization level is mapped to one given character. Figure 3 presents an example where a time series is mapped to the string "ccccccccccccgffedc". In the following, we sill say that the SAX representation of a time series T of length N is denoted by $W_T(N,w,\alpha)$, where: i) N is the number of elements in T, ii) w is the number of elements in the SAX representation of T (i.e. the length of W_T) and iii) α is the alphabet size (number of quantization levels). In the example provided in Figure 3, we have N=600, w=20 and α=7. For the sake of conciseness, we will use W_T instead of $W_T(N,w,\alpha)$, whenever possible.

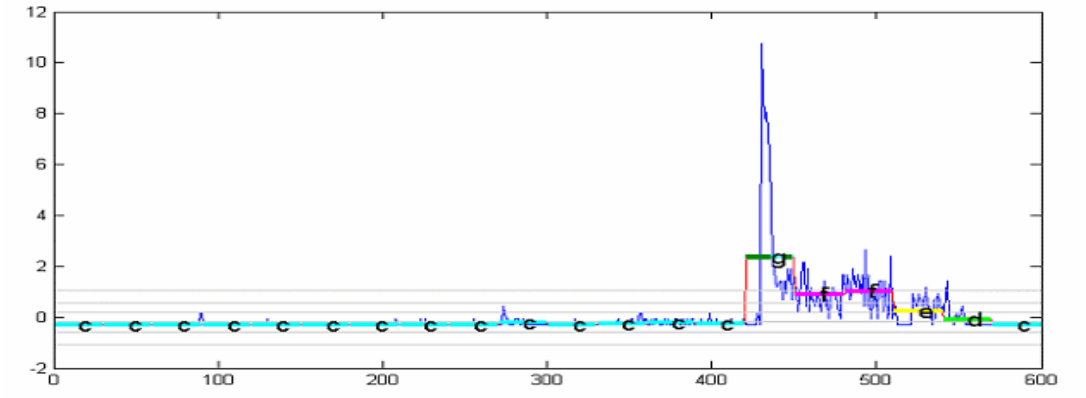


Figure 3: Example of a SAX representation of a Time Series (N=600, w=20, α=7)

The ratio $r = \frac{N}{w}$ is called the compression ratio. For instance, a value of r=10 means that 10 elements of T are mapped to a single symbol in W_T.

The computation of W_T is performed in three steps (see Figure 4):

1. T is first normalized into $T' = \frac{T - \mu_T}{\sigma_T}$, where μ_T and σ_T are respectively the mean and the standard deviation of T. T' thus has a mean of zero and a variance of one. This makes the similarity measure (the Euclidean distance in our case) invariant to shifting and scaling [19].
2. T' is then transformed into T'' using the PAA technique over intervals of length r. The i^{th} element of T'', T''(i), is computed as follows:

$$T''(i) = \frac{1}{r} \sum_{k=(i-1)r+1}^{ir} T'(k), \quad i \in \{1,..,w\}$$

3. Given that T'' is approximately normally distributed [22], the α quantization levels are chosen so as to maximize the energy of the quantized representation of the time series [24], leading, for a given T'' to its symbolic representation W_T.

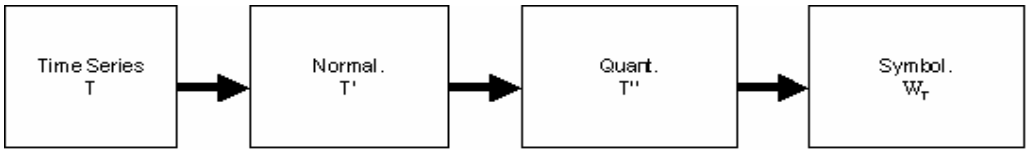


Figure 4: Three steps to get the SAX symbolic representation of T_k

3.4. Finding Similarities with SAX

SAX estimates the distance between two time series T_1 and T_2 of length N as the distance between their SAX representations W_{T1} and W_{T2}. For sake of efficiency, inter symbol distances can be pre-computed and loaded in a lookup table TAB (see [24] for details). The distance between W_{T1} and W_{T2} is computed as follows, where $W_{T1}(i)$ and $W_{T2}(i)$ are the i-th symbols of W_{T1} and W_{T2} respectively:

$$D(W_{T_1}, W_{T_2}) = \sqrt{\frac{N}{w}} \sqrt{(\sum_{i=1}^{w} TAB(W_{T_1}(i), W_{T_2}(i))^2} \quad (1)$$

Our objective is to use SAX to automatically discover similar time behaviors among the attack processes identified by our clustering algorithm (see section 2.2). This means that we must determine a threshold τ such that all pairs of time series (T_1,T_2) such that $D(W_{T1},W_{T2}) \leq \tau$ will be (visually) similar over all the measurement period. The choice of τ depends on the values of the alphabet size α, that controls the levels of details kept in W_T, and the compression ratio r, that controls the time scale at which we will look for similarities. While r is application dependent, there exist no a priori good values for α, though the authors in [20] suggest values between 4 and 10.

We want to observe similarities at a time scale (r values) of around 10 days. Actually, there is no specific requirement for this time scale. Note that we have checked that other values between 4 and 10 days also provide similar results. The choice of α requires a visual inspection of the time series deemed similar by the tool, which can only be done when τ is chosen. We first started by looking at the histogram of the distances between our 137 time series (for a total of 9316 comparisons). An interesting property that we observed is that the shape of the histogram is similar for different values of α and r: there is some mass at zero and the rest of the values are larger than 1. Figure 5 presents a typical example with α=5 and r=10.

To check whether setting τ to 0 was a good choice, we inspected all the time series with distance zero, and some time series whose distances are between 1 and 2 for α=4,5,6 and 7 and a fixed value of r=10. It turned out that:

- α=4 wipes out too many details of the time series leading to time series with a distance of zero that are visually too dissimilar;
- For α=5, the time series with a distance of zero are reasonably similar while the time series at a distance between 1 and 2 are clearly dissimilar;
- Using values $\alpha \geq 6$ is too conservative, as time series that we were considering as similar for α=5 had a distance larger than 1 when using more than 5 symbols to code them even though they were visually very similar over the 600 days period.

Thus, we conclude from the above study that a value of α=5 along with a threshold of τ=0 are adequate choices for our purpose. Note that we also checked that the results are not significantly affected by the choice of r, when r varies between 4 and 12.

As a conclusion, for the rest of the paper all results are based on the application of the SAX method with fixed parameters α=5, r=10 and τ=0 on the data set described here below.

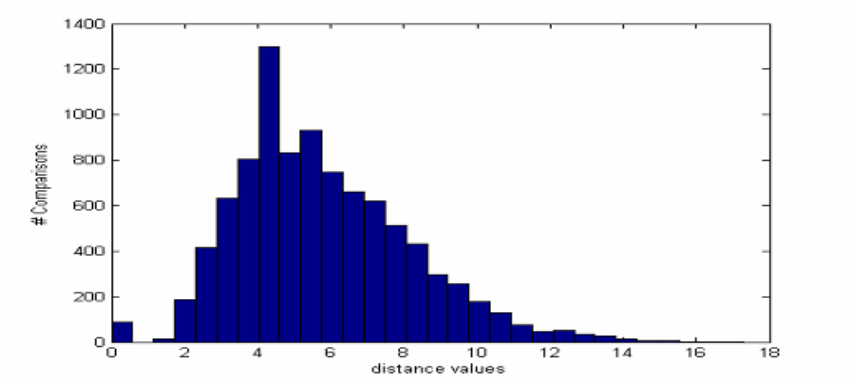


Figure 5: Distance distribution for r=10 and α=5

4. Results

We present here the results obtained when applying SAX (using the parameters discussed in the previous section) to the 137 largest clusters obtained from our dataset. Out of those 137 clusters, we found 89 pairs of similar time series (a cluster might appear in several pairs). Before discussing those similarities in detail, we first demonstrate that this high number of observed similarities is neither an artifact of SAX nor of our clustering algorithm.

4.1. Discussion

Before analyzing the similarities, we first need to be sure that SAX by itself did not lead to any false positives. This is possible if there is a non-negligible probability that two random and independent time series can be declared similar by SAX. This probability depends on the inter symbol distances defined in TAB (equation (1)), as some values in this table (that represents the inter symbol distances) are equal to zero [24]. To obtain a distance of 0 between two random and independent time series of size W, the distance between each of their W elements must be null. As there is a total of 25 elements in TAB and 13 of them are equal to zero the probability P that 2 strings out of K random and independent strings, are similar is:

$$P = \frac{K(K-1)}{2} \times \left(\frac{13}{25}\right)^{w}.$$

The probability of getting one similarity out of 9316 (=137 x 136 / 2) comparisons is thus smaller than 10^{-13}. Therefore, from a probabilistic point of view, it is quite unlikely that the 89 similarities found with SAX are observed by chance.

One could also argue that the large amount of observed similarities results from our clustering algorithm that could have erroneously split the traces due to a single tool into two different clusters. To verify the absence of such a bias, we have computed the distribution of the number of differing parameters in the clustering algorithm of all the 89 pairs of similar time series. We obtained that 60 pairs differ by more than 4 parameters while only 10 differ by a single parameter value. This result indicates that the similarities we observe cannot be simply explained by the fact that our clustering algorithm is too conservative.

4.2. Observed Similarities

We found out that the 89 pairs of similar clusters can be grouped into 3 different sets. The first group, with 28 pairs, does not contain multi-headed stealthy tools. This group results from the fact that our honeypot platform offers 3 different operating systems and that attack tools are attacking differently specific operating systems, leading to different attack fingerprints (from our clustering algorithm point of view) on each machine. As a consequence, the similar pairs of this set will share most of their source addresses.

The two other sets of similar pairs of clusters are much more interesting and deliver the promised results. In the first one, the source addresses in similar pairs of clusters differ, but a strong correlation exists with respect to their network of origin. In the last set, the time signatures are the only elements that link pairs of traces of attacks together.

4.2.1 Common IP Addresses

Looking at the source IP addresses of the 89 pairs of similar clusters, we realized that those pairs seemed either to share most of their addresses or no addresses at all. To further investigate this issue, we computed the ratio of the number of common sources for each of the 89 pairs of clusters. We found out that 28 clusters shared between 85% and 100% of their addresses, while this ratio was almost 0 for all the other pairs of clusters. Further investigation of those 28 clusters revealed that the reason why those tools have been declared dissimilar was because the sequence of ports targeted by a given attacking machine (IP address) on different machines in the platform was different. However, while the sequences of ports differ from one targeted machine to another, one port sequence is always a prefix of the other. Let PS_a and PS_b be the ports sequences associated to a pair of clusters (C_a, C_b). For all 28 pairs in this first group, we find that: $PS_a = (PS_b, *)$ or $PS_b = (PS_a, *)$. Such a behavior is a characteristic of sophisticated tools that always scan the same sequence of ports on a machine, but stop scanning if ever one of the ports is closed. Figure 1a) shows two such clusters. The first one represents attacks targeting ports sequence {135} and the other one ports sequence {135, 4444}. They correspond to MBlaster. If port 135 is found open by the worm, it further scans port 4444 to check if a remote shell is also open [8,23,33]. This behavior leads to two distinct clusters for our clustering algorithm: one for the attacks on port {135} for the virtual Linux machine and one for the attacks on ports sequence {135,4444} for the Windows virtual machines where port 135 is open. Overall, we obtained four pairs of similar time series for ports sequences {135} and {135,4444}. They correspond to four distinct variants of MBlaster. We also observed that each variant, i.e. each pair of clusters, has a very different time signature. This can be an easy way to distinguish them.

The use of time analysis is thus a good way to find out this type of tools. It represents a costless alternative to a complex reverse engineering of the code (that first needs to be captured!) that would reveal that the tool stops scanning a machine whenever a port in the pre-defined sequence of scanned ports is closed. However, we cannot term those tools multi-headed tools because there is always a (prefixing)

relationship between the ports sequences observed in the different machines. We are thus not in the case of a tool that uses a different attack each time it targets a new machine.

The fact that they share IPs addresses is an artifact of the attack tools. We could very well have observed the same phenomenon without having common IP addresses if these worms had used another propagation strategy. However, we did not observe this type of behavior for our 137 clusters.

4.2.2 Common Source Characteristics

Out of the 89 initial pairs of similar clusters, 61 pairs remain for which there is no clear relation between (i) the IP addresses of the attacking machines of each pair and (ii) the ports sequences. At this point, we took advantage of the fact that a possible form of propagation of attack tools is to randomly (and not sequentially as in the case of the first 28 pairs above) scan machines within a network. Thus, while we never observe that the very same attacking machine as targeted two different machines of our platform, we should find a relationship between the origin networks of the machines in some of the pairs of similar clusters. To investigate this issue, we first associated, to each source in each cluster, a domain defined as the set of addresses returned by a whois server for this address. Figure 6 presents the observed overlap between the domains of source IP addresses of the 61 pairs of clusters. We have computed the percentages as follows: if Cluster C_a and C_b are respectively composed of IP addresses from domains Dom_a and Dom_b:

$$P(common\ \ domains: C_a\ and\ C_b) = \frac{card(Dom_a \cap Dom_b)}{card(Dom_a) + card(Dom_b) - card(Dom_a \cap Dom_b)}.100$$

From Figure 6, we clearly observe two distinct groups of pairs of clusters. A first group consisting of 8 pairs for which the overlapping value is close to zero and a second group for which the overlapping values are (mostly) above 20%. We defer the study of the specific group of 8 pairs to the next section and concentrates here on the group of 53 pairs for which there is apparently a non-negligible domain relation among the sources.

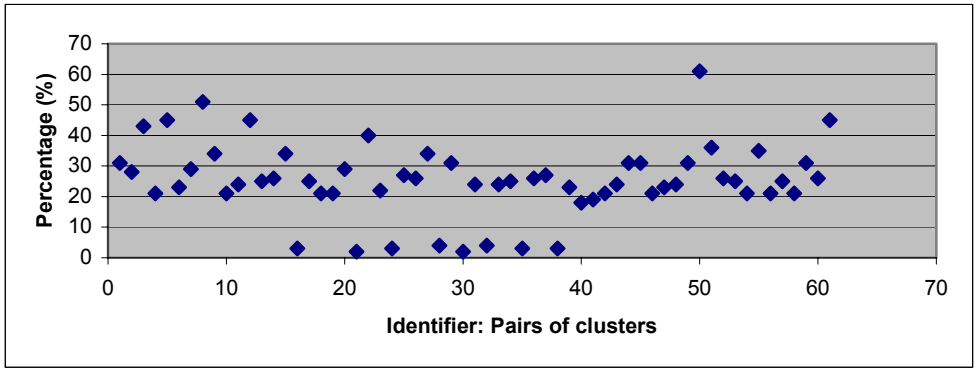


Figure 6: Percentage of overlapping domains between pairs of clusters

In this group of 53 pairs, we identified a variant of the worm Nachi, also called Welchia [2,34] that exploits one of the following vulnerabilities:

- DCOM RPC vulnerability described in MS03-026 bulletin
- WebDav vulnerability described in MS03-007 bulletin
- Workstation Service vulnerability described in MS03-049 bulletin

Welchia is an example of multi-headed tools. To infect other machines, it randomly chooses an IP address and then attacks it either against port 135 or port 445, but not both (it is thus a real multi-headed tool). From our platform viewpoint, traces left by machines infected by Welchia will look very different. They will thus be stored in two different clusters, one for the attacks against port 135 while the other contains attacks against port 445. It is quite unlikely that we will find the very same source in both clusters

but large networks containing many machines compromised by this worm have good chances to appear in both clusters. This is exactly what happens, as the overlapping ratio (pair 18 in Figure 6) for Welchia is 21%.

Another example of such multi-headed tools is Spybot.FCD [6,35]. This tool tries to exploit Windows vulnerabilities either on port 135, 445 or 443. In the case of Spybot.FCD, we have thus observed three similar clusters.

Welchia, Spybot.FCD or W32.Kobot.A are examples of multi-headed stealthy tools that have been studied and analyzed. Many more remain to be identified. Our time signature analysis provides a simple and efficient way to reveal their existence. It should provide valuable input to other research teams interested in studying specific attack tools and/or in reverse engineering them [1,7,26,27,36,37,40].

As N.C. Weaver reports in [41] about *Warhol worms*, many worms are using random scanning in order to detect new targets, but it can happen that some virulent worms start by scanning local subnets and logically adjacent networks. This is a very efficient way to speed up the propagation since vulnerable machines are often clustered together and this proves to be an excellent way to wreak havoc in internal networks. Code Red II's increased virulence was mostly caused by such a subnet scanning routine. Along the same line, Zhou et Al. discuss in [42] the potential impact of *routing worms* that make use of IANA Class A allocations combined with the information of BGP routing prefixes to obtain allocated and active IP ranges. These propagation activities do not appear clearly if we look at the common domains provided by *whois* requests, as presented in Figure 8 because the domains returned by a whois server can be of very different sizes. Thus, we performed a similar analysis, but looking at common /8, /16 and /24 networks for the pairs of clusters. /16 and /24 comparisons did not show any striking pattern. The results are much more interesting for the /8 network comparison, as depicted in Figure 7. Similarly to Figure 6, we still observe (the same) 8 pairs of clusters that do not share many /8 networks. Most of the remaining 53 other pairs have more than 40% of /8 networks in common. We also note that 13 pairs have between 80% and 100% of common /8 networks. Those networks differ, though, from one pair to another. There are 256 blocks of addresses of /8 type. However, the clusters under study share a very few number of them. For the 13 pairs with similarity level between 80 and 100%, the number of common /8 networks does not exceed 15. At this point, two different scenarios are possible. Either all machines infected by such tools are confined within the few common networks we observe or they are spread all over the world but their propagation strategy is such that we see more attacks from some networks than from others. The distributed platform we are currently deploying should help us to address this issue in the future.

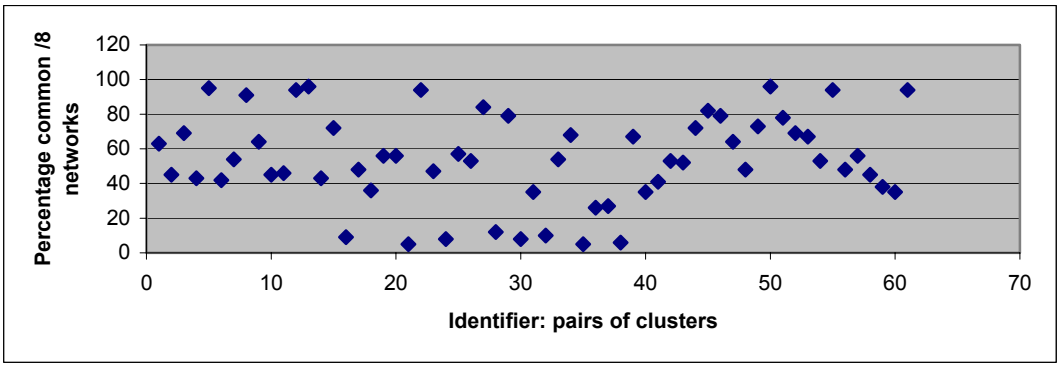


Figure 7: Percentage of common /8 networks between pairs of clusters

4.2.3 The real stealthy cases

In this section, we focus on the last 8 pairs of clusters that exhibit no significant similarity in their source IP addresses and port sequences. To better characterize those pairs, we collected additional data about the infected machines, namely a passive OS fingerprinting (using p0f) and domain name resolution outputs.

As for the passive OS fingerprinting, we first observed that almost 100% of the machines in those 8 pairs are Windows machines. We further estimated the number of personal machines by looking at specific strings in the domain names. If the domain name includes some specific strings such as '%dial%',

'%dsl%' or '%cable%', there is a good probability that those machines are personal computers. While over the whole set of clusters identified by our clustering algorithm, 34% of the machines appeared to be personal computers, this figure jumps to 63% for the case of 8 pairs of clusters under consideration. A preliminary conclusion is that the attacking machines in those 8 pairs of clusters are mostly windows machines from personal users.

We also investigated the port numbers targeted by those 8 pairs of clusters. Attacked ports are 21, 25, 80, 111, 135, 137, 139, 445, 554 and 27374. We however found no obvious relation among them.

To try to gain further insights on those tools, we picked one of those 8 pairs that consists of attacks targeting port 27374 (a port left open by some Trojans) while the other cluster targets port 21 (FTP). We refer to those two clusters as C_a and C_b respectively. Table 2a) gives the top 10 origin countries of those attacks (based on Maxmind) and Table 2b) provides the top 5 origin domains. As can be seen, there is no simple relation between those two types of information. For instance, China does not account for C_b activities while it represents one fourth of C_a activities. In addition, the weight of .com domains is totally different from one cluster to the other.

C_a	C_b
CN: 24%	US: 47%
KR: 17%	KR: 11%
TW: 14%	FR: 10%
US: 10%	CA: 7%
DE: 7%	DE: 6%

C_a		C_b	
.net	31%	.net	32%
.com	4%	.com	40%
.it	3%	.fr	9%
others	28%	others	1%
undetermined	34%	undetermined	18%

Table 2: a) Top 5 sources locations b) top domains for clusters C_a and C_b

Those 8 similarities correspond to multi-headed stealthy tools that have not, to the best of our knowledge, been identified so far. As a matter of fact, without having access to the code of the tool itself, our method appears to be the only way to pinpoint their existence. It is also very interesting to note that same pairs of similarities exist on other honeypot platforms that we monitor. It is, therefore, not something specific to that single platform. A precise characterization of those malware remains, at this stage, an open issue and we hope that this result will encourage researchers to capture and study them.

5. Conclusion

In this paper, we have highlighted the existence of so called multi-headed stealthy tools based on the similarity between time signatures of attack tools. Since they present different attack fingerprints, they are difficult to identify except by reverse engineering their code. However, we have shown that their identification is feasible by following two distinct steps: first, we group attacks with a common fingerprint on the honeypot platform into the same cluster, and then we compare the evolution of these clusters over time to find out similarities. By applying this method to a large set of real world attack traces, we have highlighted the relevance of the method. A handful of multi-headed stealthy tools have also been discovered.

Future work will follow two major directions. One is to generalize the technique on all the honeypot platforms we have deployed, as soon as they provide a suitable amount of data, in order to carry an in-depth cross-platform comparison. A second direction will consist in reverse engineering a few attack tools, for which captured packets contain the code. Our dataset as well as the graphical interface to test our tool (through a java applet) will be made publicly available.

6. References

[1] H. Andersson, T. Britton, *Stochastic Epidemic Models and Their Statistical Analysis*, Springer-Verlag, NewYork, 2000.

[2] Antivirus World. Welchia/Nachi page at: http://www.antivirusworld.com/articles/virus/welchia.php

[3] B.L. Bowerman, R.T. O'Connell. *Time Series and Forecasting*, Duxbury Press, North Scituate, Massachusetts, 1979.

[4] CAIDA Project – The UCSD Network Telescope home page: http://www.caida.org/

[5] CAIDA Project, "The Spread of the Witty Worm", available at: http://www.caida.org/analysis/security/witty/

[6] Cauzomb. "sysmsvc.exe". Article at: http://www.iamnotageek.com/history/topic.php/77580-1.html

[7] CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL", 19 July 2001, *CERT/CC*; information available on line at http://www.cert.org/advisories/CA-2001-19.html

[8] CERT® Advisory CA-2003-20 "W32/Blaster worm", August 2003, *CERT/CC*; information available on line at http://www.cert.org/advisories/CA-2003-20.html

[9] R.E. Challis, R.I.Kitney. "Biomedical signal processing (in four parts). Part 1 Time-domain methods." *Medical & Biological Engineering & Computing*, 28, 509-524.

[10] K. Chan, A. W. Fu. "Efficient Time Series Matching by Wavelets". In Proc. of *the 15th IEEE Int'l Conference on Data Engineering*, pp. 126-133, Sydney, Australia, 1999.

[11] Z. Chen, L. Gao, K. Kwiat. "Modeling the Spread of Active Worms". In Proc. of *the IEEE INFOCOM*, 2003.

[12] ANONYMIZED FOR BLIND REVIEW

[13] ANONYMIZED FOR BLIND REVIEW

[14] DShield Distributed Intrusion Detection System home page: http://www.dshield.org

[15] F-Secure Corporation. "Deloder Worm Analysis". Available at: http://www.f-secure.com

[16] P. Indyk, N. Koudas, S. Muthukrishnan. "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches". In Proc. of *the 26th VLDB Conference*, Egypt, 2000.

[17] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra. "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases". Published in *Knowledge and Information Systems KAIS,* May 2000.

[18] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra. "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases". In Proc. of *the ACM SIGMOD Conference on Management of Data*, Santa-Barbara, CA. May 2001.

[19] E. Keogh, S. Kasetty. "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration". In Proc. of *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDDM'02*. Alberta, Canada, July 2002, pp. 102-111.

[20] E. Keogh, S. Lonardi, B. Chiu. "Finding Surprising Patterns in a Time Series Database in Linear Time and Space". In Proc. of *the ACM SIGKDD'02*, Alberta, Canada, July 2002.

[21] B. Krishnamurthy, S. Sen, Y. Zhang. "Sketch-Based Detection: Methods, Evaluation, and Applications". In Proc. of *the Internet Measurement Conference IMC'03*, Florida, USA, Oct. 2003.

[22] R. J. Larsen, M.L. Marx. *"An Introduction to Mathematical Statistics and Its Applications"*. Prentice Hall, Englewoods, Cliffs, N.J. 2nd Edition, 1986.

[23] R. Lemos. "MSBlast epidemic far larger than believed". *CNET news.com*. April 2004. Available at: http://www.imri.com/Reference/MSBlastEpidemicCNet.pdf

[24] J. Lin, E. Keogh, S. Lonardi, B. Chiu. "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms". In Proc. of *the 8th ACM SIGMOD Worshop on Research Issues in Data Mining and Knowledge Discovery DMKD'03*, California, US, June 2003.

[25] McAFee Security Antivirus. "Virus profile: W32/Deloder Worm". Available at: http://us.mcafee.com/virusInfo/

[26] D. Moore. "Code-Red: a case study on the spread and victims of an internet worm". Available at: http://wwww.icir.org/vern/imw-2002/imw2002-papers/209.ps.gz, 2002.

[27] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford et N. Weaver, "Inside the Slammer Worm", In Proc. Of *the IEEE Security & Privacy*, July-August 2003, pages. 33-39.

[28] ANONYMIZED FOR BLIND REVIEW

[29] ANONYMIZED FOR BLIND REVIEW

[30] D. Shasha, Y. Zhu. "High Performance Discovery in Time Series". 190 pages, Monographs in Computer Science, Springer, 2004.

[31] E. Spafford. "An Analysis of the Internet Worm". In Proc. of *the European Software Engineering Conference*, Lecture Notes in Computer Science, Vol. LNCS 387, pp.446-468, Springler-Verlag, 1989.

[32] S. Staniford, V. Paxson, N. Weaver. "How to Own the Internet in Your Spare Time". In Proc. of *the 11th USENIX Security Symposium*, pp. 149-167, 2002.

[33] Symantec Security Response. "W32.Blaster.Worm". Descriptive page available at: http://securityresponse.symantec.com/ avcenter/venc/data/w32.blaster.c.worm.html
[34] Symantec Security Response. "Welchia/Nachi". Descriptive page available at: http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.c.worm.html
[35] Symantec Security Response. "W32.Spybot.FCD". Descriptive page available at: http://securityresponse.symantec.com/ avcenter/venc/data/w32.spybot.fcd.html
[36] A. Wagner, T. Dubendorfer, B. Plattner, R. Hiestand. "Experiences with Worm Propagation Simulations". In Proc. of *the 2003 ACM Workshop on Rapid Malcode WORM'03*, pp.34-41, Oct. 2003.
[37] J. Xu, S. Chen, Z. Kalbarczyk and R. K. Iyer, "Code Red Worm Propagation Modeling and Analysis", In Proc. of *the ACM Conference on Computer and Communication Security*, CCS'02, Washington DC, USA, November 2002, pages 138-147.
[38] C.C. Zou, W. Gong, D. Towsley. "Code Red Worm Propagation Modeling and Analysis". In Proc. of *the ACM Conference on Computer and Communications Security CCS'02*, Washington DC, USA, Nov. 2002.
[39] E.Cooke, M. Bailey, D. Watson, F. Jahanian, J. Nazario. "The Internet Motion Sensor: a distributed blackhole monitoring system". Technical Report Version. In Proc. of *the 12th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2005.
[40] P. Szor. "Virus Research and Defense". Symantec Press. 2005.
[41] N.C. Weaver. "Warhol Worms: The Potential for Very Fst Internet Plagues". Report available at: http://www.cs.berkeley.edu/~nweaver/warhol.html
[42] Cliff C. Zou, Don Towsley, Weibo Gong, and Songlin Cai. "Routing Worm: A Fast, Selective Attack Worm based on IP Address Information," Umass ECE Technical Report TR-03-CSE-06, November 2003 (CiteSeer citations: 4).
[43] The Team Cymru Darknet Project home page: http://www.cymru.com/Darknet/
[44] M. Zhou, S.D. Lang, "A Frequency-Based Approach to Intrusion Detection". In Proc. of the *Workshop on Network Security Threats and Countermeasures*, July 2003.