The Quest for Multi-headed Worms

Van-Hau Pham, Marc Dacier, Guillaume Urvoy-Keller, and Taoufik En-Najjary

Institut Eurecom, Sophia–Antipolis, France
pham,dacier,urvoy,ennajjar@eurecom.fr

Abstract. In [6], Pouget et al. have conjectured the existence of so-called multi-headed worms and found a couple of them on attack traces collected on a single honeypot. These worms take advantage of several distinct attack techniques to propagate but they use only one of them against a given target. From a victim's viewpoint, they are therefore indistinguishable from the other classical worms that always propagate using the same attack vector or same sequence of attack vectors. This paper aims at confirming the existence of these worms by studying a very large dataset. The validation process led to three important contributions. First, we establish the existence and assess the importance of three distinct classes of attacks seen in the wild. Second, we propose a new method to correlate attack traces time series and apply it to search for multi-headed worms. Third, we offer and discuss results of the analysis of 15 months of data gathered over 28 different platforms located all over the world.

1 Introduction

The concept of worm, as a programming paradigm, has been introduced more than 25 years ago [8] and has been used to propagate malicious code on a large scale as early as September 1988 with the first ADM worm targeting the DNS infrastructure [3] and with the so called Morris worm, also known as the Internet worm, hitting the Internet in November 1988 [9, 2]. However, one had to wait more than ten years to see worms routinely used by hackers and various techniques used to speed up their propagation on the Internet [10]. We refer the interested reader to the taxonomy of worms published in [12]. The authors provide several examples of worms, classifying them according to various viewpoints, namely worm target discovery and selection strategies, worm carrier mechanisms, worm activation, possible payloads, and plausible attackers who would employ a worm. As indicated in [12], worm authors are not so much interested anymore in gaining faith for having created the fastest worm or the worm having compromised the largest amount of machines. Instead, worm spreading is now seen as a preliminary phase to conduct other fraudulent activities to gain money using various techniques (spam relays, extortion with DDoS threats, pay-per-click fraud, etc.). Therefore, worms are now designed to make their propagations as stealthy as possible.

Multi-headed worms, identified by Pouget et al. in [6], belong to a new class of worms designed with stealthiness in mind. These sophisticated programs can break into target machines using several different techniques. This, by itself, is not new. The Morris worm [9], in 1988, already had this feature. It was propagating using attacks against three different services: rshd, fingerd and sendmail. The Morris worm, after

having selected a target, was trying all three attacks, one after another, interrupting the process only in the case of a successful intrusion. Several other worms have, since then, used the same strategy. They all are fairly easy to identify thanks to the known sets (or sequences) of attacks they try against their targets. Multi-headed worms, as defined in [6], use a very different strategy: they probe each target with only one of the attacks they are capable of. This strategy decreases their chance of success but increases their stealthiness. Indeed, there will be no trace left anywhere highlighting the fact that a new worm has been created combining attacks X, Y and Z as they will never be tried together by a given attacker against a given attackee.

In [6], the authors had used traces left on a simple low interaction honeypot to highlight the existence of a couple of such multi-headed worms propagating in the Internet. At that time, only one of them, Nachia, had been acknowledged by intrusion detection and antivirus vendors. This seminal work had been carried out on a single platform and, therefore, was not able to assess the seriousness of the threats posed by this new class of worms.

In this paper, we carry out a systematic identification of multi-headed worms in attack traces collected thanks to 28 distinct low interaction honeypot platforms, located in 15 different countries, over a 15 month period. In order to perform this experiment, we had to design a different method than the one originally proposed in [6] because of algorithmic complexity issues. The application of this validation process led to three important contributions: i) we establish the existence and assess the importance of three distinct classes of attacks seen in the wild; ii) we offer a new generic method to correlate attack traces time series that could be applied to other kinds of datasets; iii) we offer and discuss results of the analysis of 15 months of data gathered over 28 different platforms located all over the world.

The paper is structured as follows. Section 2 reviews the state of the art and describes the two main reasons why the solution provided in [6] does not scale. Section 3 presents the three distinct steps of the new method we propose: (i) Identification and selection of attack classes (ii) Identification of correlated platforms (iii) Root causes identification. Section 4 provides a summarized description and discussion of the most interesting results obtained. Section 5 concludes the paper.

2 Problem statement

In this section, we describe the original solution provided in [6] for the identification of multi-headed worms and explain the two main reasons why this solution does not scale. For the sake of completeness, we first start by briefly describing the data collection environment considered in that work as well as some definitions of terms used throughout this paper.

2.1 The Leurré.com environment

The Leurré.com environment is a distributed setup of low interaction honeypots. As of now, there are approximately 50 different partners that host a so-called *platform*. All platforms are configured exactly the same way. Each platform emulates, thanks to

honeyd [7], three virtual machines: a Windows 98 machine, a Windows NT Server, and a Linux RedHat 7.3. These platforms are located in 30 different countries covering the five continents. They are hosted by different types of institutions (academic, industrial, government, defense, SME, etc.). Most platforms have been active for more than 24 months; the oldest one has been running since January 2003.

Each platform captures tcpdump traces of all packets sent to and from it. These files are uploaded, on a daily basis, in a centralized Oracle database accessible to all partners to carry out various kinds of analysis. The entity relationship diagram of the database is fairly complex and its description lies outside the scope of this paper. However, a few key concepts must be precisely defined in order to avoid any misunderstandings.

- **Platform**: A physical machine, hosting three virtual machines, connected directly to the Internet and collecting tcpdump traces in the context of the Leurré.com environment.
- **Source**: A source corresponds to an IP address that has sent at least one packet to, at least, one platform. It is important to understand that a given IP address can correspond to several distinct sources. Indeed, a given IP remains associated to a given source as long as there is no more than 25 hours between 2 packets received from that IP. After such a delay, a new identifier will be assigned to the IP. By grouping packets by sources instead of by IPs, we minimize the risk of gathering packets sent by distinct physical machines that have been assigned the same IP dynamically after 25 hours.
- **Ports Sequence**: A ports sequence is a time ordered sequence of ports (without duplicates) a source has contacted on a given virtual machine. For example, if an attacker sends the following packets: icmp, 135 TCP, 135 TCP, 139 TCP to a given virtual machine, the associated ports sequence will be represented by the string $ICMP|135T|139T$. Each source can have, at most, three distinct ports sequences associated to it, per platform. As of now, we have observed around 40,000 distinct unique ports sequences on all Leurré.com platforms.
- **Cluster**: A cluster is made of a group of sources that have left highly similar traces on all platforms they have been seen on. Clusters have been precisely defined in [5]. They aim at grouping together attackers that are likely launching attacks with the very same attack tool. Traces present in a given cluster have 7 features in common, one of them being to have targeted the same ports sequence as defined here above. As of now, we have observed more than 154,900 different clusters.
- **Cluster time series**: A Cluster time series represents the amount of sources, on a daily basis, associated to a given cluster on a given platform. In other words, there are, for a given cluster, as many cluster time series as platforms.
- **Global Cluster time series**: A global cluster time series represents the sum of all cluster time series associated to a given cluster. In other words, there is a single global cluster time series associated to a given cluster.
- **Platform time series**: A platform time series represents the sum of all cluster time series associated to a given platform. In other words, there is a single platform time series associated to a given platform.

2.2 Seminal work on the identification of multi-headed worms

Pouget et al. have proposed in [6] a method to discover multi-headed attack tools. In that paper, the authors explain that sources compromised by a multi-headed worm leave, by definition, distinct traces on the honeypots depending on which attack they choose to launch against them. As a result, the sources will be classified into as many different clusters as there are different possible attacks for the worm. However, the various cluster time series associated to a given multi-headed worm should evolve over time in a similar way as they all are a function of the total amount of machines compromised by that multi-headed worm at any point in time. Therefore, by identifying cluster time series that are very similar to each other appears to be a simple yet efficient way to identify multi-headed worms. In [6], the authors have used the SAX technique [4] to calculate the distance between all pairs of cluster time series data.

The authors have shown, by means of data extracted from a single platform, the existence of a couple of interesting multi-headed worms. Having a much larger dataset at our disposal, we were interested in verifying their results on a worldwide scale. Unfortunately, we found out that the detection method in [6] does not scale to that level for two main reasons. First, the most straightforward way to generalize the approach to data collected on several platforms, instead of one, is to measure the distance between different global cluster time series. Experience shows, as discussed below, that this approach does not work when a large number of platforms located in many different places in the world are considered. The reason lies in the fact that worms do not spread in an uniform way across the IP space. Therefore, we must measure distances between cluster time series observed on distinct platforms instead of global cluster time series. Second, the authors in [6] considered a fixed time window of 1 year to assess the distance between time series. This approach works for some extreme cases but, as we demonstrate hereafter, is also likely to miss many interesting phenomena, the existence of which is only visible during a couple of weeks. Therefore, their fixed time window must be replaced by a sliding window.

Measuring distance between cluster time series on many platforms by means of a sliding window leads to an algorithmic complexity problem. In the next three sections, we provide examples of the two problems described above and offer a formal complexity analysis of these issues. Section 3 proposes a new solution to address these identified drawbacks of the original method.

Fixed time window vs sliding time window The top plot in Figure 1 presents the global cluster time series for two distinct clusters, over a period of more than 450 days. The first (resp. second) one represents sources belonging to cluster number 15715 (resp. 60231) only. The corresponding ports sequence of cluster 15715 (resp. 60231) is 1433TCP (resp. 5900TCP). The SAX distance, computed as described in [6], i.e., over the whole observation period, would lead us to consider that these two cluster time series are not correlated. However, when looking at the bottom plot in Figure 1, it is clear that these curves are highly correlated between day 60 and 90. The reason why SAX gives a low similitude is mostly due to the activities happening before day 60 as well as after day 200. It can well be that the existence of the multi-headed worm can only be detected during a limited period of time. This is especially true for multi-headed

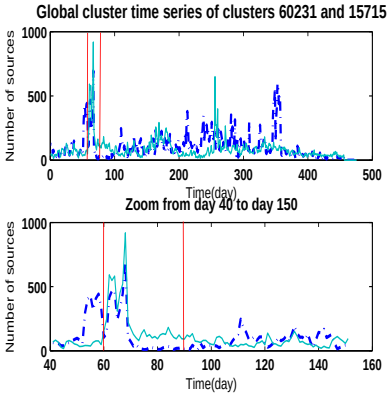

Fig. 1. Fixed time window vs sliding time window

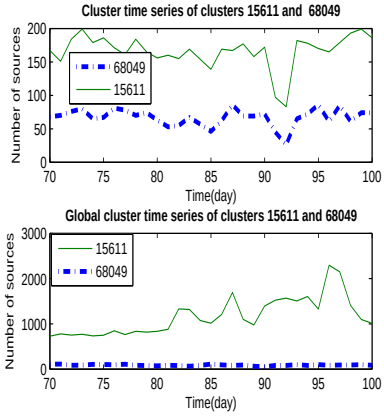

Fig. 2. Global cluster time series vs. cluster time series

worms that are using attacks that were already frequently observed when the multi-headed worm got launched. As a consequence, one cannot simply rely on the usage of a large fixed time window, as proposed in [6] to detect those worms. Using a sliding time window is obviously the way to go in order to address this issue.

Global cluster time series vs. cluster time series The top plot in Figure 2, shows two distinct cluster time series on platform 18 over a period of 30 days. The first (resp. second) one represents the evolution of cluster number 15611(resp. 68049). The bottom plot in Figure 2 represents the corresponding global cluster time series (over all platforms) over the same period for these two clusters. These figures highlight the fact that, on platform 18, the two cluster time series are highly correlated between day 70 until day 100 whereas the corresponding global cluster time series are not. This can be explained by the fact that a multi-headed worm is not necessarily observed everywhere in the world. If the multi-headed worm is reusing attack vectors that are frequently observed elsewhere, its existence will remain hidden if we use global cluster time series instead of carrying out the analysis on each platform independently.

Working with global cluster time series is thus not an option. One of the contribution of this paper will be to demonstrate that, unlike global clusters, platform time series carry enough information so as to uncover correlations among cluster time series. This is an important finding as it enables us to reduce the computational cost of the correlation search phase as shown in the next section.

2.3 Complexity Analysis

From the previous two examples, it comes out that, in order to deal with these two issues, we should apply the method proposed in [6] between all cluster time series, for every platform, over a sliding time window. Intuitively, this leads to a very large amount of computations that we detail hereafter.

Let $S = \{S_i\}, i = 1..N$, be the set of platforms and $A = \{cl_i\}, i = 1..K$ be the set of distinct clusters observed on all the platforms during a period of T consecutive days. Our objective is to identify all clusters that targeted a subset $S' \subset S$ of platforms over a period of $T' \leq T$ consecutive days in a similar way. By similar, we mean that the selected cluster time series on any two platforms of S' are highly correlated.

To do so, we compute the correlation over a sliding window of size L. For a total of M time series ($M \leq K \times N$ as not all clusters are observed on all platforms), the total number of correlations to be computed is given by:

$$C_1 = \frac{M \times (M-1)}{2}(T-L)$$
$$C_1 = O(M^2T)$$

We postpone until Section 4 the details of the numerical results obtained from the experiments but, for now, the reader should be aware that M amounts to more than 59,000 in the 15 months period considered. Clearly the simplistic generalization of the method described in [6] is too expensive.

Our solution to reduce the complexity is twofold. First, we find an automated way to select a subset M' from M such that $M' \ll M$. The reduction technique is presented in Section 3.1. Experimentally, we found out that M' can be an order of magnitude smaller than M. After such selection, the complexity comes down to $C_2 = O(M'^2T)$ and $C_2 \ll C_1$. However, the cost C_2 remains prohibitive and this leads to the second step of our method where we compute filtered platform time series corresponding to the sum of activities corresponding to these M' time series per platform. We then look for similitude between these filtered platform time series instead of between cluster time series. The cost to pay for finding similar platform time series comes down to

$$C_3 = \frac{N \times (N-1)}{2}(T-L)$$
$$C_3 = O(N^2T)$$

This leads us to the identification of a certain amount P (with $P \ll T-L$) of periods in which we have a group of G_i (with $i = 1..P$ and for $\forall i |G_i| \ll N$) correlated platform time series. For each period, we have to find the cluster responsible for the identified similarity. In other words, for each period, we must compare the M' cluster time series with, at maximum, N filtered platform time series. This leads to the identification of the root causes of the similarity on each platform. If we define $G = max G_i | i = 1..P$, an upper bound of the cost of this operation can be given by $C_4 = P \times G \times M'$. Thus, the total cost of this method is equal to $C_5 = C_3 + C_4$ and we have $C_5 = O(N^2T + PGM')$. In the general case, nothing ensures, *a priori*, that $C_5 \ll C_2 \ll C_1$ but, as we expect the values of N, P and G to be very small compared to M and M', this justifies the choice of this solution. Experimental results presented in Section 4 validate this choice.

3 Methodology

We detail in this section the three steps of our methodology we have eluded to in the previous Section:

1. All attack traces can be grouped into three distinct families. Only one of them is likely to contain traces due to multi-headed worms. Therefore, the method starts by selecting in our dataset those traces that belong to the sole interesting family.
2. Our platforms observe a limited number of hits per day. If at some points in time two platforms become the target of a multi-headed worm, we make the assumption that this will significantly impact the overall platform time series on that period. Therefore, the method identifies groups of platform time series strongly correlated over different periods of time and identifies the root causes for those similitudes. Similarly to the approach followed in [6], if a similitude is caused by many attack tools, we believe this reveals the existence of a multi-headed worm. Obviously, if the intensity of the attack is not high enough that it impacts the platform time series of at least two platforms, our method will miss it. The validity of the method is further discussed while presenting the experimental results in Section 4.
3. We search for the root causes, i.e. the clusters that are responsible, if any, for the similar shape of the filtered platform time series in each group. Once we have found them, we verify that they did not also existed on other platforms than the ones we had in the group under study. This can happen if the influence of these clusters on the other filtered platform time series was not strong enough to include them in the group of similar platforms.

3.1 Construction of filtered platform time series

As explained before, the first step of our technique aims at reducing the number of cluster time series we need to focus on. Our method to reduce the size of the problem is based on our experience with attack traces collected in the Leurré.com project. We have observed that cluster time series can be categorized into 3 distinct families[1]:

1. **Peaked family**: Time series in this family exhibit a significant peak of values during a very small period of one or two days and almost no activity otherwise. In most cases, the corresponding cluster is observed on a single platform only. We leave for future work a more in depth study of this specific type of phenomena and we thus exclude those time series when building platform time series.
2. **Stable family**: Time series in this family have a roughly constant behavior during the whole observation period. As we make assumption that correlated clusters due to multi-headed worms exhibits time series having similar noticeable variations over time, stable clusters are meaningless in the context of this analysis. We can simply remove them from our dataset. Note that removing the stable ones has little impact on the shape of the platform time series. However, as a very large number of time series falls into the stable family, removing them from our initial set dramatically reduces the computational cost.
3. **Strongly varying family**: Time series in this family are characterized by wide amplitude variations over long periods of time. Our objective is to uncover phenomena

[1] There is no reason to believe that the findings described hereafter are not also applicable to datasets collected by other projects. If that were the case, it would certainly be worth investigating the reasons why.

that involve several cluster time series over periods of time larger than a few days, we restrict our attention to those time series in the remaining of this paper.

We proceed as follows to classify each cluster time series into one of the three families introduced above. We first compute the standard deviation of the time series over the whole observation period. If it is smaller than a threshold δ, then we flag the time series as belonging to the stable family. Otherwise, we filter out the outlier values from the time series. Outliers are defined as the two greatest and smallest values of the time series. Then we compute the standard deviation of newly obtained time series. If the standard deviation is now smaller than δ, we declare the time series as being a peaked time series. Otherwise, we declare the time series as belonging to the strongly varying family and we thus keep it in our set of cluster time series. In the above procedure, we used $\delta = 2$, which is intended to be a conservative value, based on the visual inspection of a lot of cluster time series.

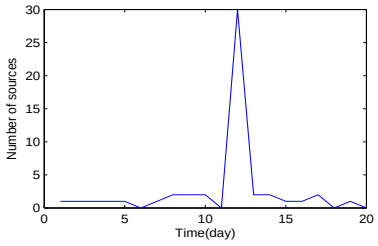

Fig. 3. Example of the peaked family time series

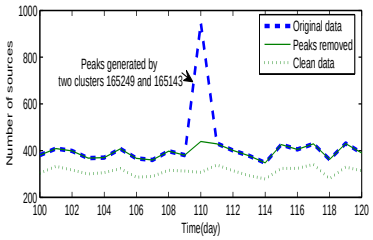

Fig. 4. Data Pre-processing

Figure 3 illustrates the algorithm for a cluster time series that spans over 20 days. The standard deviation of the time series is 6.51. Since it is greater than 2, our algorithm can not declare this time series as a stable one upfront. We next filter the extreme values from this time series , which for the case of Figure 3 boils down to cutting the peak on day 12. The resulting time series is obviously smoother than the initial one and its standard deviation is 0.46, which is smaller than the threshold 2. Hence, our algorithm eventually flags the time series of Figure 3 as belonging to the peaked family.

The cost of the above filtering process comes on top of the complexity evaluated in the previous Section but it is very small compared to C_1 since its complexity is linear with respect to the number of clusters and the algorithm involved for each cluster is much cheaper to run than the evaluation of the correlation between two clusters (over sliding windows), as discussed before.

Figure 4 illustrates our pre-processing technique. We plot three platform time series for platform 18. *Original data* is the platform time series obtained using all clusters. It is made of 6162 clusters in this specific case. *Peaks removed* is obtained once the peaked time series have been filtered out. It is made of 6108 clusters as 54 clusters were peaks in this example. *Clean data* is the platform time series data once the peaked and

stable time series have been removed. It is made of only 39 clusters! This highlights the usefulness of the preprocessing phase.

Figure 4 clearly shows that *original data* is quite different from *clean data* due to the two peaks at the same position (110). These peaks (clusters number 165249 and 165143) were created by 510 sources. This attack was neither observed before or after day 80, nor was it observed on any other platform. As we can see, the *peaks removed* and *clean data* time series have a very similar shape. They differ only with respect to their amplitude. However, we remind the reader that the *peaks removed* time series contain 6108 clusters and that only 39 (strongly varying) time series remain in *clean data*.

3.2 Groups of Correlated Filtered Platform Time Series

In this section, we explain how we identify correlated groups, i.e. groups of platforms for which any two filtered platform time series are mutually correlated for a given period of T' days. Obviously, one wants to maximize the number of platforms involved and the duration T' over which each group exists. The proposed algorithm is made of three successive steps described in the following subsections: i) pairwise comparison of filtered platform time series, ii) construction of groups of correlated platforms within a given time period and iii) reorganization of the time periods to maximize them on a group by group basis.

Pairwise correlation of filtered platform time series The first step of our algorithm consists in computing the correlation of any two platform time series using a sliding window of L days. Consider two time series Φ and Ψ. Let $cor(A, B)$ be the coefficient of correlation of two vectors A and B. The correlation vector C of Φ and Ψ is computed as follows:

$$C[k] = cor(\Phi[k, k + L], \Psi[k, k + L]), \quad k = 1, \ldots T - L$$

Φ and Ψ are considered to be correlated in the interval $[t_1, t_2]$ if $C[k]$ is greater than a given threshold for every k value in the interval $[t_1, t_2 - L]$. We use as a measure of correlation the Pearson coefficient of correlation [11].

An important parameter of our procedure is the choice of the threshold to declare that two time series are correlated. Again, we rely on experience, i.e. visual inspection of a lot of cases, to choose our threshold. We end up having a threshold of 0.75. We note that this is a high, and thus safe, value as 0.4 is already considered as a significant correlation value in the statistical literature.

Figure 5 illustrates the first step of our procedure. The platform time series for platforms 2 and 15 are deemed correlated in the interval $[t_1, t_3]$ as their correlation vector is greater than the threshold of 0.75 in the period $[t_1, t_2] = [t_1, t_3 - L]$.

Application of the above procedure to all the pairs of platform time series leads to the identification of a set of correlated pairs of platforms over different periods of time. Figure 6 illustrates the situation at the end of the first phase. It shows that platform time series 4 and 7 (curve $4\&7$) are correlated from day 1 to day 4, platform time series 1 and 8 (curve $8\&1$) are correlated from day 1 to day 6, etc.

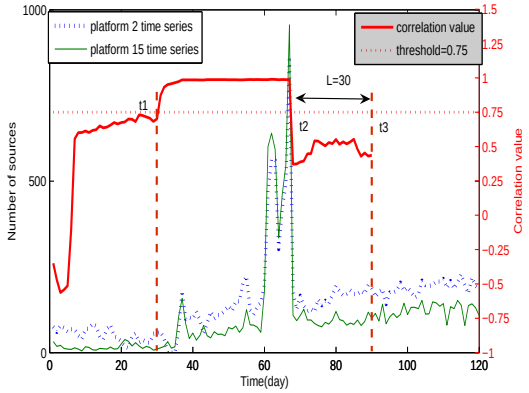

Fig. 5. Example of correlated platform time series

Correlated groups extraction per time interval Based on this first result, our next objective is to divide the time line from 0 to T into a set of time intervals such that the pairs of platforms associated to one interval are correlated over the whole duration of this interval. Within each interval, we want to identify groups of platforms such that all platforms in the group are correlated to all others. The algorithm we use to achieve this task can be summarized as follow:

1. $i = 1, T_{start,i} = 1, T_{end,i} = 1$, L is the sliding window parameter.
2. We define S_i as being the set of pairs of correlated platforms at time $T_{start,i}$.
3. We exclude from S all pairs of correlated platforms that are not correlated until, at least, $T_{start,i} + L$.
4. We define $T_{end,i}$ as being the first end point of the pairwise correlations in S. Interval i is then defined as $[T_{start,i}, T_{end,i}]$; We proceed to the next interval $i \rightarrow i + 1$
5. We define $T_{start,i}$ as being the first start point of a pairwise correlations not yet present in S.
6. If $T_{start,i} \leq T - L$, we reinitialize S to $\emptyset$ and go back to step 2; if not the algorithm terminates.

.

Applying this algorithm to the case described in Figure 6, leads to the identification of the three periods defined in Table 1 when we chose L=3.

Table 1. Periods

$T_1 = [T_{start,1}, T_{end,1}] = [1, 4]$	$S_1 = \{(4, 7), (8, 1), (1, 2), (2, 8)\}$
$T_2 = [T_{start,2}, T_{end,2}] = [3, 6]$	$S_1 = \{(8, 1), (1, 2), (2, 8), (9, 10)\}$
$T_3 = [T_{start,3}, T_{end,3}] = [4, 8]$	$S_1 = \{(5, 1), (1, 2), (2, 5)\}$

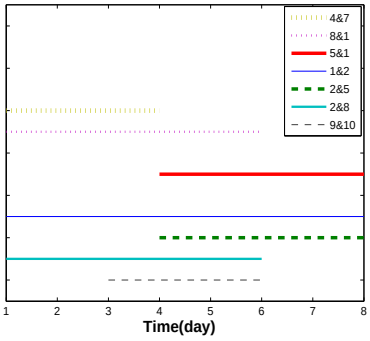

Fig. 6. Correlated pairs of platform time series over time

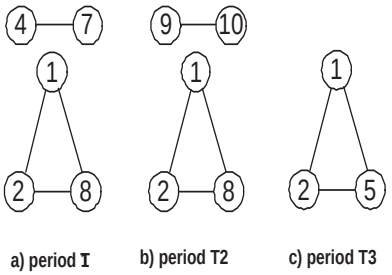

Fig. 7. Correlated groups extraction

Having identified time intervals, we now need to group together all platforms that are correlated with each other. If we use a graph representation of the correlated pairs identified in the previous stage of our algorithm, the problem corresponds to the identification of cliques[2] within the graph. We generate one graph per period. Nodes in a graph represent platform time series and if two platform time series are correlated in that period, their edges are connected. Figure 7 depicts the graphs we obtain for the periods T_1, T_2 and T_3 extracted from Figure 6. The clique extraction problem [1] is an NP-complete one. In our case, this is not an issue as the number of nodes (platforms) per period is very small, typically less than 20.

Reorganization of the time periods From the example given above, it is clear that our algorithm generates overlapping time intervals and that the very same group of correlated platforms can be found in these overlapping periods. For instance, the correlated group consisting of platforms 1,2 and 8 appears in period T_1 and also in period T_2 in Figure 7.

In the last step, we revisit the various groups obtained and, on a group by group basis, merge time intervals whenever the same group is found in two consecutive or overlapping periods. This eventually leads to the following time periods (Table 2) and groups for the preceding example.

Table 2. Groups

$T_1 = [1, 4]$	$G_1 = (4, 7)$
$T_2 = [1, 6]$	$G_2 = (1, 2, 8)$
$T_3 = [3, 6]$	$G_3 = (9, 10)$
$T_4 = [4, 8]$	$G_4 = (1, 2, 5)$

[2] A clique in an undirected graph G is a set of vertices V such that for every two vertices in V, there exists an edge connecting the two

3.3 Root Cause Analysis and Hidden Correlations

The most intuitive explanation behind the existence of correlated groups of platforms is that those platforms are targeted by the same tool, launched from a diverse set of sources in a loosely coordinated way. In that case, the same clusters(s) should be found on each platform of the group as being the root cause of the correlation of the platform time series. We could, therefore, simply search for the root causes on one platform per group. However, as explained in [6], multi-headed worms could hit platform X with cluster 1 and platform Y with cluster 2. Therefore, we take the stance of not assuming a priori that the traces left by a given attack tool are the same on the platforms of a correlated group. We thus look for the root causes behind a correlation independently for each platform in a correlated group. This means that for a period of T' days associated to a correlated group, we look, for each platform, for the set of cluster time series that are correlated with the platform time series. Here too, we use a sliding window as one can imagine that the platform time series are correlated due to two distinct and consecutive, or overlapping phenomena. Section 4.2 shows an example of such a situation found in our dataset.

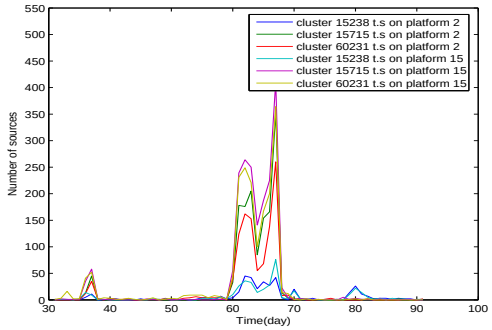

Fig. 8. cluster time series for the clusters uncovered during the root cause analysis for platforms 2 and 15

The correlated group in Figure 5 (between day 31 and day 91) provides an illustration of when the attack tool leaves the same fingerprint on each platform of a correlated group. Indeed, our root cause analysis technique identifies three clusters numbered 15238,15715 and 60231 on both platform 2 and platform 15 as the root causes behind the observed correlation. Figure 8 depicts the cluster time series over the corresponding interval. Table 3 summarizes the correlation values obtained between the different cluster time series for each pair of platforms in the extended group of platforms formed by platforms $\{2,15\}$. As we can see, the correlation coefficients between those clusters are extremely high (greater than 0.85) in this period.

We can observe the highly synchronized behavior of the activities targeting the two platforms.

Hidden Correlations The root cause analysis technique described above enables us to find a set of candidate clusters associated to each correlated group for each platform in that group. However, since we initially identify correlation based on the platform time series, it is possible that a tool targeted x platforms but the effect of the tool is only strongly influencing a subset of $y < x$ platform time series (e.g due to the activity of other local malwares) To uncover all possible hidden correlations, we check if all clusters identified as root causes for a period of T' days for a correlated group are correlated with their siblings on the platforms that are not in the correlated group.

Table 3. Correlation coefficient between clusters

cluster t.s	2 15238	2 15715	2 60231	15 15238	15 15715	15 60231
15238-2	1.0000	0.8521	0.8422	0.8916	0.8631	0.8550
15715-2	0.8521	1.0000	0.9863	0.9248	0.9938	0.9908
60231-2	0.8422	0.9863	1.0000	0.9260	0.9873	0.9873
15238-15	0.8916	0.9248	0.9260	1.0000	0.9154	0.9121
15715-15	0.8631	0.9938	0.9873	0.9154	1.0000	0.9969
60231-15	0.8550	0.9908	0.9873	0.9121	0.9969	1.0000

4 Results

We experimented our algorithms for a period of $T = 467$ days (15 months) and for 28 platforms, whose up time rate was above 90% for the considered period. Those 28 platforms are located in 15 different countries. We applied the methodology described in Section3.2 to a large dataset. It enables us to confirm the existence of multi-headed attack tools, but it also leads to a better understanding of the specific behavior of other interesting classes of attack tools. A summary of these findings is presented hereafter.

4.1 Overview

For our specific dataset, we identified 28 groups involving 111 cluster time series before the hidden correlation identification phase and 130 cluster time series after that. The groups were found in 23 distinct periods, lasting between 30 and 117 days. Figure 9 provides the distribution of number of clusters per correlated group. We observe from Figure 9 that 18 out of 28 correlated groups (ie. 64%) have been associated to more than one root cause. Table 4 lists all the clusters related to at least one correlated group. The first column contains the cluster id. The second column lists the corresponding ports sequences. If a cluster contacts two (resp. three) machines it will have two (resp. three) ports sequences separated by a comma. The last column indicates the number of groups that the cluster is involved in. Figure 10 shows the distribution of the size of correlated groups. We observe from Figure 10 that most of the groups have a small size: 90% of the groups have less than 7 platforms. This observation relates to the fact that malware

attack processes are in general not uniform over the IP address space. The observed phenomena appear to be localized. This is confirmed by Figure 11 which shows that most phenomena target a single /8 network. However, we observe that 21 out of the 28 platforms are involved in at least one correlated group showing that these phenomena are visible all over the world. These 21 platforms are located in 13 (resp 12) out of 15 countries (16 /8 networks).

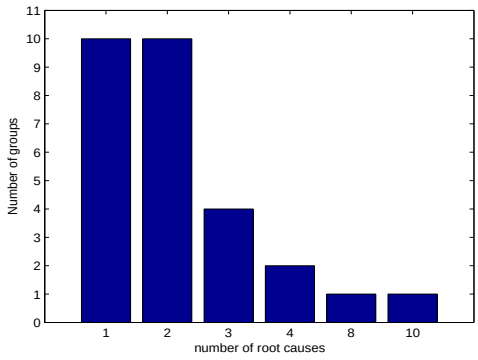

Fig. 9. Distribution of number of clusters

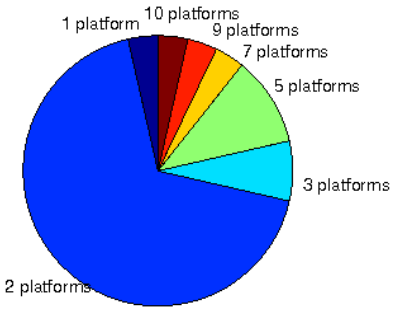

Fig. 10. Platform size distribution

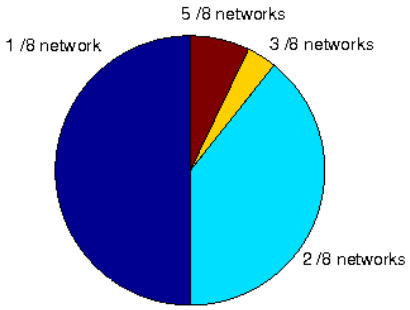

Fig. 11. /8 network distribution

4.2 Root causes analysis

Based on the nature of correlated groups, we classify them into four different families as follows: single root cause, variant signature attack tools, fingerprint worm, and multi-headed worm.

Table 4. Cluster description

Cluster Id	Ports sequences	Number of groups
15611	ICMP	7
15715	1433T	6
17466	135T	5
14647	445T	4
60231	5900T	4
60943	$ICMP, ICMP$	4
0	unclassified	3
17718	$ICMP\|445T$	3
175309	2967T	3
15238	139T	2
15610	ICMP	2
54623	1025T	2
65710	$1026U, 1026U, 1026U$	2
75851	$ICMP\|445T\|139T\|445T\|139T\|445T$	2
75853	$ICMP\|445T\|139T\|445T\|139\|T445T$	2
136244	$ICMP\|445T\|139T\|445T\|139T\|445T$	2
136323	$ICMP\|445T\|139T\|445T\|139\|T445T$	2
17470	1026U	1
65862	$1026U, 1026U$	1
72377	1028U	1
76768	$445T\|5000T\|445T\|5000T$	1
81280	$5900T, 5900T$	1
145554	$445T\|5000T\|445T\|5000T\|135T\|5000T$ $135T\|5000T\|135T\|5000T\|135T$	1
147436	$ICMP\|445T\|80T$	1
147476	$ICMP\|445T\|80T$	1
150691	$2967T, 2967T, 2967T$	1
164629	$2967T, 2967T$	1
168772	$1027U\|1028U\|1026U$	1
171073	$1027U\|1026U\|1028U$	1
174163	$1026U\|1028U\|1027U$	1

Single root cause Table 5 presents all single root cause groups. They correspond to phenomena where a single, and always the same, cluster is the root cause of the correlation of platform time series. They could have been easily detected by computing the correlation (still using a sliding window approach) between all the cluster time series corresponding to the same cluster on each platform. This is in contrast with the multi-headed tools that require comparisons between cluster time series that do not correspond to the same clusters.

As an example, the top plot of Figure 12 represents the attacks corresponding to cluster number 170309 on two platforms 7 and 27 from day 194 to day 290 (group 7 in Table 5), targeting Symantec System Center Agent (SSC Agent) service on port 2967 TCP. As we can see, its cluster time series on these two platforms are almost the same.

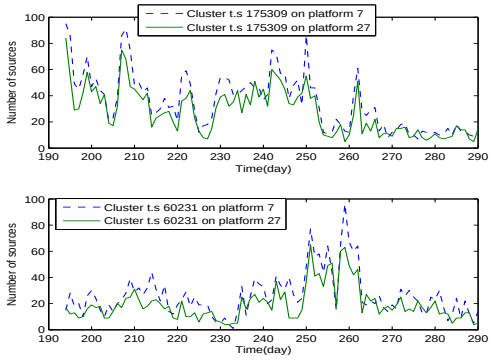

Fig. 12. Single root cause example

Table 5. Single root cause

group	platforms	root causes	start,end dates
1	6 8 22 24 26	17466	13,116
2	24 26	0	2,119
3	2 15	17466	31,91
4	7 27	15715	194,290
5	7 27	54623	198,263
6	7 27	60231	194,290
7	7 27	175309	194,290
8	6 8 17 22 26	17515	241,286
9	2 3 8 9 10 12 15 24 26	0	241,286
10	2 3 8 9 10 12 15 24 26	14647	412,452

The bottom plot represents the highly correlated attacks on the same two platforms and also during the same period, but related to cluster 60231, targeting Virtual Networking Computing service on port 5900 TCP (group 6 in Table 5). The interesting thing is that the attacks of these two clusters are totally dissimilar. This shows the usefulness of the sliding window technique during the root cause identification phase. We can see other groups related to these 2 platforms around the same period. A more in-depth analysis of these identified groups and clusters would reveal interesting findings, from a forensic point of view, highlighting relationships between phenomena which, otherwise, would have been studied isolated from each other. Instead, our grouping can help those in charge of attributing attacks to malicious actors, on the basis of their modus operandi.

Variant signature attack tools Our clustering algorithm classifies sources into clusters on a basis of a set of attributes such as the number of packets sent by the sources to our platforms, the ports sequences, the number of virtual hosts contacted,etc. Not all attack tools have a deterministic behavior. Some may probe ports in a random order, a variable number of times, etc. As a result, traces left by such tools will appear in distinct clusters

group	platforms	root causes	start,end dates	Multi-headed	Finger-print	Var-iant
11	5 13	15610 15611 17718 60943 75851 75853 136244 136323	52,119		Y	Y(1,2)
12	5 13	15611 17718	157,194		Y	
13	27	0 14647 145554	316,364		Y	
14	10 21	14647 15611 76768	332,364		Y	
15	9 27	72377 168772 171073 174163	371,408			Y(2)
16	24 26	15611 60943	419,452			Y(1)
17	7 27	175309 164629	73,119			Y(1)
18	8 11	17470 65862	156,202			Y(1)
19	7 27	60231 81280	316,364			Y(1)
20	1 2 3 6 10 12 15 22 24 26	15611 60943	2,119			Y(1)
21	6 8 10 17 22 24 26	150691 175309	56,91			Y(1)
22	9 23	15611 65710	405,448	Y		
23	2 15	15238,15715,60231	31,91	Y		
24	2 15	14647 15238 15715 17466	246,286	Y		
25	6 8 17 22 26	15715 17466	253,286	Y		
26	5 13 28	15610 15611 17718 60943 75851 75853 136244 136323 147436 147476	120,156	Y	Y	Y(1,2)
27	2 15	15715 17466 60231	163,194	Y		
28	7 8 27	54623 65710	214,245	Y		

Table 6. Multiple root cause groups

that will appear in correlated groups. Table 6 lists them with the value ”Y” in the column labeled ”Variant”. In this specific dataset, we found two reasons for which clusters can be ”splitted”.

The first one is that they have contacted a different number of targets (marked Y(1) in Table 6). One cluster contacts only 1 honeypot and the other cluster contacts two honeypots. By our observation, two-honeypot-contacted clusters have a smaller number of sources than the one-honeypot-contacted clusters. It may be explained as follows: if one source randomly chooses its target in a network, the probability for it to hit only one of our machines is much higher than to hit two (or even three) of them. As an example, the left plot of Figure 13 represents the attacks of all cluster time series related to group 26 in Table 6. The middle plot of Figure 13 represents only the attacks of two clusters 15611 and 60943 on platform 5. Cluster 15611 contacts 1 honeypot and cluster 60943 contacts two honeypots.

The other case is that the attack tool sends different amount of packets each time it attacks our platform. These groups are marked ’Y(2)’ in the ”Variant” column. The right plot of Figure 13 represents the attacks of three clusters numbered 75851, 75853 and 136323 also on platform 5. The three clusters have the same ports sequence:

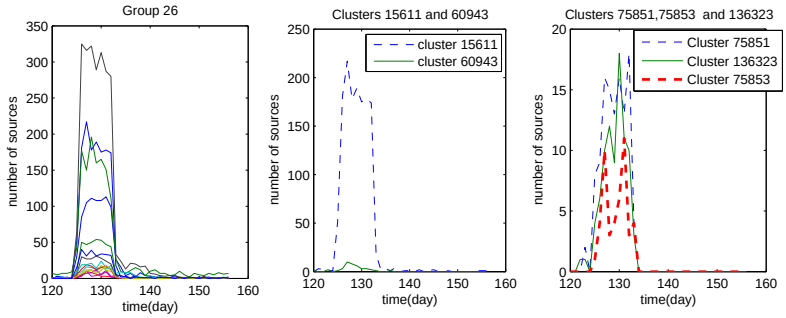

Fig. 13. Example variant worm

$ICMP|445T|139T|445T|139T|445T$. The difference resides in the number of packets sent by each source in these clusters.

Fingerprint worm OS fingerprint is a well-known attack tactic. The idea is that before launching the attack, the attacker checks the type of target system it faces and then launches, or not, the appropriate attack. We have found worms that automatized this idea. We call them "Fingerprint worm". If a fingerprint worm learns that it is attacking a non vulnerable host (w.r.t its attack model), it gives up. Since on our platforms, we deploy two kinds of virtual machines: Windows and Linux, the fingerprint worms will leave different traces on these two platforms. In terms of ports sequences, fingerprint worms may leave two different ports sequences on two kinds of virtual machines. One ports sequence may be the prefix of the other. We have found 5 cases of fingerprint worm in our dataset. They are presented in Table 6 with the value "Y" in the column "fingerprint". For instance, we plot 4 clusters numbered 75851, 75853, 136323, and 17718 of platform 5 from, again, group 26 (in Table 6) on Figure 14. The three clusters numbered 75851,75853 and 136323 (resp. 17718) have the corresponding ports sequence $ICMP|445T|139T|445T|139T|445T$ (resp. $ICMP|445T$). Cluster 17718 is mostly observed on the Linux machine (296 sources). There are only 64 sources that sent packets to the other two windows machines. The three other clusters however, are only observed on the two windows machines (251 sources in total). The explanation is that since port 445TCP is closed on the Linux machine, the attack tool is "intelligent enough" not to try port 139 TCP since it knows that the target is not vulnerable w.r.t its attacks. The fact that 64 sources have contacted the two Windows machines but have given up can probably be explained by packet losses, either in the network (e.g packet losses, firewall filters,etc..) or at the host (e.g congestion while launching too many scans in parallel). Here too, the identification of this class of attacks helps in understanding the threats on the Internet.

Multi-headed attack tools As being mentioned before, attack tools belonging to the multi-headed family have different attack techniques, but each time they use only one of them against the victim. The services targeted are usually different. Table 6 indicates

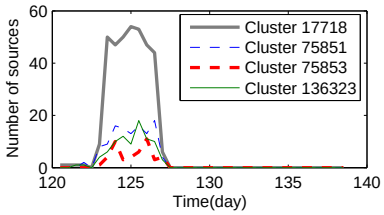

Fig. 14. Example Fingerprint worm

all the multi-headed groups we found. They have the ”Y” value in the column labeled ”Multi-headed”. As an example, group 23 in Figure 8 consists of three clusters targeting Microsoft NetBios Service (port 139 TCP), Virtual Network Computing service (port 5900 TCP) and Microsoft SQL Server (port 1433 TCP). The coordinated attacks of these three clusters spanned from day 31 to day 91. The top plot of Figure 15 represents group 24. It consists of four clusters numbered 14647(port 445 TCP), 15238 (port 139 TCP), 15715 (port 1433 TCP) and 17466(135 TCP). Their time series on platforms 2 and 15 are highly correlated from day 246 to day 286. As a sanity check, we found very low correlation coefficient between these cluster time series when computing their correlation coefficients over the whole period. For instance, the bottom plot of Figure 15 shows the dissimilitude of two cluster time series 17466 and 14647 on platform 2 from day 1 to day 245 (the interval just before the correlation). We could not have discovered this group if we had applied the algorithm for the whole period.

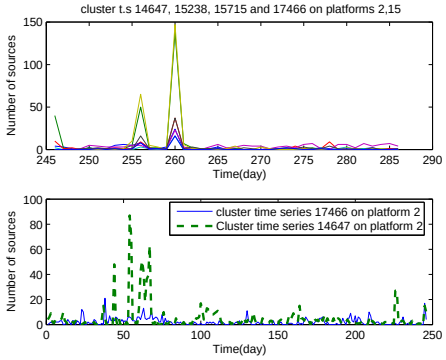

Fig. 15. Example of multi-header worm

5 Conclusion

In this paper, we revisit the problem of discovering multi-headed worms mentioned in [6], but in the context of a larger dataset collected from a distributed honeypot net-

work. Compared to the approach in [6] where correlation was investigated over the whole period of observation, our technique is able to look for correlation over smaller periods of time. To avoid comparing all possible cluster time series over different time windows, which is very costly, we worked around this issue by using filtered platform time series. Our expectation was that the phenomena we were looking for would be enough spatially and timely localized so as to be visible in the filtered platform time series over some periods of time. Applying our technique to a 15 month dataset, we are not only able to confirm the existence of multi-headed worms (on many places), but also bring to the community insight knowledge about worm behaviours. Besides that, the results obtained can also be used to improve our clustering algorithm. However, work remains to take full advantage of the obtained result in order to carry out a systematic analysis of the identified phenomena and to help in studying the so called attack attribution problem.

References

1. Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.
2. Mark W. Eichin and Jon A. A. Rochlis. With microscope and tweezers: An analysis of the internet virus of november 1988. In *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland, Ohio, 1989.
3. Greg Hoglund and Gary Mc Graw. *Exploiting Software: How to Break Code*. Addison-Wesley Professional, 2004.
4. Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, New York, NY, USA, 2003. ACM Press.
5. Fabien Pouget and Marc Dacier. Honeypot-based forensics. In *AusCERT2004, AusCERT Asia Pacific Information technology Security Conference 2004, 23rd - 27th May 2004, Brisbane, Australia*, May 2004.
6. Fabien Pouget, Guillaume Urvoy Keller, and Marc Dacier. Time signatures to detect multi-headed stealthy attack tools. In *18th Annual FIRST Conference, June 25-30, 2006, Baltimore, USA*, Jun 2006.
7. Niels Provos. Home page of honeyd. see http://www.honeyd.org/.
8. J. Shoch and J. Hupp. The worm programs: Early experience with a distributed computation. *Commun. ACM*, 25(3):172–180, 1982.
9. Eugene H. Spafford. The internet worm program: an analysis. *SIGCOMM Comput. Commun. Rev.*, 19(1):17–57, 1989.
10. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, Berkeley, CA, USA, 2002. USENIX Association.
11. William Trochim and James P. Donnelly. *The Research Methods Knowledge Base*. Atomic Dog, December 2006.
12. Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 11–18, New York, NY, USA, 2003. ACM Press.