The WOMBAT Attack Attribution method: some results

Marc Dacier[1], Van-Hau Pham[2], and Olivier Thonnard[3]

[1] Symantec Research
Sophia Antipolis, France
marc_dacier@symantec.com
[2] Institut Eurecom
2229 Route des Crètes,
Sophia Antipolis, France
van-hau.pham@eurecom.fr
[3] Royal Military Academy
Polytechnic Faculty
Brussels, Belgium
olivier.thonnard@rma.ac.be

Abstract. In this paper, we present a new *attack attribution* method that has been developed within the WOMBAT[4] project. We illustrate the method with some real-world results obtained when applying it to almost two years of attack traces collected by low interaction honeypots. This analytical method aims at identifying large scale attack phenomena composed of IP sources that are linked to the same root cause. All malicious sources involved in a same phenomenon constitute what we call a *Misbehaving Cloud* (MC). The paper offers an overview of the various steps the method goes through to identify these clouds, providing pointers to external references for more detailed information. Four instances of misbehaving clouds are then described in some more depth to demonstrate the meaningfulness of the concept.

1 Introduction

There is no real consensus on the definition of "attack attribution" in the cyber domain. Most previous work related to that field tend to use the term "attribution" as a synonym for *traceback*, which consists in "determining the identity or location of an attacker or an attacker's intermediary" [25]. In the context of a cyber-attack, the obtained identity can refer to a person's name, an account, an alias, or similar information associated with a person or an organisation. The location may include physical (geographic) location, or any virtual address such as an IP address or Ethernet address. The rationale for developing such attribution techniques is mainly due to the untrusted nature of the IP protocol, in which the source IP address is not authenticated and can thus be easily spoofed.

[4] Worldwide Observatory of Malicious Behaviors and Threats - http://www.wombat-project.eu

An extensive survey of attack attribution techniques used in the context of IP traceback can be found in [25].

In this paper, we refer to "attack attribution" as something quite different from what is described here above. We are primarily concerned with larger scale attacks. In this context, we aim at developing an analytical method to help security analysts in determining their root causes and in deriving their *modus operandi*. These phenomena can be observed through many different means (e.g., honeypots, IDS's, sandboxes, web crawlers, malware collecting systems, etc). In most cases, we believe that attack phenomena manifest themselves through so-called "attack events", which can be observed with distributed sensors that are deployed in the Internet. Typical examples of attack phenomena that we want to identify vary from worm or malware families that propagate through code injection attacks [9], to established botnets controlled by the same people and targeting machines in the IP space. All malicious sources involved in the same root phenomenon constitute what we call a *Misbehaving Cloud* (MC).

The structure of the paper is as follows: Section 2 describes the experimental environment used to validate the method presented. Section 3 offers a high level overview of the attack attribution method defined within the WOMBAT project and Section 4 gives some more information on the multi criteria fusion approach used in the method. Section 5 discusses a couple of illustrative examples obtained by applying the method on honeynet traces, and Section 6 concludes the paper.

2 Description of the experimental environment

This paper offers an empirical analysis of some attacks collected during two years by a set of low interaction honeypots deployed all over the world by the Leurré.com Project [10]. We refer the interested reader to [8, 19] for an in-depth presentation of the data collection infrastructure. From an analytical viewpoint, our attack attribution method builds upon previous results, namely [18, 4, 16, 24, 17]. For the sake of clarity, we start by introducing some important terms that have been defined in these previous publications.

2.1 Terminology

1. **Platform**: A physical machine running three virtual honeypots, which emulate three distinct machines thanks to *honeyd* [20]. A platform is connected directly to the Internet and collects tcpdump traces that are gathered on a daily basis in a centralized database [10].
2. **Source**: an IP address that has sent at least one packet to, at least, one platform. An IP address remains associated to a given Source as long as no more than 25 hours[5] elapse between two packets sent by that IP. After such

[5] By grouping packets by originating sources instead of by IPs, we minimize the risk of mixing together the activities of two distinct physical machines (as a side effect of the dynamic address allocation implemented by ISP's).

a delay, the IP will be associated to a new source identifier if we observe it again.

3. **Attack**: refers to all packets exchanged between a malicious source and a platform.
4. **Cluster**: all the sources that execute the same attack against any of the platforms constitute an *(attack) Cluster*. In practice, such a cluster groups all malicious sources that have left highly similar network traces on our platforms. How to identify clusters and how those clusters look like are issues that have been explained in other publications [18, 8].

2.2 Honeynet dataset

Machines used in the Leurré.com project are maintained by partners all over the world, on a voluntary basis. Some of these platforms can thus become unavailable. In the context of this paper, we wanted to apply our analytical method on a dataset that would be, as much as possible, unimpacted by these operational issues. Therefore, we have selected a subset of 40 stable platforms from all platforms at our disposal. A total of 3,477,976 attacks have been observed by those platforms. We represent the total number of attacks per day over the whole analysis period (800 days, from Sep 2006 until November 2008), as a time series denoted by TS. Similarly, we can represent, for each platform, the number of attacks observed on it, on a daily basis. This leads to the definition of 40 distinct attack time series (each made of 800 points), denoted by TS_X where X represents a platform identifier.

We can go even further in splitting our time series in order to represent which type of attack was observed on which platform. To do this, we split each TS_X into as many time series as there are *attack clusters*, as defined before. These newly obtained time series are represented by $\Phi_{[0-800),c_i,p_j} \forall \,\text{cluster}\, c_i$ and $\forall \,\text{platform}\, p_j$. That is, the i^{th} point of the time series $\Phi_{[0-800),X,Y}$ represents the amount of sources attacking, on day i, the platform Y by means of the attack defined by the cluster identifier X. We represent by TS_L2 the set of all these observed cluster time series (in total, 395,712 time series).

In [17], it has been shown that a large fraction of these time series barely vary in amplitude on a daily basis. This continuous, low-intensity activity is also referred to as the Internet *background radiation* [13]. In this paper, we do not consider those flat curves, and we instead focus on time series that show some significant variations over time, indicating the existence of some ephemeral phenomena. To automatically identify these time series of interest, we have applied the method presented in [17], which finally gives a subset of time series denoted by TS_L2'. In our dataset, TS_L2' contains now only 2,127 distinct time series. However, they still comprise a total of 2,538,922 malicious sources. TS_L2' represents the set of time series we have used for this analysis.

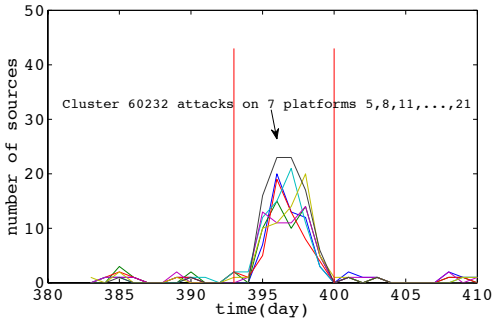


Fig. 1. An example of $\mathcal{M}$-event, composed of seven μ-events (on seven different platforms) that are correlated in the same time interval. Cluster 60332 corresponds to a malicious activity on the VNC port (5900/TCP).

3 Overview of WOMBAT attribution method

The WOMBAT attack attribution method is made of two distinct steps. In the very first one, we identify periods of time where some of the time series from TS_L2' exhibit a pattern that indicate that a specific phenomenon worth of interest is happening. We call a *micro attack event* such period of time for a given time series from TS_L2'. Moreover, we call *macro attack event* a group of micro attack events that are correlated during the same period of time.

The second step of the method consists in characterizing each of these *micro attack events* and in trying to establish connections between them. All micro attack events that share enough features constitute what we call a *Misbehaving Cloud* (MC). We hypothesize that all malicious sources involved in a Misbehaving Cloud have a common root cause. By identifying them and studying their global behavior, we hope to get a better insight into the modus operandi and the strategies of those responsible for them.

We further detail the two steps of the method in the next subsections.

3.1 Step 1: Micro and Macro attack events identification

Definition (μ-event): A *micro attack event* (or μ-event) is defined by a tuple $(\mathcal{T}, \mathcal{C}_i)$ where $\mathcal{T}$ represents a limited period of time (typically a few days) during which a significant attack activity is observed, and $\mathcal{C}_i$ represents the time series corresponding to cluster $\mathcal{C}$ observed on the platform i.
Definition ($\mathcal{M}$-event): A set of micro attack events observed over the same period of time, and during which the corresponding time series are strongly correlated is defined as a *macro attack event* (or $\mathcal{M}$-event).

Figure 1 illustrates this concept by representing a $\mathcal{M}$-event composed of seven μ-events that are correlated in the same time interval.

Identification of μ-events. The micro attack event identification relies mostly on some well-known signal processing techniques. The goal is to segment the time series into periods of interest. Such periods are characterized by some *intense period* of activities isolated by periods of very stable or non existent activities. Several techniques exist to detect abrupt changes in a signal [1]. In this paper, the method we have used is the one that has been precisely presented in [15].

Identification of $\mathcal{M}$-event. Once we have identified all μ-events of interest in our dataset, we need to identify all those that are strongly correlated over the same period of time, which form thus a $\mathcal{M}$-event. The problem is not as trivial as it may sound, because *i)* μ-events may have overlapping periods, and *ii)* within a given period of time, several distinct phenomena may have taken place. Here too, we have presented and compared various approaches and we refer the interested reader to [17, 15] for an in-depth explanation of the algorithms used.

3.2 Step 2: Multi criteria fusion of attack events features

The purpose of this second step consists in deciding whether several distinct μ-events are likely due to a same root phenomenon (i.e., the same Misbehaving Cloud), on the basis of different characteristics derived from the network traffic generated by malicious sources involved in such events.

Our approach is based on three components:

1. Attack Feature Selection: we determine which *attack features* we want to include in the fusion process, and we thus characterize each μ-event according to this set of features;
2. Graph-based Clustering: a graph of μ-events is created regarding each feature, based on an appropriate distance for measuring pairwise similarities. Fully connected components can then be identified within each graph;
3. Multi criteria fusion: the different graphs are then combined using an *aggregation function* that models some dynamic behavior.

This approach is mostly unsupervised, i.e., it does not rely on a preliminary training phase to attribute μ-events to larger scale phenomena. In the next Section, we describe the three steps of this method.

4 On the Multi criteria fusion approach

4.1 Attack Features Selection

In most clustering tasks, the very first step consists in selecting some key characteristics from the dataset, i.e., salient features that may reveal meaningful *patterns* [6]. In this analysis, we have selected some *features* that we consider useful to analyze the behavior of global phenomena.

One of the key features used in this attribution technique is the spatial distributions of malicious sources involved in μ-events, in terms of originating countries and IP blocks. Looking at these statistical characteristics may reveal attack

activities having a specific distribution of originating countries or IP networks, which can help for instance to confirm the existence of "unclean networks" [3]. In practice, for each μ-event, we create a feature vector representing the distribution of countries of sources (as a result of the IP to geolocation mapping), or a vector representing the distribution of IP addresses (grouped by their Class A-prefix, to limit the vector's size).

We have also selected an attack characteristic related to the *targeted platforms*. Looking at which specific platform has observed a μ-event is certainly a pertinent feature. At the same time, we combine this information with the $\mathcal{M}$-event identification, since (by definition) $\mathcal{M}$-events are composed of μ-events that are strongly correlated in time (which indicates a certain degree of coordination among them).

Besides the origins and the targets, the *type of activity* performed by the attackers seems also relevant. In fact, worm or bot software is often crafted with a certain number of available exploits targeting a given set of TCP or UDP ports. So, it makes sense to take advantage of similarities between the *sequences of ports* that have been probed or exploited by malicious sources.

Finally, we have decided to compute, for each pair of μ-events, the ratio of common IP addresses. We are aware of the fact that, as time passes, some machines of a given botnet (or misbehaving cloud) might be cured while others may get infected (and thus join the cloud). Additionally, certain ISPs apply a quite dynamic policy of IP allocation for residential users, which means that infected machines can have different IP addresses when we observe them at different moments. Nevertheless, considering the huge size of the IP space, it is still reasonable to expect that two μ-events are probably related to the same root phenomenon when they have a high percentage of IP addresses in common.

To summarize, and to provide a short-hand notation in the rest of this paper, for each μ-event we define a set of features that we denote by:

$$F = \{F_i\}\,,\ i \in \{geo, sub, targ, ps, cip\}$$

where:

$$\begin{cases} geo = \text{geolocation, as a result of mapping IP addresses to countries;} \\ sub = \text{distribution of sources IP addresses (grouped by Class A-subnet);} \\ targ = \text{targeted platforms + degree of coordination } (\mathcal{M}\text{-event membership);} \\ ps = \text{port sequences probed or targeted by malicious sources;} \\ cip = \text{feature representing the ratio of common IP addresses among sources;} \end{cases}$$

4.2 Graph-based Clustering

The second component of our attribution method implements an unsupervised clustering technique that aims at discovering groups of strongly connected μ-events, when these are represented within a graph. In [22, 23], we have given a detailed description of this graph-based clustering technique. However, to make this paper as self-contained as possible, we briefly describe the high-level principles of this technique.

As defined by Jain and Dubes in [6], many typical clustering tasks involve the following steps:

i) feature selection and/or extraction (as described in the previous Subsection);
ii) definition of an appropriate distance for measuring the similarities between pairs of elements with respect to a given feature;
iii) application of a grouping algorithm, such as the classical hierarchical clustering or K-means algorithm;
iv) data abstraction (if needed), to provide a compact representation of each cluster;
v) optionally, the assessment of the clusters quality and coherence, e.g. by means of validity indices.

Steps (iv) and (v), while important, lie outside the scope of this paper. Instead, we will simply use four anecdotal examples to intuitively demonstrate the quality, i.e., the meaningfulness, of the groups created by the method. Steps (ii) and (iii) are described here after.

Choosing a distance function How to measure *pairwise similarities* between two feature vectors is obviously an important step, since it will have an impact on the coherence and the quality of the resulting clusters.

When we have to deal with observations that are in the form of probability distributions (or frequencies), like in the case of features F_{geo} and F_{sub}, we need to rely on statistical distances. One commonly used technique is the Kullback-Leibler divergence [7]. Let p_1 and p_2 be for instance two probability distributions over a discrete space X, then the K-L divergence of p_2 from p_1 is defined as:

$$D_{KL}(p_1||p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)} \tag{1}$$

which is also called the information divergence (or *relative entropy*). Because D_{KL} is not considered as a true metric, it is usually better to use instead the Jensen-Shannon divergence (JSD) [11], defined as:

$$JS(p_1, p_2) = \frac{D_{KL}(p_1||\bar{p}) + D_{KL}(p_2||\bar{p})}{2} \tag{2}$$

where $\bar{p} = (p_1 + p_2)/2$. In other words, the Jensen-Shannon divergence is the *average* of the KL-divergences to the *average distribution*.

Finally, to transform pairwise distances d_{ij} to similarity weights sim_{ij}, we still have to define a mapping function. Previous studies found that the similarity between stimuli decay exponentially with some power of the perceptual measure distance [21]. As customary, we can thus use the following functional form to do this transformation:

$$sim(i,j) = exp(\frac{-{d_{ij}}^2}{\sigma^2}) \tag{3}$$

where σ is a positive real number that affects the decreasing rate of w.

Measuring pairwise similarities for the other considered features ($F_{targ}, F_{ps}, F_{cip}$) is more straightforward. In those cases, we can use simpler distance functions, such as the *Jaccard similarity coefficient*. Let s_1 and s_2 be two sample sets (for instance with F_{ps}, s_1 and s_2 are sets of ports that have been probed by sources of two μ-events), then the Jaccard coefficent is defined as the size of the intersection divided by the size of the union of the sample sets, i.e.:

$$sim(i,j) = \frac{|s_1 \bigcap s_2|}{|s_1 \bigcup s_2|}$$

The Jaccard similarity coefficient can also be used to compute the ratio of common IP addresses between attack events (F_{cip}). Regarding F_{targ}, a simple weighted means is used to combine two scores: *i)* one score in $[0, 1]$ as given by the simple comparison of the two targeted platforms, and *ii)* another score (also in $[0, 1]$) indicating whether two μ-events belong to the same $\mathcal{M}$-event (indicating a time coordination).

Grouping algorithm In this step, we formulate the problem of clustering μ-events using a graph-based approach. The vertices (or nodes) of the graph represent the patterns (or feature vectors) of the μ-events, and the edges (or links) express the similarities between μ-events, as calculated with the distance metrics described before. Then, we can extract so-called *maximal cliques* from the graph, where a maximal clique is defined as an induced subgraph in which the vertices are fully connected and it is not contained within any other clique. To do this, we use the *dominant sets* approach of Pavan et al. [14], which proved to be an effective method for finding maximal *weighted* cliques. This means that the weight of every edge (i.e., the relative similarity value) is also considered by the algorithm, as it seeks to discover maximal cliques whose total weight is maximized.

By repeating this process, we can thus create an undirected edge-weighted graph G_i for each attack feature F_i, in which the edges are similarity weights $\in [0, 1]$ that can be seen as *relatedness degrees* between μ-events (where a zero value indicates totally unrelated events). Then, the clique algorithm extracts one set of cliques per feature, which reveals the cohesions among μ-events regarding each F_i.

4.3 Multi-Criteria Aggregation

Definition (Aggregation function). An aggregation function is formally defined as a function of n arguments ($n > 1$) that maps the (n-dimensional) unit cube onto the unit interval: $f : [0, 1]^n \longrightarrow [0, 1]$, with the following properties [2]:

(i) $f(\underbrace{0, 0, \ldots, 0}_{n\text{-}times}) = 0$ and $f(\underbrace{1, 1, \ldots, 1}_{n\text{-}times}) = 1$

(ii) $x_i \leq y_i$ for all $i \in \{1, \ldots, n\}$ implies $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$

Aggregation functions are used in many prototypical situations where we have several criteria of concern, with respect to which we assess different options. The objective consists in calculating a combined score for each option, and this combined output forms then a basis from which decisions can be made. For example, aggregation functions are largely used in problems of *multi criteria decision analysis* (MCDA), in which an alternative has to be chosen based on several, sometimes conflicting criteria. Usually, the alternatives are evaluated from different attributes (or features) that are expressed with numerical values representing a degree of preference, or a degree of membership.

In our application, we have n different attack features given by the F_i's, and thus a vector of criteria $\mathbf{x} \in [0,1]^n$ can be constructed from the similarity weights, i.e., $x_i = A_i(j,k)$, with A_i being the similarity matrix of graph G_i corresponding to attack feature F_i. Our approach consists in combining the n values of each criteria vector $\mathbf{x}$ (which reflect the set of all relationships between a pair of μ-events), in order to build an aggregated graph $G' = \sum G_i$ from which we can then extract the connected components. A straightforward but rather simplistic approach would consist in combining the criteria using a simple arithmetic mean, or by assigning different weights to each criteria (weighted mean). However, this does not allow us to model more complex behaviors, such as "most of", or "at least two" criteria to be satisfied in the overall decision function. Yager has introduced in [26] a type of operator called *Ordered Weighted Averaging* (OWA), which allows to include certain relationships between multiple criteria in the aggregation process. An OWA aggregation operator differs from a classical weighted means in that the weights are not associated with particular inputs, but rather with their *magnitude*. As a result, OWA can emphasize the largest, smallest or mid-range values. It has become very popular in the research community working on fuzzy sets.

Definition (OWA). For a given weighting vector $\mathbf{w}$, $w_i \geq 0$, $\sum w_i = 1$, the OWA aggregation function is defined by:

$$OWA_w(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{\searrow(i)} = < \mathbf{w}, \mathbf{x}_{\searrow} > \tag{4}$$

where we use the notation $\mathbf{x}_{\searrow}$ to represent the vector obtained from $\mathbf{x}$ by arranging its components in decreasing order: $x_{(1)} \geq x_{(2)} \geq \ldots \geq x_{(n)}$.

It is easy to see that for any weighting vector w, the result of OWA lies between the classical **and** (=min) and **or** (=max) operators, which are in fact the two extreme cases when $\mathbf{w} = (0, 0, \ldots, 1)$ (then $OWA_w(\mathbf{x}) = min(\mathbf{x})$) or when $\mathbf{w} = (1, 0, \ldots, 0)$ (then $OWA_w(\mathbf{x}) = max(\mathbf{x})$). Another special case is when all weights $w_i = \frac{1}{n}$, which results in obtaining the classical arithmetic mean.

To define the weights w_i to be used in OWA, Yager suggests two possible approaches: either to use some learning mechanism with sample data and a regression model (i.e., fitting weights by using training data and minimizing the least-square residual error), or to give some semantics to the w_i's by asking an expert

to provide directly those values, based on domain knowledge. We selected the latter approach by defining the weighting vector as $\mathbf{w} = (0.1, 0.35, 0.35, 0.1, 0.1)$, which translates our intuition about the dynamic behavior of large-scale attack phenomena. It can be interpreted as: *at least three criteria must be satisfied, but the first criteria is of less importance compared to the* 2^{nd} *and* 3^{rd} *ones* (because only one correlated feature between two μ-events might be due to chance only).

These weights must be carefully chosen in order to avoid an unfortunate linkage between μ-events when, for example, two events involve IP sources originating from popular countries and targeting common (Windows) ports in the same interval of time (but in reality, those events are not due to the same phenomenon). By considering different worst-case scenarios, we verified that the values of the weighting vector $\mathbf{w}$ work as expected, i.e., that it minimizes the final output value in such undesirable cases. Moreover, these considerations enable us to fix our decision threshold to an empirical value of about 0.25, which has been also validated by a sensibility analysis. In other words, all combined values that are under this threshold will be set to zero, leading to the removal of corresponding edges in the aggregated graph G'.

Finally, we can easily identify misbehaving clouds by extracting the connected components (or subgraphs) from G'. As a result, for any subset of events of a given MC, we will find a sufficient number of evidences that explain why those events have been linked together by the multi criteria aggregation process.

5 Experimental Results

5.1 Overview

When applying the technique described in Section 3.1 to the dataset described in Section 2.2, we obtain 690 $\mathcal{M}$-events which consist of 2454 μ-events. We use these μ-events as input for the multi-criteria fusion approach (Section 4), and we consequently identify 83 Misbehaving Clouds (MCs), which correspond to 1607 μ-events, and 506,835 attacking sources. The phenomena involve almost all common services such as NetBios (ports 139/TCP, 445/TCP), Windows DCOM Service (port 135/TCP), Virtual Network Computing (port 5900/TCP), Microsoft SQL Server (port 1433/TCP), Windows Messenger Service (ports 1025-1028/UDP), Symantec Agent (port 2967/TCP), and some others. Figure 7a shows the distribution of μ-events per MC. As we can see, in most cases, the MCs contain few μ-events. However, around 20% of MCs contain more than 15 μ-events, and some even contain up to 300 events. Figure 7b represent the CDF of the MCs lifetime. Such lifetime is defined as the time interval, in days, between the very first and the very last attack event of a given MC. As showed in Figure 7b, 67% of MCs exist during less than 50 days but around 22% of them last for more than 200 days.

Figure 7c represents the CDF of the number of platforms targeted by MC. As showed in the Figure, in 94% of the cases, the MCs are seen on less than 10 platforms.

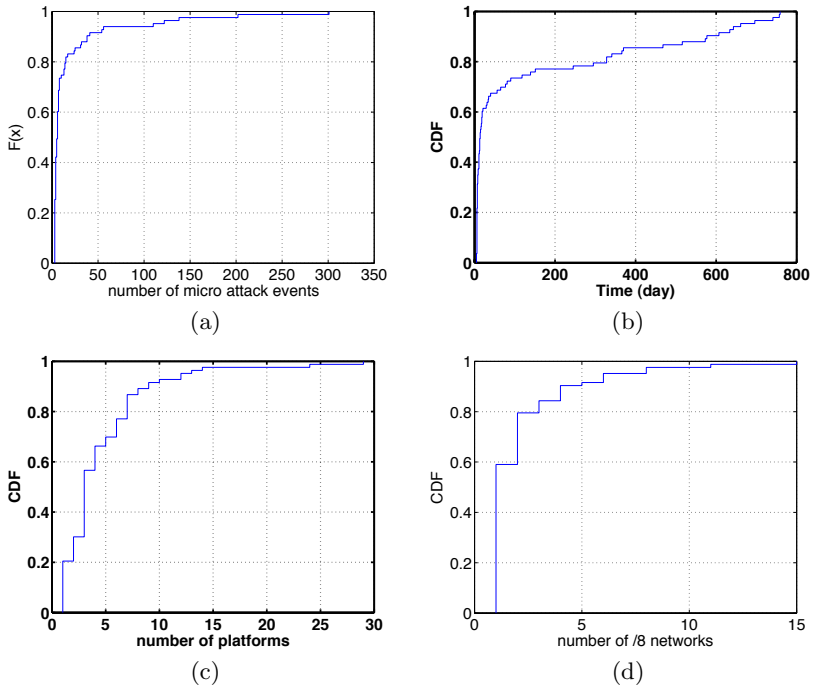


Fig. 2. Some global characteristics of the obtained MCs

These various characteristics suggest that the root causes behind the existence of these MCs are fairly stable, localised attack processes. In other words, different places of the world do observe different kind of attackers but their modus operandi remain stable over a long period of time. We are, apparently, not that good at stopping them from misbehaving.

5.2 Case Studies

It is certainly not our intention to detail extensively the behavior and characteristics of every MC that has been found in our 2-year data set. Instead, in this Section, we detail only four MCs, which, although anecdotal, still reflect the kind of findings that our method can provide automatically. Table 1 provides some high-level characteristics of these four MCs phenomena under study. Each MC is analyzed in some detail in the following pages.

***MC2*: Worm-behaving cloud.** $MC2$ consists of 122 μ-attack events. These μ-events exhibit a shape which is fairly similar to the one left by a typical worm: its trace exists for several days, it has a small amplitude at the beginning but grows quickly, exhibits important drops that can correspond to subnets being

Table 1. High-level characteristics of four *MCs* under study. The colon *Root cause* refers to the presumed type of phenomenon, based on the results of the attack attribution method.

MC Id Nr	Events Nr	Sources	Duration	Root cause	Targeted ports
2	122	45,261	741	Worm-behaving cloud	1433T (MSSQL), 1025T (RPC), 139T (Netbios), 5900T (VNC), 2967T (Symantec)
3	56	48,007	634	UDP spammers (botnet)	1026U (Windows Messenger)
10	138	26,243	573	P2P	Unusual ephemeral ports (TCP)
20	110	195,018	696	UDP spammers (botnet)	1026U, 1027U, 1028U

cured or blacklisted, and it eventually dies slowly (see [15] for a more formal description of this class of phenomena).

The interesting thing with $MC2$ is that it is made of a sequence of *worm-like* shaped μ-events. The lifetime of this MC is 741 days! It is composed of μ-events that have targeted a number of distinct services, including 1025T, 139T, 1433T, 2967T and 5900T. The results of the multi-criteria fusion algorithm indicate that those μ-events have been grouped together mainly because of the following three features: geographical location, targeted platform, and ports sequence. Moreover, a detailed analysis reveals that an important amount of IP addresses is shared by many μ-events composing this MC.

To illustrate the kinds of μ-events found in this MC, Figures 3a and 3b represent four μ-events time series. Figure 3a represents two of them, namely e626 and e628, consisting of activities against Microsoft SQL Server (1433/TCP). Whereas Figure 3b represents the other two, namely e250 and e251, consisting of activities against a Symantec Service (2967/TCP). Figure 3c zooms on these last two μ-events from day 100 to day 150. We can observe the slow increase of the two curves that are typical of worm-related attacks [15, 27].

The two μ-events on the left (resp. middle) share 528 (resp. 1754) common IP addresses with each other. Given these elements, we are tempted to believe that e626 and e628 (resp. e250 and e251) are generated by the same worm, called $WORM\_A$ (resp. called $WORM\_B$). Both worms, $WORM\_A$ and $WORM\_B$, target the same two platforms: 25 and 64. Furthermore, we found that these four μ-events share an important amount of common compromised machines. This could indicate that both worms, before having contacted our honeypots, had contaminated a relatively similar population of machines. A plausible explanation could be that both had been launched from the same initial set of machines and that they were using the same, or similar, code to choose their targets.

From the attack vector point of view, these two worms have nothing in common since they use very different types of exploits. Furthermore, they have been active in different periods of time. However, the analysis reveals that they exhibit a very similar pattern both in terms of propagation strategy and in terms of success rates. Thus, even if the infection vector differs between the two, the starting point of the infection as well as the code responsible for the propagation are, as explained, quite likely very similar. This reasoning can be generalized to all 122 μ-events, revealing the high probability that all these different attack

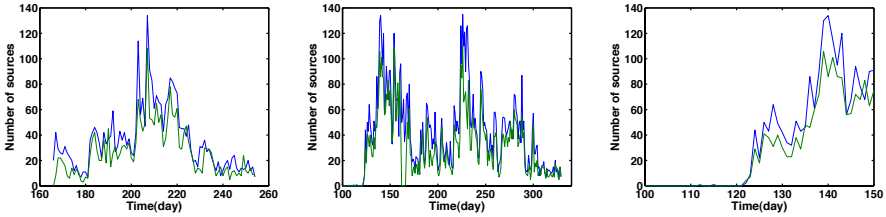


Fig. 3. Attack time series (nr of sources by day) of some μ-events from $MC2$, targeting (a) MS SQL Server (1433/TCP), (b) Symantec agent (2967/TCP). Fig. (c) is a zoom on (b).

phenomena have some common root cause(s). This does not, per se, mean that all these attacks are due to the very same person or organisation -even if this is likely- but it indicates that the same core piece of code has probably been reused, from a very similar starting point to launch a number of distinct attacks. This reveals some aspect of the modus operandi of those who have launched these attacks and this is an important piece of information for those who are in charge of identifying these misbehaving groups and their tactics.

***MC*3 and *MC*20: Windows Messenger Spammer.** In this other case study, we look at two distinct MCs: $MC3$ and $MC20$. Both are made of μ-events that have exclusively tried to send spam to innocent victims thanks to the Windows Messenger service, using UDP packets. Both MCs have been observed over a large period of time, more than 600 days in both cases. Even if they, conceptually, look similar, there are important differences between $MC3$ and $MC20$. First, the targeted ports are not identical: in $MC3$, UDP packets are being sent to three different UDP ports, namely 1026, 1027 and 1028, while in $MC20$ packets are sent exclusively to the 1026 UDP port. Then, as illustrated in Fig.4 where we can see the cumulative distribution (CDF) of sources IP addresses (grouped by /8 blocks of addresses), we observe that $MC3$ is uniformly distributed in the IPv4 space. This result is absurde since large portions of the IPv4 space can not be allocated to individual machines (multicast, bogons, unassigned, etc.) and, in all these regions, it is impossible to find compromised machines sending spams. If we find these IPs in packets hitting our honeypots, it clearly means that these are spoofed IP addresses. Furthermore, the uniform distribution of all the IP addresses in that MC leads us to believe that all other IPs are also spoofed. On the other hand, $MC20$ has a constant distribution pointing exclusively to a single /8 block owned by an ISP located in Canada[6]. A likely explanation is that those spammers have also used spoofed addresses

[6] Actually, a closer inspection of sources IP addresses reveals they were randomly chosen from only two distinct /16 blocks from this same /8 IP subnet.

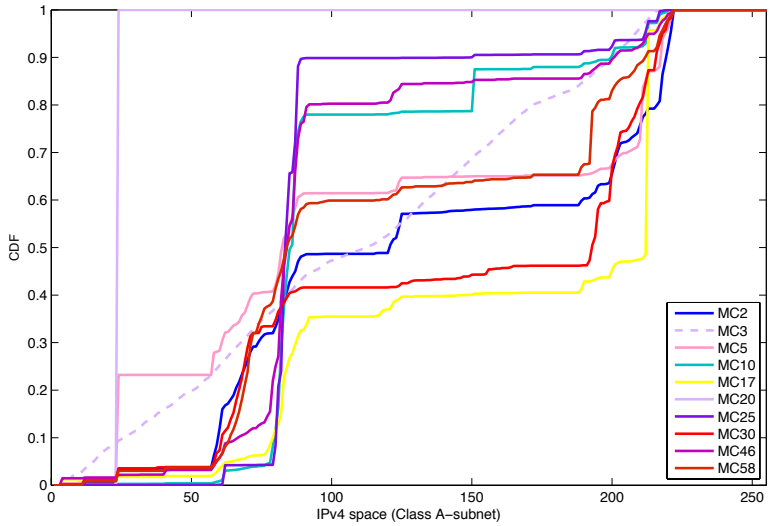


Fig. 4. CDF's of originating IP subnet distributions for the largest phenomena.

to send UDP messages to the Windows Messenger service, and they have been able to do so for 600 days without being disturbed!

To further validate these results, we also looked at the payloads of the UDP packets by computing a hash for each packet payload. What we discovered is quite surprising: all payloads sent by the sources have exactly the same message template, but the template was different for the two clouds. Fig.5 and Fig.6 show the two different templates used by spammers of $MC3$ and $MC20$ respectively. Regarding $MC3$, we also observe many alternate URL's, such as: 32sys.com, Fix64.com, Key32.com, Reg64.com, Regsys32.com, Scan32.com, etc, whereas spammers in $MC20$ use apparently almost[7] always the same URL (www.registrycleanerxp.com).

This knowledge has been derived from the observation of the MCs automatically built by our method. This illustrates the richness and meaningfulness of the analyses that can be performed. At this point, there are still two questions left unanswered when we look at those two UDP spam phenomena:

i) Do all those UDP packets really use spoofed IP addresses, and how were they sent (e.g., from a single machine in the Internet or from a very large botnet)?
ii) Could it be that those two phenomena have in fact the same root cause, i.e., the same (group of) people running in parallel two different spam campaigns?

[7] For $MC20$, only a few instances of spam messages were observed with a different URL: nowfixpc.com

```
SYSTEM  ALERT  - STOP! WINDOWS REQUIRES IMMEDIATE ATTENTION.
Windows has found CRITICAL SYSTEM ERRORS.

To fix the errors please do the following:
1. Download Registry Cleaner from: http://www.wfix32.com
2. Install Registry Cleaner
3. Run Registry Cleaner
4. Reboot your computer
FAILURE TO ACT NOW MAY LEAD TO DATA LOSS AND CORRUPTION!
```

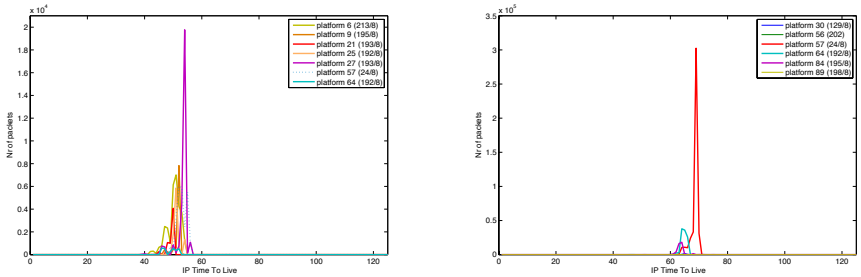
Fig. 5. Spam template used in $MC3$.

```
Local System User
CRITICAL ERROR MESSAGE! - REGISTRY DAMAGED AND CORRUPTED.

To FIX this problem:
Open Internet Explorer and type:  www.registrycleanerxp.com
Once you load the web page, close this message window

After you install the cleaner program
you will not receive any more reminders or pop-ups like this.

VISIT www.registrycleanerxp.com IMMEDIATELY!
```

Fig. 6. Spam template used in $MC20$.

To answer the first question, we have extracted from the UDP packets the Time To Live (TTL) value of their IP headers. We have computed the distributions of these TTL values for both phenomena, grouped by targeted platform. The results, illustrated in Fig.7, seems to confirm our intuition about spoofed UDP packets, since these TTL distributions are too narrow to originate from a real population of physical machines. In both cases ($MC3$ and $MC20$), we observe that the TTL distributions have a width of about 5 hops, whereas TTL distributions for non-spoofed packets are normally much larger, certainly when sources are largely distributed. As a sanity check, we retrieved the TTL distributions for another phenomenon, which has been validated as a botnet of machines. As one can see in Fig.8, the TTL distributions are much larger (around 20 hops) than for spoofed UDP packets. Another finding visible in Fig.7 is the unusual initial value used for TTL's, which also indicates that those packets were probably forged using raw sockets, instead of using the TCP/IP protocol stack of the operating system.

Finally, trying to answer the last question (same root cause or not), we looked at one additional feature of the attacks. We generated a distribution of sources by grouping them based on the *day and hour of the week* they have been observed by our platforms (using the same universal time reference, which is GMT+1 in this case). As one can see in Fig.9, the result is very intriguing: although there



(a) (b)

Fig. 7. TTL distribution of UDP packets for $MC3$ (a) and $MC20$ (b) (grouped by targeted platform)

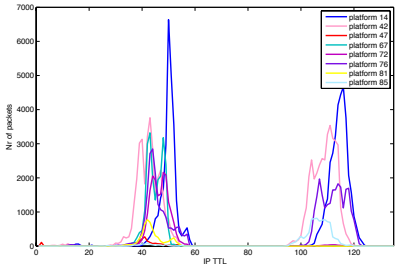


Fig. 8. TTL distribution of TCP packets for a phenomenon (MC28) attributed to a botnet targeting ports 445T and 139T (grouped by targeted platform).

is no privileged day or time interval in the week on which we observe a specific pattern, the UDP traffic created by $MC3$ (in dashed) and $MC20$ (in green) look apparently synchronized. Since both phenomena have lasted more than 600 days, it is quite unlikely that such correlation could be due to chance only. So, while we have no true evidence to verify this, we can reasonably assume that both phenomena have been orchestrated by the same people, or at least using the same software tool and sets of compromised machines.

***MC*10: P2P aberrations** $MC10$ is a very interesting, yet intriguing, cloud. Our technique has grouped together 138 μ-events that have been observed over a period of 573 days. All these events share a number of common characteristics that we have some difficulty to explain:

1. The vast majority of these μ-events target a single platform, located in China. A very few μ-events have also hit another platform in Spain.
2. The vast majority of these μ-events originate from Italy and Spain only.

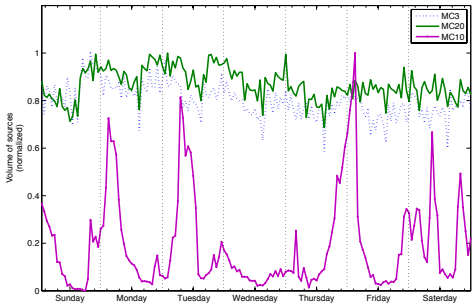


Fig. 9. Distribution of malicious sources grouped by weekdays. For each MC, a data point represents the accumulated number of sources observed for a given day and hour of the week.

3. All these μ-events exist during a single day.
4. All these μ-events target a single high TCP port number, most of them not being assigned to any particular protocol (e.g. 10589T, 15264T, 1755T, 18462T, 25618T, 29188T, 30491T, 38009T, 4152T, 46030T, 4662T, 50656T, 53842T, 6134T, 6211T, 64264T, 64783T, 6769T, 7690T)
5. these μ-events share a substantial amount of source addresses between them.
6. A number of high port numbers correspond to port numbers used by well known P2P applications (e.g., 4662/TCP, used by eDonkey P2P network).

This last remark leads us to hypothesize that this extremely weird type of attack traces may have something to do with P2P traffic aberrations. It can be a misconfiguration error or, possibly, the side effect of a deliberate attack against these P2P networks, as explained in [12, 5], in which authors argued that it is possible to use P2P networks to generate DDoS attacks against any arbitrary victim.

Also, Figure 9 highlights the fact that these 138 μ-events are not randomly distributed over the hours of the week but that, instead, they seem to exist on a limited number of recurrent moments.

All these elements tend to demonstrate the meaningfulness of grouping all these, apparently different, attack events. Even if we are not able, at this stage, to provide a convincing explanation related to their existence, our method has, at least, the merit of having highlighted the existence of these, so far, unknown phenomena.

It is our hope that other teams will build upon this fundational result to help all of us to better understand these numerous threats our approach has identified.

6 Conclusions

In this document, we have presented the WOMBAT attack attribution method. We have explained its motivations, its principles, the various steps it was made of, as well as some of the interesting results it had delivered so far. We have applied that technique to 2 years of attack traces captured on 40 low interaction honeypots located all over the world. It is worth noting that the method could as easily be applied on completely different threats-related events. In fact, the interim Symantec report published mid October 2009 on the analysis of rogue AV web sites offers results of the application of this very same method to the problem of understanding the modus operandi of malicious users setting up rogue AV campaigns.

It is our hope that people will be interested in trying to understand the rationales behind the *Misbehaving Clouds* we have identified. We are eager to share as much information as possible with such interested parties. Similarly, we are looking forward in having other opportunities to apply this method to other security datasets that future partners would be willing to share with us.

References

1. Michele Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes:Theory and Application*. Prentice Hall, 1993.
2. G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Berlin, New York, 2007.
3. M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.
4. Marc Dacier, Fabien Pouget, and Hervé Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.
5. Karim El Defrawy, Minas Gjoka, and Athina Markopoulou. Bottorrent: misusing bittorrent to launch ddos attacks. In *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.
6. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.
7. S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951.
8. C. Leita, V. H. Pham, O. Thonnard, E. Ramirez Silva, F. Pouget, E. Kirda, and M. Dacier. The leurre.com project: collecting internet threats information using a worldwide distributed honeynet. In *1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands*, Apr 2008.
9. Corrado Leita and Marc Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.
10. Leurre.com, Eurecom Honeypot Project. http://www.leurrecom.org/, [[s]ep 2009].
11. J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991.

12. Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM.
13. Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of Internet Background Radiation. In *Proceedings of the 4th ACM SIGCOMM conference on the Internet Measurement*, 2004.
14. M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
15. Van-Hau Pham. *Honeypot traces forensics by means of attack event identification.* PhD thesis, TELECOM ParisTech, 2009.
16. Van-Hau Pham and Marc Dacier. Honeypot traces forensics : the observation view point matters. In *NSS 2009, 3rd International Conference on Network and System Security, October 19-21, 2009, Gold Coast, Australia*, Dec 2009.
17. Van-Hau Pham, Marc Dacier, Guillaume Urvoy Keller, and Taoufik En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.
18. Fabien Pouget, Marc Dacier, and Hervé Debar. Honeypot-based forensics. In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference 2004*, Brisbane, Australia, May 2004.
19. Fabien Pouget, Marc Dacier, and Van Hau Pham. Leurre.com: on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference, 29-30th March 2005, Monaco*, Mar 2005.
20. Niels Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.
21. Roger N. Shepard. Multidimensional scaling, tree fitting, and clustering. *Science*, 210:390–398, 1980.
22. Olivier Thonnard and Marc Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.
23. Olivier Thonnard and Marc Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.
24. Olivier Thonnard, Wim Mees, and Marc Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *KDD'09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th - July 1st, 2009, Paris, France*, Dec 2009.
25. D. Wheeler and G. Larsen. Techniques for Cyber Attack Attribution. *Institute for Defense Analyses, Oct 2003*, 2008.
26. Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.
27. Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005.