SEVENTH FRAMEWORK PROGRAMME
Theme ICT-1-1.4 (Secure, dependable and trusted infrastructures)

WORLDWIDE OBSERVATORY OF
MALICIOUS BEHAVIORS AND ATTACK THREATS

D22 (D5.2) Root Causes Analysis: Experimental Report

Contract No. FP7-ICT-216026-WOMBAT

Workpackage	WP5 - Threats Intelligence
Author	Olivier Thonnard
Version	1.0
Date of delivery	M40
Actual Date of Delivery	M40
Dissemination level	Public
Responsible	SYMANTEC
Data included from	EURECOM, HISPASEC, TUV

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°216026.

SEVENTH FRAMEWORK PROGRAMME

Theme ICT-1-1.4 (Secure, dependable and trusted infrastructures)

The WOMBAT Consortium consists of:

France Telecom	Project coordinator	France
Institut Eurecom		France
Technical University Vienna		Austria
Politecnico di Milano		Italy
Vrije Universiteit Amsterdam		The Netherlands
Foundation for Research and Technology		Greece
Hispasec		Spain
Research and Academic Computer Network		Poland
Symantec Ltd.		Ireland
Institute for Infocomm Research		Singapore

Contact information:
Dr Marc Dacier
2229 Routes des Cretes
06560 Sophia Antipolis
France

e-mail: Marc_Dacier@symantec.com
Phone: +33 4 93 00 82 17

Contents

Abstract

This deliverable offers an extensive report of all experiments carried out with respect to *root cause analysis* techniques. This final deliverable for Workpackage 5 (Threats Intelligence) builds upon D12 (D5.1 - Technical Survey on Root Cause Analysis) and benefits from the modifications made to the various software modules developed in WP4, following up the experimental feedback.

The R&D efforts carried out in WP5 with respect to root cause analysis have produced a novel framework for *attack attribution* called TRIAGE. This framework has been successfully applied to various WOMBAT datasets to perform intelligence analyses by taking advantage of several structural and contextual features of the data sets developed by the different partners. These experiments enabled us to get insights into the underlying root phenomena that have likely caused many security events observed by sensors deployed by WOMBAT partners.

In this deliverable, we provide an in-depth description of experimental results obtained with TRIAGE, in particular with respect to *(i)* the analysis of Rogue AV campaigns (based on HARMUR data), and *(ii)* the analysis of different malware variants attributed to the *Allaple* malware family (based on data from SGNET, VirusTotal and Anubis).

Finally, we describe another experiment performed on a large spam data set obtained from *Symantec.Cloud* (formerly *MessageLabs*), for which TRIAGE was successfully used to analyze spam botnets and their ecosystem, i.e., how those botnets are used by spammers to organize and coordinate their spam campaigns. Thanks to this application, we are considering a possible technology transfer of TRIAGE to *Symantec.Cloud*, who is interested in carrying out regular intelligence analyses of their spam data sets, and may also consider the integration of TRIAGE to their *Skeptic*® spam filtering technology.

1 Introduction

Understanding the existing and emerging threats on the Internet should help us to effectively protect the Internet economy, our information systems and the net citizens. This assertion may look blindingly obvious to many people. However, things are less evident when looking more closely at the problem. Among security experts, there is at least one thing on which everybody agrees: combating cyber-crime becomes harder and harder [63, 17, 61]. Recent threat reports published by major security companies have also acknowledged the fact that the cyber-crime scene is becoming increasingly more organized, and more consolidated [71, 72, 35, 49].

Since 2003, there seems to be a shift in the nature of attacks in the Internet, from server-side to client-side attacks and from fast spreading worms to profit-oriented activities like identity theft, fraud, spam, phishing, online gambling, extortion. Most of those illegal activities are supported by large *botnets* controlled by criminal organizations. All facts and figures presented in public security threat reports are certainly valuable and help to shed some light on those cyber-criminal phenomena, but a lot of unknowns remain.

In fact, current analysis tools do not allow us to automatically perform *intelligence analysis* on attack phenomena, even less from a strategic viewpoint. Even though there are some plausible indicators about the origins, causes, and consequences of malicious activities, very few claims can be backed up by scientific evidence. The main reason is that no global threat analysis framework exists to rigorously investigate emerging attack phenomena using different *viewpoints* (e.g., different data sources), together with effective *aggregation* methods that would enable an analyst to combine many different attack features in an appropriate way.

1.1 Attack Attribution

Many open issues remain regarding the *root causes* and the attribution of most security events observed or collected by various means. For example, who is really behind the observed attacks, i.e., how many organizations are responsible for them? Where do they originate? How many groups control the largest botnets used for sending spam? What are the emerging strategies used in cyber-crime? Which "rogue networks" [69] are used

as bullet-proof hosting (e.g., RBN[1], Atrivo a.k.a. Intercage, McColo, or 3FN, and maybe some others), but more importantly, how do they evolve over time ? Are botnets able to coordinate their actions?

As another example, we observe a growing number of malware samples of various types spreading all over the Internet, sometimes at a very high pace. Some WOMBAT partners, such as VirusTotal and Symantec, receive hundreds of thousands of seemingly unique malware samples per week. Figuring out which groups of malware samples are likely due to the same criminal organization, or could be linked to the same root phenomenon, is a daunting task.

To succeed, defenders need to have at their disposal efficient techniques that *prioritize* security events, and highlight the ones they should first look at, depending on their likely impact. Security analysts must have tools and techniques to help them characterize the threats and produce countermeasures in an automated way, as much as possible.

1.2 TRIAGE: Towards Automated Intelligence Analysis

All previously described issues are related to a common security problem often referred to as *attack attribution*, i.e., how to attribute (potentially) different attacks to a common root cause, based on the combination of all available evidence.

In WOMBAT deliverable D12 (D5.1), we have provided an extensive survey of root cause analysis technique. Recall that, by *root cause*, we do not refer to the identification of a given machine that has launched one specific, isolated attack (i.e., we are not interested in what is sometimes called *IP traceback*). Instead, we are more interested in having a better idea of the various individuals, groups or communities (of machines) that are responsible for large-scale attack phenomena. Remember also that the ultimate goal of this WorkPackage (WP5 - Threats Intelligence) is not to offer names of individuals to law enforcement agencies. The goal is, instead, to provide models of the acting entities that we are facing. However, through generalization, these models can help in understanding the threats that every person or organization who connects to the Internet currently faces.

As a result, the R&D efforts carried out by the WOMBAT consortium in WorkPackage 5 (WP5 - Threats Intelligence) have produced a novel software framework, called TRIAGE [75], a multi-criteria analysis framework that supports *attribution* and root cause analysis of security threats. This framework has been applied to various WOMBAT

[1]The Russian Business Network (RBN) is a multi-faceted cybercrime organization, which is notorious for its hosting of illegal and dubious businesses such as phishing, spam, child pornography and malware distribution `http://www.bizeul.org/files/RBN_study.pdf`.

datasets to perform intelligence analyses, by taking advantage of several structural and contextual features of the data sets developed by consortium partners in the context of WP3 and WP4.

As demonstrated by the experimental results of this deliverable, TRIAGE has enabled us to get new insights into the underlying root phenomena that have likely caused many security events observed by sensors deployed by WOMBAT partners. Throughout experiments performed on different datasets, we illustrate how this framework enables a precise analysis of the *modus operandi* of the attackers, and thus can help an analyst to get a better understanding of how cybercriminals operate in the real-world, as well as the strategies they are using.

1.3 Structure of the Document

The rest of this document is organized as follows. To make this deliverable as self-contained as possible, Chapter 2 introduces the TRIAGE software framework and describes formally the various techniques we have implemented in TRIAGE components.

The next chapters provide an in-depth description of some experimental results obtained with TRIAGE: Chapter 3 describes an analysis of *Rogue AV campaigns* (based on HARMUR data), and Chapter 4 provides an in-depth analysis of malware variants propagating like the *Allaple* malware family (based on data from SGNET, VirusTotal and Anubis).

Chapter 5 describes another experiment performed on a large spam data set obtained from *Symantec.Cloud* (formerly *MessageLabs*), for which TRIAGE was successfully used to analyze spam campaigns and the spam botnets ecosystem. It is worth mentioning that we are considering a possible technology transfer of TRIAGE to *Symantec.Cloud*, who is interested in carrying out regular intelligence analyses of their spam data sets, and may also consider the integration of TRIAGE to their *Skeptic*® spam filtering technology.

Finally, we conclude this deliverable in Chapter 6 where we give some interesting perspectives on how to further improve the TRIAGE framework.

2 TRIAGE: A Framework for Attack Attribution

2.1 Introduction

This Chapter introduces TRIAGE, a generic and multi-criteria software analysis framework that has been developed in WOMBAT to address, in a rigorous and systematic way, the *attack attribution* problem. Such a framework must enable us to systematically discover, extract and combine patterns from a security dataset, according to a set of potentially useful characteristics, and with limited knowledge on the phenomena under scrutiny. The underlying analytical methods must be sufficiently generic so that it can be applied to virtually any type of dataset comprising security events (e.g., attack events observed by honeypots, network attack events, IDS alerts, malware samples, spam messages, etc.).

By applying TRIAGE to various security datasets, our objective is to identify *attack phenomena* occurring at a larger scale, but also understand their *root cause*, and get insights into the *modus operandi* of attackers.

TRIAGE stands for *attribution of attacks using graph-based event analysis*[1]. It relies on a novel combination of graph-based analysis with a data fusion process inspired by Multi-Criteria Decision Analysis (MCDA). As illustrated in Fig. 2.1 , TRIAGE is based on three components:

1. **Attack feature selection**: we determine which (potentially) relevant *attack features* we want to include in the overall analysis, and we characterize each element (i.e., each *security event*) of the data set according to this set of selected features, denoted by $\mathcal{F} = \{F_k\}$, $k = 1, \ldots, n$ (e.g., by creating feature vectors for each element);

2. **Graph-based clustering**: an undirected edge-weighted graph is created with respect to every feature F_k, based on an appropriate distance for measuring pairwise similarities. As an additional step, a graph analysis can be performed on a single feature basis, to identify strongly connected components within each graph.

[1] In the medical domain, the term "triage" has a specific meaning, i.e., it refers to the process of prioritizing patients based on the severity of their condition.

This step can help reveal the structure of the data set and the various relationships among groups of security events that have strong correlations w.r.t. a given feature;

3. **Multi-criteria aggregation**: this data fusion step combines the different weighted graphs using an *aggregation function* that models the expected behavior of the phenomena under study.

As output of TRIAGE, we obtain Multi-Dimensional Clusters (or MDC's) which group security events likely due to the same *root cause*, because all events within an MDC are linked by a certain number of common features as defined by the analyst. Note also that the approach is mostly unsupervised, i.e., it does not rely on a preliminary training phase.

At this stage, it is already important to stress that, in contrast to all classical clustering techniques, the MCDA approach enables us to model more complex behaviors regarding the way security events are linked together. For example, the precise set (or combination) of features needed to link security events together does not have to be specified in advance. As a result, an MDC can perfectly be made of different subsets of events that are characterized by different *combinations of features*, but all subsets of events are still somehow interconnected by a minimum amount of features. As described more in detail in Section, multi-criteria decision-making techniques provide a more flexible way of combining attack features, allowing one to include more complex decision-making constraints such as "at least 3 (out of n) features are required to attribute events to the same phenomenon, but features F_i and F_j are not really independent and are somehow redundant. On the other hand, feature F_k is twice as important as F_i, and slightly more important than F_j".

In the next Sections, we further describe each TRIAGE component.

2.2 Attack Feature Selection

In data mining, one of the very first steps consists in selecting some key characteristics from the data set, i.e., salient features that may reveal interesting *patterns*. As described by [33], typical clustering activities involve following steps:

(i) feature selection and/or extraction;

(ii) definition of an appropriate distance for measuring similarities between pairs of feature vectors;

(iii) applying a grouping algorithm;

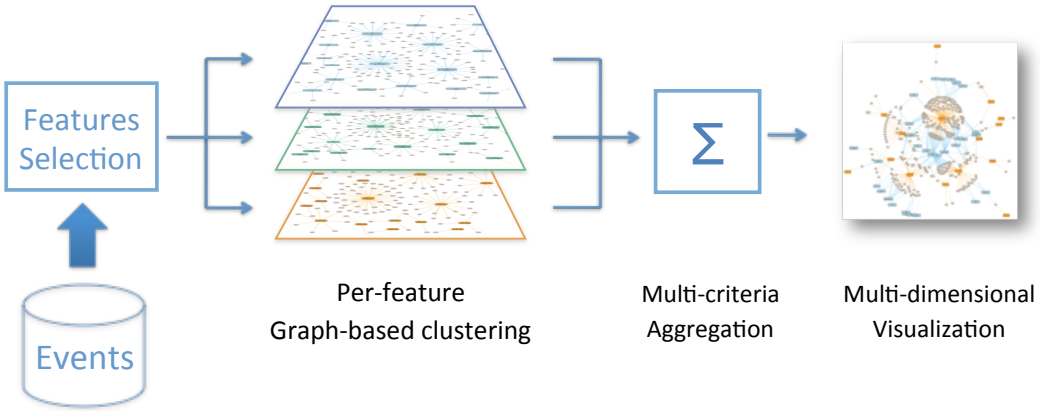


Figure 2.1: Overview of the TRIAGE attack attribution method, in which multiple weighted graphs are aggregated into a combined graph using a multi-criteria decision analysis (MCDA) approach. As output, TRIAGE provides Multi-Dimensional Clusters (or MDC's) that group security events likely due to the same root cause. By using appropriate visualizations, MDC's can then help emphasize the *modus operandi* of attackers.

(iv) data abstraction (if needed), to provide a compact representation of each cluster;

(iv) assessment of the clusters quality and coherence (also optional), e.g., by means of validity indices.

Feature selection is the process of identifying, within the raw data set, the most effective subset of characteristics to use in clustering. The selection of these features may optionally be completed by a *feature extraction* process, i.e., one or more transformations of the input to produce features that are more suited to subsequent processing. Pattern representation refers to the number of categories, classes, or variables available for each feature to be used by the clustering algorithm.

More formally, we have thus a data set $\mathcal{D}$ composed of m objects, which are usually defined as security *events*. We define a feature set $\mathcal{F}$ made of n different features F_k, $k = 1, \ldots, n$, that can be extracted for each event e_i from $\mathcal{D}$ $(i = 1, \ldots, m)$.

Let us denote by $\mathbf{x}_i^{(k)}$ the feature vector extracted for the event e_i w.r.t. feature F_k. In fact, $\mathbf{x}_i^{(k)} \in \mathbb{R}^d$ is a d-dimensional vector of real values, i.e.:

$$\mathbf{x}_i^{(k)} = \{x_{i,1}^{(k)}, \ldots, x_{i,d}^{(k)}\}$$

where d is the dimension of the vector and is a function of the feature F_k.

Finally, we can group all feature vectors defined w.r.t. a given feature into a data set $\mathbf{X}_k = \{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_m^{(k)}\}$. In many data mining books, it is customary to use a matrix notation to represent a set of feature vectors $\mathbf{X}_k$, i.e.:

$$\mathbf{X}_k = \begin{bmatrix} x_{1,1}^{(k)} & x_{1,2}^{(k)} & \cdots & x_{1,d}^{(k)} \\ x_{2,1}^{(k)} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_{m,1}^{(k)} & \cdots & \cdots & x_{m,d}^{(k)} \end{bmatrix}$$

where the i^{th} row represents the feature vector $\mathbf{x}_i^{(k)}$ extracted for the event e_i of $\mathcal{D}$, obtained for the k^{th} feature F_k.

Summarizing, our problem space is made of three dimensions: m is the number of security events, n is the number of attack features, and d is the dimensionality of the feature vector (the latter is variable and is a function of each considered feature F_k).

Illustrative example

A typical example of feature vector $\mathbf{x}_i^{(k)}$ can be the geographical distribution of attacking machines for a given security event composed of several similar probes observed on a sensor at a given point in time (e.g., honeypot, IDS, etc). In this case, for each event e_i we may want to create a feature vector $\mathbf{x}_i^{(geo)}$ made of $d = 229$ positions corresponding to all countries (ordered alphabetically) where attackers apparently reside. In other words,

$$\mathbf{x}_i^{(geo)} = \{x_{i,1}^{(geo)}, \ldots, x_{i,229}^{(geo)}\}$$

where every element of the vector $x_{i,j}^{(geo)}$ represents the number of attackers observed for country j. Another standard representation for this type of feature vector would be under the form of relative frequencies, e.g.: US(35%), CN(7%), DE(5%), CA(5%), others(47%).

Quite obviously, the geographical origin of attackers is only one possible feature that may be useful in root cause analysis. Depending on the type of data set and phenomenon being studied, the analyst may want to include many other features, such as network-related features (IP address, the /24, /16, or /8 subnets of the address, the ISP or the ASN, etc), DNS or Whois-related information, malware features (MD5, PE header, and behavioral features), timing information (day-hour of the observations), or other application-specific features (e.g., subject lines, embedded URI's and From-domains used in spam messages).

2.3 Single-Feature Graph Clustering

Cluster analysis aims at finding natural groupings from a data set and is essentially an approach relying on *exploratory data analysis* (EDA) [80]. Regarding this exploratory aspect, *clustering* refers to a process of *unsupervised classification* by which unlabeled patterns are grouped into clusters based on a measure of similarity. As a result, all patterns found in a "valid" cluster should be more similar to each other than they are to patterns of another cluster. The goal of clustering consists in discovering interesting and meaningful patterns from a data set, without any prior knowledge on the phenomena being studied. For attack attribution purposes, this can be helpful to understand the underlying phenomena that may have created the security events observed by any type of sensor. Clusters can also help provide a data abstraction level, since every cluster can be described by a prototype that is representative of all attack patterns being grouped in a particular cluster.

There exists a plethora of clustering algorithms, which can be roughly categorized as either *partitional* or *hierarchical*. Partitional techniques aim at finding the most effective partition of the data set by optimizing a clustering criterion (e.g., minimizing the sum of squared distances within each cluster). Hierarchical clustering methods produce a nested series of partitions (i.e., a hierarchical tree-like structure) in which the decision to merge objects or clusters at each level is performed based on a similarity criterion and a linkage method (e.g., the smallest or largest distance between objects).

However, since clustering is mostly a *data-driven* process, it can be hard sometimes to define what really constitutes a cluster, as underlined by several authors (e.g., [34, 48, 74]). Indeed, most clustering techniques rely on several input parameters which can largely influence the results. Furthermore, some algorithms assume some sort of structure for the clusters (e.g., spherical, elliptical, density-based, etc). Thus, if they are given a certain data set, most clustering algorithms will find clusters, regardless of whether they are really present in the data or not.

As illustrated in [75], clustering real-world data sets can thus be a difficult task, and different clustering methods will quite probably yield different results. For this reason, the TRIAGE attribution framework is not limited to a given clustering algorithm. Regarding this component, the only requirement of our method is to use a graph-based representation (i.e., *edge-weighted graphs*) in which all pairwise distances are calculated ahead of time for every attack feature. We are aware of the fact that this pairwise approach can be computationally intensive for very large data sets, especially regarding memory requirements. However, we argue that the computation of those pairwise similarities can be easily parallelized since all computations are independent of each other. Many database systems can even provide support for storing and indexing structures

like distance matrices.

Identifying clusters within each edge-weighted graph can provide interesting *viewpoints* on the attack events being analyzed. Even though this clustering step is not really mandatory in the overall attribution method, it can still help to find interesting patterns among groups of security events that share common features. Furthermore, this step can help the analyst to decide on which features to include in the multi-criteria fusion step.

To perform the clustering, we have opted for a novel graph-theoretical algorithm that extracts *dominant sets* from the graph, which is further detailed in Section 2.3.1. Our choice is motivated by the following reasons [75]:

- the simplicity to formulate the problem, i.e., a graph can be easily represented by its edge-weighted adjacency matrix (or proximity matrix);
- the *dominant sets* algorithm does not require a number of clusters as input, and can naturally extract the most significant groups (i.e., *cliques*) in the first stages of the algorithm execution;
- finding dominant sets (or *cliques*) in a graph can be formulated as a straightforward continuous optimization technique. This is interesting as it can be coded in a few lines of any high-level programming language, and could be easily implemented in a parallel network, should scalability become an issue;
- as it will become clear in the next Section, multiple edge-weighted graphs (as obtained for different attack features) can be easily combined using MCDA aggregation functions that model the behavior of phenomena under scrutiny.

Quite obviously, other classical clustering algorithms, such as K-Means, Hierarchical Clustering (single or complete-linkage), Connected Components, etc, can still be used within the framework, if necessary.

2.3.1 Dominant sets clustering

For each attack feature F_k, we build an edge-weighted graph G_k in which the vertices (or nodes) are mapped to the feature vectors $\mathbf{x}_i^{(k)}$, and the edges (or links) reflect the similarity between data objects regarding the considered feature. As customary, we can represent the undirected edge-weighted graph (with no self-loops) obtained for a given feature F_k by:

$$G_k = (V_k, E_k, \omega_k) \tag{2.1}$$

$$\text{where} \begin{cases} V_k = \{\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}, \ldots, \mathbf{x}_m^{(k)}\} & \text{is the vertex set} \\ E_k \subseteq V_k \times V_k & \text{is the edge set (i.e., relations among vertices)} \\ \omega_k : E_k \to \Re^+ & \text{is a positive weight function} \end{cases}$$

In practice, we can represent each graph G_k with its corresponding weighted *adjacency matrix* (or dissimilarity matrix), which is the $m \times m$ symmetric matrix $A_k(i,j)$ defined as:

$$A_k(i,j) = \begin{cases} \omega_k(i,j), & \forall (i,j) \in E_k \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that the weight function $\omega_k(i,j)$ must be defined with a similarity metric that is appropriate to the nature of the feature vector under consideration, as explained here after in Section 2.3.2.

In graph theory, many clustering algorithms consist of searching for certain combinatorial structures, such as a minimum spanning tree [86] or a minimum cut [66, 83]. Among these methods, a classic approach to clustering reduces to a search for complete subgraphs, which is also known as the "complete-link" algorithm. Indeed, the maximal complete subgraph, also called a *(maximal) clique*, was considered the strictest definition of a cluster in [2] and [59].

The concept of a maximal clique was originally defined on unweighed graphs; however, it has been generalized to the edge-weighted case by Pavan et al. who proposed in [58] a new framework for pairwise clustering based on *dominant sets*. The formal properties of dominant sets make them reasonable candidates for a new formal definition of a cluster in the context of edge-weighted graphs. Furthermore, Pavan et al. established a correspondence between dominant sets and the extrema of a continuous quadratic form over the standard simplex. This means that we can find dominant sets (or clusters) using straightforward continuous optimization techniques such as *replicator equations*, a class of dynamical systems arising in evolutionary game theory. Such systems are interesting since they can be coded in a few lines of any high-level programming language.

The method introduced by Pavan consists of iteratively finding dominant sets in an edge-weighted graph, and then removing it from the graph until all vertices have been clustered (complete partitioning), or as soon as a given *stopping criterion* is met, which could give eventually an incomplete partition as output. Some examples of constraints used as stopping criterion are: (i) a minimum threshold for the remaining nodes within the graph; (ii) a lower threshold (absolute or relative) on the sum of all remaining edge-weights. Thus, let $W_{origin} = \sum a_{ij}$ be the sum of all weights in the original graph, then the procedure could stop when the sum of all remaining edge-weights is less than $(0.01 \cdot W_{origin})$.

Algorithm 2.1 Dominant sets Clustering

Input: weighted graph $G = (V, E, \omega)$
Output: a partition $\mathcal{P}$ (eventually incomplete)
  $\mathcal{P} = \emptyset$
  while $STOPPING\_CRITERION(G)$ **do**
    $S \leftarrow DOMINANT\_SET(G)$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{S\}$
    $V \leftarrow V \setminus S$
  end while
  return $\mathcal{P}$

The dominant sets clustering algorithm is described in the pseudo-code given in algorithm 2.1. As one can see, the cornerstone of this algorithm is the procedure DOMINANT_SET, which boils down to making a particular temporal expression converge. More precisely, consider the following dynamical system represented with its discrete time equation, where A_k is the adjacency matrix of an edge-weighted graph G_k:

$$x_i(t+1) = x_i(t) \cdot \frac{(A_k \, \mathbf{x}(t))_i}{\mathbf{x}(t)^T A_k \, \mathbf{x}(t)} \tag{2.3}$$

with $i = 1, \ldots, m$. Starting from an arbitrary initial state, this dynamical system will eventually be attracted by the nearest asymptotically stable point. Thus, the procedure DOMINANT_SET simply involves the simulation of the system given in equation 2.3. The solution to this dynamical system is a stable point, called a characteristic vector $\mathbf{x}^S$, which corresponds to a dominant set (as it has been proved in [57]).

We refer the interested reader to [75] for a more detailed discussion and an objective comparison of various clustering algorithms against dominant sets.

2.3.2 Similarity metrics

Most clustering algorithms rely on certain distance metrics to group objects into clusters. A *similarity metric* is a function that indicates how alike objects are to each other. However, it is quite common to calculate instead the *dissimilarity* between two patterns (which is just the opposite) using a distance metric defined on the feature space.

As mentioned here above (definition 2.2), the weight function $\omega_k(i, j)$ must be defined with respect to a distance metric appropriate to the type of feature vector. Clearly, the choice of this distance metric is fundamental, since it has an impact on the properties of the final clusters, such as their size, quality, and consistency.

Probably one of the most commonly used distance measures is the Euclidean distance, which is in fact a special case of the Minkowski metric (with $p = 2$):

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = (\sum_{k=1}^{d} |x_{i,k} - x_{k,j}|^p)^{\frac{1}{p}} \tag{2.4}$$

As observed in [46], Euclidean metrics work well when the data set contains compact or isolated clusters, but the drawback of these metrics is their sensitivity to the scale of the features. This problem can be alleviated with the normalization of the vectors. However, Euclidean distances suffer from other drawbacks, e.g., they can be completely inappropriate with high-dimensional data. This problem is known as the *curse of dimensionality*[2], which is caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space. In fact, several previous works have showed that in high-dimensional space, the concept of proximity, distance or nearest neighbor may not even be qualitatively meaningful when relying on commonly used metrics such as L_k norms, especially in data mining applications [1].

Another common similarity measure that can be used with real-valued vectors is the *sample correlation* between observations treated as sequences of values:

$$d_{corr}(\mathbf{x}_i, \mathbf{x}_j) = \frac{(\mathbf{x}_i - \overline{\mathbf{x}})^T(\mathbf{x}_j - \overline{\mathbf{x}})}{\sqrt{(\mathbf{x}_i - \overline{\mathbf{x}})^T(\mathbf{x}_i - \overline{\mathbf{x}})}\sqrt{(\mathbf{x}_j - \overline{\mathbf{x}})^T(\mathbf{x}_j - \overline{\mathbf{x}})}} \tag{2.5}$$

where $\overline{\mathbf{x}}$ represents $(\mathbf{x}_i + \mathbf{x}_j)/2$.

The sample correlation (also called the *Pearson coefficient*) reflects the strength of the linear dependence between two real-valued vectors, which can also be viewed as a similarity degree between the "shapes" of the vectors. It is thus directly linked to the covariance of the vectors. A correlation value of 1 implies a perfect linear relationship between the two vectors (as x_i increases, x_j increases proportionally). A closely related similarity measure is the *cosine similarity* obtained by computing the cosine of the angle formed by the two vectors, which is commonly used for clustering document data in text mining [74].

The interpretation of a correlation coefficient depends on the context and purposes; however, a value between 0.5 and 1 is usually considered as an indication of a strong dependence between observations.

Finally, to measure distances between probability distributions (e.g., histograms), there exist specific statistical distances that are more appropriate. One such technique

[2]This term was first coined by Richard Bellman in 1957 [11]

(which is commonly used in information theory) is the Kullback-Leibler divergence ([38]). Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be for instance two feature vectors that represent two probability distributions over a discrete space X, then the K-L divergence of $\mathbf{x}_j$ from $\mathbf{x}_i$ is defined as:

$$D_{KL}(\mathbf{x}_i||\mathbf{x}_j) = \sum_{k=1}^{d} \mathbf{x}_i(k) \log \frac{\mathbf{x}_i(k)}{\mathbf{x}_j(k)}$$

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 ([44]), defined as:

$$D_{JS}(\mathbf{x}_i, \mathbf{x}_j) = \frac{D_{KL}(\mathbf{x}_i||\bar{\mathbf{x}}) + D_{KL}(\mathbf{x}_j||\bar{\mathbf{x}})}{2} \tag{2.6}$$

where $\bar{\mathbf{x}} = (\mathbf{x}_i + \mathbf{x}_j)/2$. In other words, the Jensen-Shannon divergence is the *average* of the KL-divergences to the *average distribution*. To be a true metric, the JS divergence must also satisfy the triangular inequality, which is not true for all cases of $(\mathbf{x}_i, \mathbf{x}_j)$. Nevertheless, it can be demonstrated that the *square root* of the Jensen-Shannon divergence is a true metric ([25]).

An alternative metric for measuring the similarity of two discrete probability distributions is the *Bhattacharyya* distance ([12]), a quite popular metric used in the signal processing community. It gives an approximate measurement of the amount of overlap between two frequency vectors.

Transforming distances to similarities.

To transform pairwise distances d_{ij} to similarity weights s_{ij}, we can use different mapping functions. Some of the most commonly used functions are:

$$s_{ij} = \begin{cases} 1 - d_{ij} \\ c - d_{ij}, \text{ for some constant } c \\ (1 + d_{ij})^{-1} \end{cases}$$

However, previous studies found that the similarity between stimuli decay exponentially with some power of the perceptual measure distance ([65]). So, it is also customary to use the following function to do this transformation:

$$s_{ij} = exp(\frac{-{d_{ij}}^2}{\sigma^2}) \tag{2.7}$$

where σ is a positive real number which affects the decreasing rate of s. In [75], we have proposed an empirical method to determine appropriate ranges of values for σ according to the statistical properties of the data set, and the expected similarities.

Several authors have also extensively studied problems related to proximity measures, and how to choose them in a consistent manner based on the clustering problem and the features at hand (see for instance [23, 6, 36]).

2.3.3 Cluster validity indices

Various cluster validity indices have been proposed in previous work to assess the quality and the consistency of clustering results [32]. In graph clustering, most indices are based on the comparison of intra-cluster connectivity (i.e., the compactness of clusters) and the inter-cluster variability (i.e., the separability between clusters). In [13], the authors provide a nice review of different validity indices that are particularly appropriate for evaluating graph clustered structures.

In [75] , we have defined certain validity indices that are well-suited for evaluating the quality of dominant sets found by TRIAGE, but also the clustering results obtained via more classical methods. In particular, we have used three different validity indices:

- the Graph compactness, which indicates how compact the clusters are;
- the Davies-Bouldin index, which evaluates the inter versus intra-cluster connectivity;
- the Silhouette index, which is linked to the characteristics of nodes' neighborhood.

To assess the quality of the experimental results presented in this deliverable, we will mainly focus on the graph compactness, which is a quite effective, yet easy to compute, validity index. We refer the interested reader to [13, 75] for more information on the other cluster validity indices.

Graph compactness

The *graph compactness* C_p is a validity index that is very easy to calculate, and which can be helpful to evaluate graph clustering results. C_p is mainly based on the characteristics of graphs connectivity. That is, for any cluster C_k, we can calculate a normalized compactness index:

$$\mathrm{C}_{p_k} = \frac{\sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} \omega(i,j)}{N_k(N_k-1)/2} \tag{2.8}$$

where $\omega(i, j)$ is the positive weight function reflecting node similarities in the graph, and N_k is the number of members within cluster k. Since C_{p_k} only depends upon similarity values and the composition of the clusters, it can also be used to evaluate many other clustering methods.

We can also define an average compactness index $\overline{\mathrm{C}_p}$ for a partition made of K clusters, which takes into account the individual compactness values of the clusters, but also their respective sizes:

$$\overline{\mathrm{C}_p} = \frac{\sum_{k=1}^{K} \mathrm{C}_{p_k} N_k}{\sum_{j=1}^{K} N_j} \tag{2.9}$$

2.4 MCDA-based Attack Attribution

The graph-based clustering component (based on dominant sets) can be a useful and effective way of extracting informative patterns from a set of security events. By repeating this process for different attack features, we obtain one set of clusters for each attack feature, which provide interesting viewpoints on the underlying phenomena.

However, similarly to criminal forensics, a security analyst often needs to synthesize different pieces of evidence in order to investigate the root causes of attack phenomena. This can be a tedious, lengthy and informal process mostly relying on the analysts expertise. A somehow naïve approach of doing this aggregation of features consists in performing *cross-feature analysis* by computing intersections among all clusters obtained for each feature separately. Even though it could work in fairly simple cases, we observed that this approach does not hold for many attack phenomena we have analyzed so far. The reasons are twofold:

(i) the *uncertainty* problem, which is due to the *fuzzy* aspect of real-world phenomena. Measuring similarities can be intrinsically hard for certain characteristics, and provides usually a continuous value in the interval $[0, 1]$ that reflects a certain degree of correlation between two feature vectors. Based on a single value, it is usually quite difficult to decide whether or not two attack events might be linked to the same root cause.

(ii) the *dynamicity* problem, which is due to the evolutive nature of real-world phenomena. As certain characteristics of attack phenomena may evolve (e.g., the origins of a botnet may change over time, polymorphic malware try to randomize certain aspects of their code, spammers tend to use disposable URI's or "From domains" for their spam campaigns, etc), it can be difficult to define *a priori* which set of features must be used to link events to the same phenomenon.

2.4.1 Multi-criteria Decision Analysis

As suggested here above, the problem of combining attack features looks quite similar to typical situations handled in *multi-criteria decision analysis* (MCDA), also called sometimes *Multi-Attribute Utility Theory* (MAUT). In a classical MCDA problem, a decision-maker evaluates a set of *alternatives* w.r.t. different *criteria*, and a global score is then calculated for each alternative using a well-defined *aggregation method* that models the preferences of the decision-maker or a set of constraints.

Generally speaking, the alternatives are evaluated w.r.t different attributes (or features) that are expressed with numerical values representing a degree of preference, or a degree of membership[3]. The two most common aggregation methods used in MAUT are the weighted arithmetic and geometric means.

Another related application is the *group decision-making*, where n experts express their evaluations on one (or more) alternatives. The goal is then to combine all experts' preferences into a single score (like in many sports competitions, where the criteria are scores given by different judges). The most commonly-used technique for combining experts' scores is the weighted arithmetic mean, since experts may be assigned different weights according to their standing.

Yet other typical examples involving an aggregation process can be found in *fuzzy logic* and *rule-based systems*. In this case, the inference engine is made of fuzzy rules in which the rule antecedents model attributes that are subject to vagueness or uncertainty. The aim is to evaluate the "firing strength" of each fuzzy rule using logical connectives, and then to combine all rules' output into a single, crisp value that can be used to make a decision.

2.4.2 Formalizing the aggregation problem

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 (or alternative), and this combined output forms then a basis from which decisions can be made.

Definition 2.1. (Aggregation function [10]) *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: $f_{aggr} : [0,1]^n \longrightarrow [0,1]$, with the following properties:*

[3] Obviously, some attributes can sometimes be expressed using ordinal or qualitative values. Then, a commonly-used approach consists in converting ordinal values to a numerical scale using utility functions.

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

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

All unit intervals $[0,1]$ *are considered here to be continuous, i.e., a variable defined on this unit interval may take any real value between the lower and upper bounds.*

In our multi-criteria attribution method, we have n different attack features F_k, whose indices can be put into a set $\mathcal{N} = \{1,2,\ldots,n\}$. For each F_k, recall that we have built an edge-weighted graph $G_k = (V_k, E_k, \omega_k)$, represented by its corresponding similarity matrix $A_k(i,j) = \omega_k(i,j)$, with ω_k defined as an appropriate distance function.

For every pair of events (i,j) of the security data set $\mathcal{D}$, a *vector of criteria* $\mathbf{z}_{ij} \in [0,1]^n$ can be constructed from the similarity matrices (i.e., from the set of graphs G_k):

$$\mathbf{z}_{ij} = [A_1(i,j), A_2(i,j), \ldots, A_n(i,j)] \tag{2.10}$$

Informally, our approach consists in combining these n values of each criteria vector $\mathbf{z}_{ij}$ which reflects the set of all relationships between a pair of security events. As illustrated in Fig.2.2, the multicriteria aggregation creates a combined graph $G^* = \sum G_k$, represented by its adjacency matrix A^*, which is obtained by applying an aggregation function f_{aggr}:

$$A^*(i,j) = f_{aggr}(\mathbf{z}_{ij}),\ \forall (i,j) \in \mathcal{D} \tag{2.11}$$

Finally, we can extract strongly *connected components* from G^* to identify all subgraphs in which any two vertices are connected to each other by a certain path:

$$\begin{aligned} \mathcal{P} &= components(A^*) \\ &= \{P_1, P_2, \ldots, P_m\} \end{aligned}$$

which gives a set of connected subgraphs $\mathcal{P}$, where $P_x \subseteq G^*$, and $\forall (i,j) \in P_x$: $f_{aggr}(\mathbf{z}_{ij}) \geq \epsilon$, with $\epsilon \in]0,1]$. Since the events of a subgraph can form different clusters with respect to individual features (or clustering dimensions), we have called these subgraphs *multi-dimensional clusters* (or MDC's).

Note that this final step turns out to be also a graph clustering problem very similar to the per-feature clustering problem described in Section 2.3. As a result, a set of MDC's (or subgraphs) $\mathcal{P}$ could be obtained by applying the very same *dominant sets* algorithm

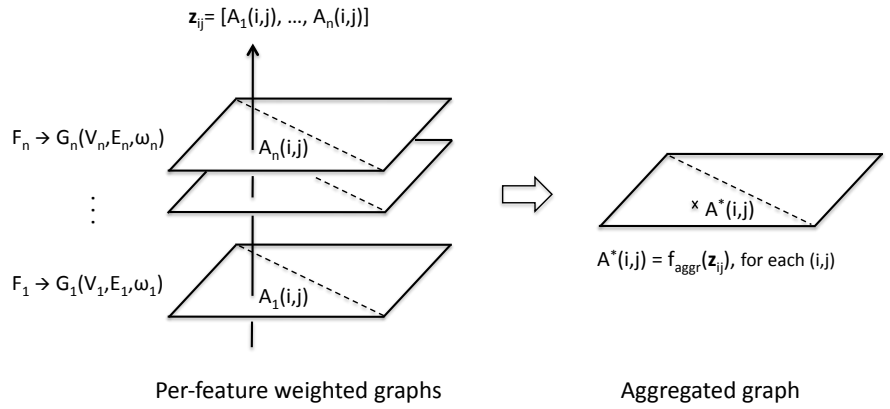


Figure 2.2: The MCDA aggregation process is performed on n edge-weighted graphs (represented by their respective proximity matrix), which leads to the construction of a combined graph G^* that takes into account the *semantics* of the aggregation function f_{aggr}.

given in Section 2.3.1.

However, searching for dominant sets in the aggregated graph may be too restrictive in certain cases, since dominant sets provide very coherent groups. The attack events within an MDC can be linked by different *combinations of features*, which means that an attack event observed at instant t_0 can have very different characteristics from another event observed at a later point in time.

As a consequence of this evolving behavior, clusters in the combined graph G^* can present elongated shapes in which attack events are linked in a sort of *chaining structure*. While this "chaining effect" is usually not desirable in single-feature clustering, it is usually required in the case of the combined graph G^* resulting from the aggregation of multiple features.

As shown in the next Chapters, by analyzing and visualizing MDC's through their individual and common features, an analyst can immediately get a global picture of all important relationships among security events and how these are interconnected, and hence he also gets a better insight into the behavior of the underlying phenomena.

2.4.3 Choice of an aggregation function f_{aggr}

It is quite evident that the choice of the aggregation function f_{aggr} used to combine attack features is fundamental, as this function will model the behavior of the phe-

nomenon under study. In other words, this choice must be guided by the *semantics* that the analyst wants to give to the aggregation procedure, which aims at modelling the characteristics of the attack phenomena under scrutiny.

Aggregation functions can be divided into following classes:

- averaging aggregation functions, where the aggregated value of a vector of criteria $\mathbf{z}$ is bounded by
$$\min(\mathbf{z}) \leq f_{aggr}(\mathbf{z}) \leq \max(\mathbf{z})$$
- conjunctive aggregation functions, where the resulting value is bounded by
$$f_{aggr}(\mathbf{z}) \leq \min(\mathbf{z}) = \min(z_1, z_2, \ldots, z_n)$$
- disjunctive aggregation functions, where the resulting value is bounded by
$$f_{aggr}(\mathbf{z}) \geq \max(\mathbf{z}) = \max(z_1, z_2, \ldots, z_n)$$
- mixed aggregation functions, where the $f_{aggr}(\mathbf{z})$ exhibits different types of behavior on different parts of the domain.

In [75], we have mainly considered two families of averaging functions, namely *Ordered Weighted Averaging* functions and the family of *Choquet integrals*, and we have showed in details how to take advantage of them to address attack attribution problems. To make this deliverable as self-contained as possible, we provide here after a brief overview of these aggregation functions.

2.4.4 OWA methods

OWA functions are a family of averaging functions that basically rely on two main characteristics: (i) a weighting vector (like in a classical weighted mean), and (ii) sorting the inputs (usually in descending order), hence the name of *Ordered Weighted Averaging*. This reordering of the components of the input vector $\mathbf{z}$ introduces a non-linearity in the aggregation function. However, it allows the decision-maker to design slightly more complex modeling schemes when dealing with data fusion tasks.

Ordered Weighted Average (OWA)

In [84], a new type of averaging operator called *Ordered Weighted Averaging* (OWA) was introduced by Yager. The OWA operator allows one to include certain relationships among criteria, such as "most of" or "at least" k criteria (out of n) to be satisfied in the overall aggregation process. OWA 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 2.2. (OWA) [84, 10] *For a given weighting vector* $\mathbf{w}$, $w_i \geq 0$, $\sum w_i = 1$, *the* OWA *aggregation function is defined by:*

$$OWA_{\mathbf{w}}(\mathbf{z}) = \sum_{i=1}^{n} w_i z_{(i)} =< \mathbf{w}, \mathbf{z}_{\searrow} > \tag{2.12}$$

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

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

Obviously, the problem of defining the weights w_i to be used in OWA still remains. Yager suggests two possible approaches: (i) either to use some learning mechanism, with sample data and a regression model (e.g., fitting weights by using training data and minimizing the least-square residual error), or (ii) to give some semantics, or meaning to the w_i's by asking a decision-maker or an expert to provide directly those values, based on *domain knowledge*. In many attack attribution cases, we have to rely on the latter, since the process is mostly unsupervised, and thus we have no training samples for the phenomena we aim to identify.

Weighted OWA (WOWA)

Weighted averaging functions, such as OWA or the weighted mean, can be quite convenient aggregation functions when we deal with data fusion tasks, in which criteria of interest are expressed with numerical values (usually, in $[0, 1]^n$). However, the weights

used in weighted mean (WM) and the ones defined for the OWA operator play very different roles. The WM takes into account the *reliability* of each information source (or each expert), whereas the weights used in OWA reflect the importance of the *values*, regardless of their source.

Torra proposed in [77] a generalization of both WM and OWA, called *Weighted OWA* (WOWA). This aggregation function combines the advantages of both types of averaging functions by allowing the user to quantify the reliability of the information sources with a vector $\mathbf{p}$ (as the weighted mean does), and at the same time, to weight the values in relation to their relative position with a second vector $\mathbf{w}$ (as the OWA operator).

The rationale underlying the WOWA function becomes clear in certain intelligent systems in which you have different sensors having a certain reliability (which is known, thus we have the weights $\mathbf{p}$), and where the measurements of those sensors have to be somehow prioritized, irrespective of their reliability. Think, for example, of an automated braking system which assists the driver in a vehicle. In this case, the reliability of the sensors should probably be taken into account (since less reliable sensors may fail or give erroneous measurements); however, the distance measurements to the nearest obstacles may be even more important in certain situations, regardless of which sensors provide the inputs. A WOWA function provides exactly this kind of combination.

In the rest of this section, we start by formally defining the WOWA aggregation function, which can replace the OWA operator in equation **??**. Then, we illustrate its usefulness in the context of the attack attribution method, and how it performs compared to the classical WM or OWA.

Definition 2.3. (Weighted OWA [77]) *Let* $\mathbf{w}, \mathbf{p}$ *be two weighting vectors with* $w_i, p_i \geq 0$, $\sum w_i = 1$, $\sum p_i = 1$. *The* Weighted OWA *aggregation function is defined by:*

$$WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = \sum_{i=1}^{n} u_i z_{(i)}, \tag{2.13}$$

where $z_{(i)}$ *is the* i^{th} *largest component of* $\mathbf{z}$ *and the weights* u_i *are defined as*

$$u_i = G\left(\sum_{j \in H_i} p_j\right) - G\left(\sum_{j \in H_{i-1}} p_j\right)$$

where the set $H_i = \{j | z_j \geq z_i\}$ *is the set of indices of* i *largest elements of* $\mathbf{z}$*, and* G *is a monotone non-decreasing function that interpolates the points* $(i/n, \sum_{j \leq i} w_j)$ *together with the point* $(0, 0)$*. Moreover,* G *is required to have the two following properties:*

1. $G(i/n) = \sum_{j \leq i} w_j$, $i = 0, \ldots, n$;

2. *G is linear if the points $(i/n, \sum_{j \leq i} w_j)$ lie on a straight line.*

In fact, the $WOWA$ operator can be seen as an OWA function with the weights u_i, which are obtained by combining both vectors $\mathbf{w}, \mathbf{p}$ using a generating function G. The mathematical properties of the $WOWA$ operator have been studied and described in [76, 77]. Among others, it has been showed that the weighting vector $\mathbf{u}$ satisfies $u_i \geq 0$, and $\sum u_i = 1$. It is also worth noting that if all $w_i = \frac{1}{n}$, then it turns out that $WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = M_{\mathbf{p}}(\mathbf{z})$, the weighted arithmetic mean. Similarly, when all $p_i = \frac{1}{n}$, then $WOWA_{\mathbf{w},\mathbf{p}}(\mathbf{z}) = OWA_{\mathbf{w}}(\mathbf{z})$.

Obviously, the weights $\mathbf{u}$ also depend on the choice of the interpolation function G, also called W^* in [78]. As suggested by different authors, this function can be chosen as a linear spline that interpolates the points $(i/n, \sum_{j \leq i} w_j)$, or it can be also a monotone quadratic spline as was suggested in [77, 9], as long as the function satisfies the properties stated here above (i.e., the straight line condition). In the experiments performed in this document, we have used the interpolation method described in [78]. Clearly, the advantage of using the WOWA operator (instead of OWA) comes from the additional vector of weights $\mathbf{p}$, which provides more flexibility in the decision-making process by quantifying the reliability of each feature.

2.4.5 Fuzzy integrals

In many multi-criteria problems, and by consequent also in attack attribution problems, it happens quite often that certain criteria are not completely independent. For example, certain combinations of criteria may show some sort of *synergy* (or complementarity), whereas other subsets of criteria could have globally less importance due to the presence of *redundancy* (or substitutability) among them.

Fuzzy integrals are a large family of aggregation functions that can provide effective means for modeling this kind of behavior in MCDA problems. There are mainly two families of fuzzy integrals:

- the *Choquet integral*, which is used for the aggregation of criteria defined on cardinal scales. That is, the scores of the criteria are expressed by real numbers reflecting a degree of satisfaction, a preference, or a degree of membership.

- the *Sugeno integral*, which can be viewed as the ordinal version of Choquet, i.e., the scores of the evaluation criteria are expressed on a finite ordinal (or qualitative)

scale, and thus the Sugeno integral involves a combination of min-max logical operations on a fuzzy measure.

In our experiments, we have only considered the Choquet integral for combining attack features, since most criteria we were dealing with were defined on cardinal scales (usually, on $[0, 1]$). When an analyst is confronted with multi-criteria problems where criteria are defined on both ordinal and cardinal scales, a commonly used approach consists in trying to turn the ordinal problem into a cardinal one, or to get cardinal information from the ordinal one, for example by counting the number of times an alternative is better or worse than the other ones on a given criterion. When all criteria are ordinal ones, then obviously, the Sugeno integral provides more appropriate means to aggregate them.

The purpose of this Section is to demonstrate how the Choquet integral offers a much greater flexibility in feature aggregation by modeling interactions among subsets of features. However, to define the Choquet integral, we need first to introduce a few essential concepts that are associated to it.

Fuzzy measures

Fuzzy integrals are defined with respect to so-called *fuzzy measures*. Given a set of criteria $\mathcal{N}$ (e.g., attack features), a fuzzy measure is simply a set function used to define, in some sense, the importance (or strength) of any subset belonging to the power set of $\mathcal{N}$. It is worth noting that fuzzy measures are not necessarily additive (i.e., the measure of a given set is not necessarily equal to the sum of its subsets). This property of additivity has been somehow relaxed on fuzzy measures by requiring only the *monotonicity* of the measure.

More formally, a fuzzy measure (alternatively called a *capacity* in the literature) is defined as follows.

Definition 2.4. (Fuzzy measure or Capacity) *Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be the index set of n criteria. A* capacity [15] *or* fuzzy measure [70] *is a set function $v : 2^{\mathcal{N}} \rightarrow [0, 1]$ which is monotonic (i.e., $v(\mathcal{A}) \leq v(\mathcal{B})$ whenever $\mathcal{A} \subset \mathcal{B}$) and satisfies $v(\emptyset) = 0$. The measure is normalized if in addition $v(\mathcal{N}) = 1$.*

In multi-criteria decision making, a fuzzy measure is thus a set of 2^n real values where each value can be viewed as the degree of importance of a combination of criteria (also called a *coalition*, in particular in game theory). In other words, from definition 2.4, any subset $\mathcal{A} \subseteq \mathcal{N}$ can be considered as a coalition of criteria, and thus $v(\mathcal{A})$ reflects the

importance of this coalition with a given weight. Note that when new elements are added to a given coalition, it can not decrease its weight (due to the monotonicity condition).

A basic example of a fuzzy measure is

$$v(\mathcal{A}) = \frac{|\mathcal{A}|}{n}, \tag{2.14}$$

for any subset $\mathcal{A} \subseteq \mathcal{N}$, and where $|\mathcal{A}|$ denotes the number of elements in $\mathcal{A}$.

Two mathematical properties of fuzzy measures are particularly of interest. First, a fuzzy measure is *additive* if for all disjoint subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, we have $v(\mathcal{A} \cup \mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B})$. That is, when a fuzzy measure is additive, it suffices to define the n coefficients $v(\{1\}), \ldots, v(\{n\})$ to define the fuzzy measure entirely. Note that in the general case, one needs to define the $2^n - 2$ coefficients corresponding to the 2^n subsets of $\mathcal{N}$, except $v(\emptyset)$ and $v(\mathcal{N})$.

Secondly, a fuzzy measure is *symmetric* if the value $v(\mathcal{A})$ depends only on the cardinality of the set $\mathcal{A}$, i.e., for any subsets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, $|\mathcal{A}| = |\mathcal{B}|$ implies $v(\mathcal{A}) = v(\mathcal{B})$. The example given by equation 2.14 here above is an example of a fuzzy measure which is both additive and symmetric. Note that this type of measure is usually too restrictive in the modeling of a multi-criteria aggregation (in fact, the integral of such fuzzy measures coincides with the arithmetic mean).

A fuzzy measure can also be transformed into an alternative representation, called the *Möbius representation*, which can be helpful in expressing various concepts or quantities related to aggregation functions. For example, we will see in section 2.4.6 that it is convenient to express certain interaction indices in a more compact form. The Möbius representation of a fuzzy measure can be obtained with the Möbius transform.

Definition 2.5. (Möbius transform) [62] *The* Möbius transform *of a fuzzy measure v, denoted by $\mathcal{M}_v$, is a set function defined for every $\mathcal{A} \subseteq \mathcal{N}$ as:*

$$\mathcal{M}_v(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{A}} (-1)^{|\mathcal{A} \setminus \mathcal{B}|} v(\mathcal{B})$$

Example of fuzzy measure.

A convenient way of representing fuzzy measures consists to use a *lattice form* (i.e., a Hasse diagram) of the inclusion relation defined on the set of subsets of $\mathcal{N}$ ([10]). For example, for $n = 3$, we can represent the 2^n elements of a fuzzy measure v as:

$$\begin{array}{ccc} & v(\{1,2,3\}) & \\ v(\{1,2\}) & v(\{1,3\}) & v(\{2,3\}) \\ v(\{1\}) & v(\{2\}) & v(\{3\}) \\ & v(\emptyset) & \end{array}$$

Let v be a fuzzy measure given by

$$\begin{array}{ccc} & 1 & \\ 0.9 & 0.5 & 0.3 \\ 0.5 & 0 & 0.3 \\ & 0 & \end{array}$$

Then, its Möbius transform $\mathcal{M}_v$ is given by

$$\begin{array}{ccc} & 0.1 & \\ 0.4 & -0.3 & 0 \\ 0.5 & 0 & 0.3 \\ & 0 & \end{array}$$

For example,

$$\begin{aligned} \mathcal{M}_v(\{1,2\}) &= (-1)\cdot v(\{1\}) + (-1)\cdot v(\{2\}) + (-1)^2 \cdot v(\{1,2\}) \\ &= -0.5 - 0 + 0.9 = 0.4 \end{aligned}$$

Observe that the sum of all values in the Möbius representation is equal to 1, and the values of v and $\mathcal{M}_v$ coincide on singletons.

We can now introduce the *Choquet integral*, which is defined with respect to a fuzzy measure.

Definition 2.6. (Choquet integral) [15] *The (discrete) Choquet integral of an input vector* $\mathbf{z}$ *with respect to a fuzzy measure (or capacity)* v *is given by*

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} z_{(i)} \left[v(\{j | z_j \geq z_{(i)}\}) - v(\{j | z_j \geq z_{(i+1)}\}) \right] \tag{2.15}$$

where $z_{(1)} \leq z_{(2)} \leq \ldots \leq z_{(n)}$*, i.e.,* $z_{(i)}$ *is the* i^{th} *largest component of the input vector* $\mathbf{z}$*.*

By rearranging the terms of the sum here above, the Choquet integral can alternatively be written as ([10]):

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} \left[z_{(i)} - z_{(i-1)} \right] v(H_i) \tag{2.16}$$

where $H_i = \{(i), \ldots, (n)\}$ is the subset of indices of the $n - i + 1$ largest components of $\mathbf{z}$, and $z_{(0)} = 0$ by convention.

For example, let $n = 3$ and $z_2 \leq z_1 \leq z_3$. Then, using equation 2.15, we have

$$C_v(z_1, z_2, z_3) = z_2 \left[v(\{2, 1, 3\}) - v(\{1, 3\}) \right] + z_1 \left[v(\{1, 3\}) - v(\{3\}) \right] + z_3 v(\{3\})$$

Special cases.

It is worth mentioning that the class of Choquet integrals generalizes averaging functions, such as those discussed previously. In fact, it turns out that weighted means and OWA functions are just special cases of Choquet integrals with respect to additive and symmetric fuzzy measures respectively. More precisely, when a fuzzy measure v is *additive*, the Choquet integral reduces to a weighted arithmetic mean:

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} v(\{i\})\, z_i$$

When a fuzzy measure v is *symmetric*, the Choquet integral reduces to an OWA function as introduced in Section **??**, with weights given by

$$C_v(\mathbf{z}) = \sum_{i=1}^{n} (v_{n-i+1} - v_{n-i})\, z_{\nearrow(i)}$$

with $v_i := v(\mathcal{A})$, such that $|\mathcal{A}| = i$.

Similarly, it can be showed that the WOWA operator is also a special case of Choquet integral [79]. Finally, the Choquet integral with respect to a *symmetric additive* fuzzy measure (as the example 2.14) coincides with the arithmetic mean.

2.4.6 Interactions among criteria

The flexibility of the Choquet integral comes also with a certain complexity, which is mainly due to the fact that a fuzzy measure v must be defined by 2^n values. As a result, the behavior of the decision-making process does not always appear as clearly

when looking at all values of v. Moreover, in multi-criteria problems, it is often the case that certain criteria are not independent, i.e., there is some interaction (positive or negative) among the criteria. For example, two criteria may point essentially to the same concept, and hence should be considered as redundant in the aggregation. Therefore, it is interesting to define some indices to measure the importance of a given criterion, or the interactions among criteria.

To do this, we can use the *Shapley value*, which measures the importance of a criterion i in all possible coalitions. It was first proposed by Shapley [64] in the context of cooperative game theory.

Definition 2.7. (Shapley value [64]) *The Shapley index of a criterion $i \in \mathcal{N}$ w.r.t. a fuzzy measure v is given by*

$$\phi(i) = \sum_{\mathcal{A} \subseteq \mathcal{N} \setminus i} \frac{(n - |\mathcal{A}| - 1)!|\mathcal{A}|!}{n!} [v(\mathcal{A} \cup \{i\}) - v(\{\mathcal{A}\})]$$

The Shapley value *is the vector $\phi(v) = (\phi(1), \ldots, \phi(n))$.*

The Shapley value can be interpreted as the average contribution of each criterion alone in all possible coalitions. With the help of the Möbius transform, the Shapley index can be expressed in a more compact form, which can be also more convenient to calculate:

$$\phi(i) = \sum_{\mathcal{B} |\, i \in \mathcal{B}} \frac{1}{|\mathcal{B}|} \mathcal{M}_v(\mathcal{B})$$

Another important measure is the *interaction index*, introduced by Murofushi and Soneda [53], which quantifies the way two criteria i, j interact in all possible coalitions. As mentioned before, a certain criterion may be irrelevant when considered alone, but its importance regarding the overall decision value may sharply rise when taken in conjunction with other criteria.

Definition 2.8. (Interaction index [53]) *The interaction index between two criteria $i, j \in \mathcal{N}$ w.r.t. a fuzzy measure v is given by*

$$I_{ij} = \sum_{\mathcal{A} \subseteq \mathcal{N} \setminus \{i,j\}} \frac{(n - |\mathcal{A}| - 2)!|\mathcal{A}|!}{(n-1)!} [v(\mathcal{A} \cup i, j) - v(\mathcal{A} \cup i) - v(\mathcal{A} \cup j) + v(\mathcal{A})]$$

When $I_{ij} < 0$, we can say that criteria i, j are linked by a negative synergy (redundancy, or substitutability). Inversely, a positive interaction $I_{ij} > 0$ depicts a positive synergy (or complementarity) between criteria i, j ([26, 27]). When $I_{ij} = 0$, we say that criteria i, j are independent, because the degree of satisfaction due to their combination is equivalent to the sum of the individual contributions of both criteria (i.e., it reflects a null synergy).

This definition of I_{ij} due to Murofushi and Soneda holds for a pair of criteria only, but it was extended by Grabisch et al. [31] for any coalition $\mathcal{A}$ (and not just pairs) of criteria:

$$I(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{N} \setminus \mathcal{A}} \frac{(n - |\mathcal{B}| - |\mathcal{A}|)!|\mathcal{B}|!}{(n - |\mathcal{A}| + 1)!} \sum_{\mathcal{C} \subseteq \mathcal{A}} (-1)^{|\mathcal{A} \setminus \mathcal{C}|} v(\mathcal{B} \cup \mathcal{C}) \tag{2.17}$$

which again can be expressed in a more compact form using the Möbius transform:

$$I(\mathcal{A}) = \sum_{\mathcal{B} |\, \mathcal{A} \subseteq \mathcal{B}} \frac{1}{|\mathcal{B}| - |\mathcal{A}| + 1} \mathcal{M}_v(\mathcal{B}) \tag{2.18}$$

Clearly, $I(\mathcal{A})$ coincides with I_{ij} for $\mathcal{A} = \{i, j\}$, and coincides with $\phi(i)$ for $\mathcal{A} = \{i\}$. As we show in the next section, those interaction indices present a special interest when the analyst must deal with the problem of defining fuzzy measures.

2.4.7 Construction of fuzzy measures

The flexibility of the Choquet integral has also a major drawback, which is related to its exponential complexity (remember that $2^n - 2$ real values must be defined in a fuzzy measure). Just like for OWA weighting vectors, a decision-maker or an analyst may want to define a fuzzy measure "by hand", based on his domain knowledge. Quite obviously, this approach becomes rapidly impractical when the number of criteria increases.

Another approach consists in fitting a fuzzy measure to empirical data, assuming that we are able to collect a set of training samples. This can be again formulated as an optimization problem, using either a least squares criterion or a least absolute deviation to minimize the error [75]. However, the problem of collecting meaningful training data remains. For attack attribution purposes, it is not so evident to gather sound experimental measurements that can be used as training data set, since we usually do not know the "ground truth" about the underlying phenomena.

Another issue with Choquet is the difficulty to interprete those values (certainly when $n \nearrow$), and thus to analyze the behaviour of the aggregation model. To deal with these issues, several particular sub-models have been proposed.

λ-fuzzy measures

Sugeno [70] has proposed a simplified sub-model based on λ-fuzzy measures as a way of reducing the complexity of a fuzzy measure[4]. The idea is to define the values of the fuzzy measure v only for individual criteria, and then to solve a linear system to determine all other values for the coalitions, based on some constraints imposing a sub- or super-additivity on the fuzzy measure.

Definition 2.9. (λ-fuzzy measure [70]) *Given a parameter $\lambda \in]-1, \infty[$, a λ-fuzzy measure is a fuzzy measure v that, for all disjoint sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, satisfies*

$$v(\mathcal{A} \cup \mathcal{B}) = v(\mathcal{A}) + v(\mathcal{B}) + \lambda v(\mathcal{A}) v(\mathcal{B})$$

Under these conditions, all values $v(\mathcal{A})$ can be immediately calculated from the n independent values $v(\{i\})$, $i = 1, \ldots, n$, by using the following formula

$$v(\bigcup_{i=1}^{m} \{i\}) = \frac{1}{\lambda} \left(\prod_{i=1}^{m} (1 + \lambda v(\{i\})) - 1 \right), \ \lambda \neq 0 \tag{2.19}$$

where the coefficient λ is determined from the boundary condition $v(\mathcal{N}) = 1$, and involves solving following equation on $(-1, 0)$ or $(0, \infty)$

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda v(\{i\})) \tag{2.20}$$

A λ-fuzzy measure is either sub- or super-additive, when $-1 < \lambda \leq 0$ or $\lambda \geq 0$ respectively. Note that a λ-fuzzy measure is an example of a distorted probability measure [54].

k-additive fuzzy measures

To decrease the exponential complexity of fuzzy measures, Grabisch proposed another sub-model called k-order additive fuzzy measures, or shorter *k-additive* fuzzy measures [28]. The idea is to construct a fuzzy measure where the interaction among criteria is limited to groups of size k (or less). For example, in a 2-additive fuzzy measure, we can only model pairwise interactions among criteria, but no interactions in groups of 3

[4] For this reason, λ-fuzzy measures are also called Sugeno measures.

or more. In fact, all values of the fuzzy measure for groups of size larger than k are determined by various linear constraints.

k-additive fuzzy measures provide a good trade-off between complexity and flexibility of the model. Instead of $2^n - 2$ values, they require only $\sum_{i=1}^{k} \binom{n}{i}$ values to be defined. 1-additive fuzzy measures are just ordinary additive measures (for which only n values are needed), but they are usually too restrictive for an accurate representation of complex problems[5]. In practice, 2-additivity seems to be the best compromise between low complexity and richness of the model [28]. In this case, only $n(n+1)/2$ values need to be defined.

Definition 2.10. (k-additive fuzzy measure [28]) *A fuzzy measure v is said to be k-additive ($1 \leq k \leq n$) if its Möbius transform verifies*

$$\mathcal{M}_v(\mathcal{A}) = 0$$

for any subset $\mathcal{A}$ with more than k elements, $|\mathcal{A}| > k$, and there exists a subset $\mathcal{B}$ with k elements such that $\mathcal{M}_v(\mathcal{B}) \neq 0$.

A fundamental property of k-additive fuzzy measures is

$$\begin{cases} I(\mathcal{A}) = 0, \text{ for every} \mathcal{A} \subseteq \mathcal{N} \text{ such that } |\mathcal{A}| > k \\ I(\mathcal{A}) = \mathcal{M}_v(\mathcal{A}), \text{ for every} \mathcal{A} \subseteq \mathcal{N} \text{ such that } |\mathcal{A}| = k \end{cases}$$

Defining a 2-additive measure

From a practical viewpoint, a decision-maker will usually define a 2-additive fuzzy measure, since this is a good trade-off between complexity of the model (in terms of the number of values to determine) and its effectiveness and flexibility to include interactions (synergies or redundancies) among pairs of features. To do this, two different approaches can be used. In both cases, we start by defining the interaction indices $I_2(\mathcal{A})$ of v_2, for all combinations of 2 criteria or less. In other words, for all $\mathcal{A} \subseteq \mathcal{N}$ such that $|\mathcal{A}| > 2$, we can set the values of $I_2(\mathcal{A}) = 0$ (by definition of a 2-additive measure).

Then, based on our domain knowledge about the attack features we have considered, we have to define the interaction indices $I_2(\mathcal{A})$ of the fuzzy measure v_2. For example, with $n = 4$, we have to define the four indices of the singletons (on the lower row of

[5] 1-additivity does not permit interaction, and the Choquet integral w.r.t. 1-additive fuzzy measures is simply a weighted arithmetic mean.

the lattice here under), and all pairwise interactions for the 6 couples of features (on the third row of the lattice). One possible example is given in the diagram here under, in which we have set a strong redundancy between features 1 and 2 (note the negative index for $I_2(\{1,2\})$), and a weak synergy between pairs involving features 2, 3 and 4:

$$\begin{array}{ccccccc} & & & 0 & & & \\ & & 0 & 0 & 0 & 0 & \\ \textbf{-0.20} & & 0 & 0 & 0.05 & 0.05 & 0 \\ & & 0.10 & 0.20 & 0.40 & 0.30 & \\ & & & 0.50 & & & \end{array}$$

Note that the values $I_2(\mathcal{A})$ for singletons correspond in fact to the *Shapley* values introduced previously in Definition 2.7, which represent the importance factors of each criterion alone.

Next, a first approach consists to convert $I_2(\mathcal{A})$ to its corresponding fuzzy measure v_2 (which will be by definition 2-additive). To do this transformation $I_2(\mathcal{A}) \rightarrow v_2(\mathcal{S})$, we just need to use the conversion formula given by Grabisch in [29]:

$$v(\mathcal{S}) = \sum_{\mathcal{A} \subset \mathcal{N}} \beta^{|\mathcal{A}|}_{|\mathcal{S} \cap \mathcal{A}|} I(\mathcal{A})$$

where β is a quantity related to the Bernouilli numbers B_k, and is given by

$$\beta^l_k = \sum_{j=0}^{k} \binom{k}{j} B_{l-j}, \; k, l = 0, 1, 2, \ldots$$

The resulting 2-additive fuzzy measure v_2 obtained using this method is defined by the following values, and formula 2.15 or 2.16 can then be used to calculate the 2-additive Choquet integral C_{v_2}.

$$\begin{array}{ccccccc} & & & 1 & & & \\ & & 0.65 & 0.55 & 0.85 & 0.95 & \\ 0.20 & & 0.55 & 0.45 & 0.65 & 0.55 & 0.60 \\ & & 0.15 & 0.25 & 0.35 & 0.25 & \\ & & & 0 & & & \end{array}$$

A second approach consists to calculate the Choquet integral directly from the values provided by $I_2(\mathcal{A})$ by using the expression of C_v given by Grabisch in [30]:

$$C_v(\mathbf{z}) = \sum_{i,j \in N | I_{ij} > 0} (z_i \wedge z_j) I_{ij} + \sum_{i,j \in N | I_{ij} < 0} (z_i \vee z_j) |I_{ij}| + \sum_{i \in N} z_i \left[\phi_i - \frac{1}{2} \sum_{j \neq i} |I_{ij}| \right] \quad (2.21)$$

where ϕ_i is the Shapley value of v defined in 2.7, I_{ij} is the interaction index between criteria i and j (defined in 2.8), and $\mathbf{z}$ is the vector of criteria scores obtained from the concatenation of all edge-weigted graphs, as defined previously in Section 2.3.

As pointed out by Grabisch [30], the expression 2.21 is remarkable for two reasons:

- It explains well the meaning of the interaction index and Shapley value: a positive interaction induces a conjunctive aggregation of scores (*necessarily both* scores have to be high to produce a high overall score), while a negative interaction induces a disjunctive aggregation (it is *sufficient* that one score is high). Clearly, the Shapley value is the linear part of the model, while interaction is the non-linear part.

- Coefficients are non-negative, and moreover, if the capacity is normalized, they sum up to 1. In other words, this means that the Choquet integral is a convex combination of the scores z_i on all criteria, and of all disjunctive and conjunctive combinations of scores on pairs of criteria. Hence, the coefficient of a given term can be interpreted as the percentage of contribution of such term to the overall score. This feature is highly appreciated in practice, because it allows an analyst to easily understand each decision behavior between two security events. Indeed, each non-null term of this formula can be seen as the elementary contribution of the corresponding singleton or pairs of criteria to the overall aggregation score.

When defining interaction indices for pairs of criteria, the only conditions one must verify are the additivity and monotonicity of the resulting measure. That is, it is sufficient to verify that the following quantity is always positive $\forall i \in N$:

$$\left[\phi_i - \frac{1}{2} \sum_{j \neq i} |I_{ij}| \right] \geq 0 \quad (2.22)$$

Yet other methods have been developed to build decision-making models with reduced complexity, such as p-symmetric fuzzy measures [51] or k-tolerant and k-intolerant fuzzy measures [47]. However, we limit our discussion to the two aforementioned methods

as these have been implemented in the TRIAGE framework, and we will illustrate their application in the context of attack attribution in the next Chapters.

Note also that various practical examples are given in [75] to illustrate the use of OWA operators and fuzzy integrals to model aggregation schemes in attack attribution. From these examples, it turns out that 2-additive fuzzy measures best performed on difficult decision-making cases, and could better emphasize the separation between *desired* and *unwanted* cases of security events linkage.

2.5 Conclusion

In this Chapter, we have presented TRIAGE, a generic software analysis framework that has been developed in WOMBAT to address in a systematic way the *attack attribution* problem.

The framework relies on a novel combination of graph-based representation and clustering, with a data fusion process inspired by Multi-Criteria Decision Analysis (MCDA). In particular, we have demonstrated how an intelligent combination of multiple attack features can effectively help a security analyst in the process of identifying attack phenomena, and perhaps more importantly, how it helps to model their dynamic behaviors thanks to different aggregation methods, such as OWA functions, or more advanced methods like the *Choquet integral*.

In the next Chapters, we describe extensively the experimental results obtained with TRIAGE and we show how this framework has been successfully applied to various WOMBAT datasets to perform intelligence analyses by taking advantage of many structural and contextual features of security data sets developed by the partners.

3 Analysis of Rogue AV Campaigns

3.1 Introduction

One of the requirements in the design of the our attack attribution framework was its applicability to a broader set of problems relating to Internet threats, intelligence analysis, or more generally to the analysis of any security data set. In this Chapter, we demonstrate how TRIAGE has been used to address an emerging security problem, namely *rogue security software*. This type of misleading application pretends to be legitimate security software, such as an antivirus scanner, but in reality, these programs provide little or no protection and, in fact, may actually install the very malicious code it purports to protect against.

In the following Sections, we describe how we leveraged our multi-criteria aggregation method to analyze the *campaigns* through which this type of malware is distributed, i.e., what are the underlying techniques, server infrastructure and coordinated efforts employed by cyber-criminals to spread their rogue software. In the experimental results, we give a more in-depth presentation of some typical networks of rogue domains that are likely linked to the same campaign, which helped reveal the *modus operandi* of the criminal organizations behind them. Finally, we have compared our findings with a different type of client-threats, i.e., the ones related to browser exploits. Thanks to TRIAGE experimental results, we could underline some profound differences in the structures and dynamics of these two threat ecosystems.

Some experimental results presented here after have been published in the Symantec Internet Security Threat Report in 2009 (with a special edition *Symantec Report on Rogue Security Software* [73]), and have been presented at international academic conferences [20, 19].

3.1.1 Rogue AV Ecosystem

A rogue security software program is a type of misleading application that pretends to be a legitimate security software, such as an anti-virus scanner, but which actually provides the user with little or no protection. In some cases, rogue security software (in the following, more compactly written *rogue AV*) actually facilitates the installation of the very malicious code that it purports to protect against ([73]).

Rogue AV makes its way on victim machines in two prevalent ways. First, social engineering techniques, such as Web banner advertisements, pop-up windows and attractive messages on blogs or sent via spams, can be used to convince unexperienced users that a rogue tool is free and legitimate and that its use is necessary to remediate often inexistent threats found on the victim's computer (hence, the other name *scareware* given to those programs [73])). A second, more stealthy technique consists in attracting victims to malicious web sites that exploit vulnerabilities in the client software (typically, the browser or one of its plugins) to download and install the rogue programs, sometimes without any user intervention (i.e., via *drive-by* downloads).

Rogue AV programs are distributed by cyber-criminals to generate a financial profit. In fact, after the initial infection, victims are typically tricked into paying for additional tools or services (e.g., to upgrade to the full version of the program or to subscribe to an update mechanism), which are of course fictitious and completely ineffective. For the victims, the initial monetary loss of these scams ranges from $30 to $100. Some examples of prevalent rogue security applications (as reported by Symantec for the period July 2008 - June 2009 [73]) are known as SpywareGuard 2008, AntiVirus 2008, AntiVirus 2009, Spyware Secure, and XP AntiVirus.

Despite its reliance on relatively unsophisticated techniques, rogue AV has emerged as a major security threat, in terms of the size of the affected population (Symantec's sensors alone reported 43 million installation attempts over a one-year monitoring period [73]), the number of different variants unleashed in-the-wild (over 250 distinct families of rogue tools have been detected by Symantec [73]), and the volume of profits generated by cyber-crooks. Their business model relies on an affiliate-based structure, with per-installation prices for affiliate distributors ([37, 73] reported earnings of as much as $332,000 a month in affiliate commissions alone, as observed on a distribution website called `TrafficConverter.biz`).

The prevalence and effectiveness of this threat has spurred considerable research by the security community [73, 55, 56]. These studies have led to a better understanding of the technical characteristics of this phenomenon (e.g., its advertising and installation techniques) and of the quantitative aspects of the overall threat (e.g., the number and geolocation of the web sites involved in the distribution of rogue programs and of their victims).

However, a number of areas have not been fully explored. Indeed, malware code, the infrastructure used to distribute it, and the victims that encounter it do not exist in isolation, but are different aspects of the coordinated effort made by cyber-criminals to spread or distribute rogue AV. We refer to such a coordinated activity as a rogue AV *campaign*. Indeed, one assumption that can reasonably be made is that a campaign is managed by a group of people, who are likely to reuse, at various stages of the campaign,

the same techniques, strategies, and tools (for obvious reasons of development cost).

Consequently, we have applied TRIAGE to a specific data set made of 5,852 rogue web sites, as observed by HARMUR during a two-month reporting period (July to August, 2009), with the purpose of identifying any emerging patterns in the way rogue domains are created, grouped, and interconnected with each other, based upon common elements (e.g., rogue AV-hosting sites, DNS servers, domain registration) that are likely due to the same root cause, i.e., the same rogue AV campaign.

3.1.2 HARMUR Dataset

The analysis of the rogue AV threat has been performed by leveraging HARMUR, a **H**istorical **AR**chive of **M**alicious **URL**s, which was introduced in D13 (D3.3) as an aggregator of information on web threats. HARMUR [41] builds upon two types of information feeds: URL feeds that provide lists of fresh URLs likely to be of interest, and analysis feeds that build a wide range of contextual information around each URL introduced in the system by the URL feeds. By revisiting the analysis of each URL on a periodic basis, HARMUR is capable of providing insights on the dynamics of the structure of these threats. In the context of this work, however, we do not take the dynamics into account and we consider a "flat" representation of the URL metadata as a source of contextual information on a given threat, similarly to what has been done in D18 (D4.6) when describing the HARMUR "feature space".

To build an initial seed of domains associated to the rogue AV distribution, we aggregated information from a number of different sources:

- Norton Safeweb (`http://safeweb.norton.com`)
- Malware Domain List (`http://malwaredomainlist.com`)
- Malware URL (`http://www.malwareurl.com`)
- Hosts File (`http://www.hosts-file.net`)

All these sources offer a rough categorization of the type of each malicious domain they are listing, and allowed us to systematically collect all the domains that were believed to be correlated to the rogue AV distribution by means of simple heuristics.

To complete our picture on the collected domains, we have integrated our domain list with the information generated by freely accessible IP-NAME mapping datasets (`http://www.robtex.com`). This allowed us to discover all the domain names hosted on each IP where at least one rogue domain had been found.

Once the initial list of domains was created, we have have used HARMUR to collect as much information as possible on each of them, on their relation to the associated web servers, and on the registration dynamics. In the specific context of this experiment with TRIAGE, we have selected some of the contextual features descrbed in D18 to generate the necessary contextual information on the identified rogue AV domains, and on all the other domains that were discovered to be sharing the same server as rogue AV domains thanks to DNS mapping information:

- Information on the security state of a domain.
    - **Norton Safeweb information.** For each domain, we have queried its security status taking advantage of the Norton Safeweb website reputation service[1]. This allowed us to retrieve information on a variety of threats known to be present on each domain, ranging from browser exploits, to malware samples, to phishing sites.
    - **Google Safe Browsing information.** We have taken advantage of the Google Safe Browsing API[2] to detect the presence of threats within a given domain.

- Information on the domain.
    - **Registration information.** We have parsed the registration data obtained via the WHOIS protocol in order to get information on the identity of the registrant and of the provided contact email address, as well as the name of the registrar[3].
    - **DNS relations.** By means of DNS queries, we have retrieved for each domain the associated *NS* records and all the *A* records associated to all the hostnames known to belong to it. Whenever only one domain name was available and we had no information on the associated hostnames, we considered as hostnames the domain name itself and the hostname generated by prepending the standard "www" name.

- Information on the servers.

[1] `http://safeweb.norton.com`

[2] `http://code.google.com/apis/safebrowsing/`

[3] The WHOIS specification requires WHOIS records to be human readable, and does not specify their syntax and their semantics. As a consequence, the data stored by different registrars is often in different formats. We have built a generic parser that handles a vast number of registrars and 17 specific parser for other common registrars, but despite of this effort registration information is not available for all the domains taken into consideration.

- **Geolocation and AS information.** For each web server associated to the rogue domain through a DNS *A* record, we have collected information on its geographical location as well as its associated Autonomous System number.
- **Server uptime and version string.** By means of HTTP HEAD packets, we have tested the responsiveness of the discovered servers and, by looking at the HTTP response headers, we have collected information on the server configuration by looking at the advertised server version string.

By periodically iterating the collection of this information, we have been able to generate a complete dataset on the structure and the dynamics of the rogue AV threat landscape.

Experimental dataset

The HARMUR data set we have considered for this analysis was collected over a period of approximately two months, in July and August 2009. The rogue AV-hosting servers were identified through a variety of means, including automated and manual feeds.

To build our experimental data set, we considered 5,852 DNS entries[4] pointing to 3,581 distinct IP addresses of web servers that were possibly hosting rogue security software. After analysis, the 3,581 Web servers have been broken down into the following categories:

- 2,227 Web servers (as identified by their unique IP addresses) were hosting domains serving only rogue security software,
- 118 servers hosted rogue security software along with domains that served malicious code,
- the remaining IP addresses served malicious code along with innocuous domains.

It is worth noting that at least 45% of these domains were registered through just 29 out of several hundred existing domain registrars. This may indicate that rogue security software distributors are choosing specific registrars, possibly because of perceived lax security or oversight of the registration of names.

Looking at the email addresses provided by all Registrants of rogue AV domains, we observed that, besides a list of popular email hosting services (like Gmail, Yahoo! Mail, Lycos, etc.), about 26% of the analyzed domains make use of anonymous domain registration services such as `domainsbyproxy.com`, `whoisprivacyprotect.com`,

[4] In the paper published at RAID [20], we extended this list of rogue AV domains to 6,500 DNS entries.

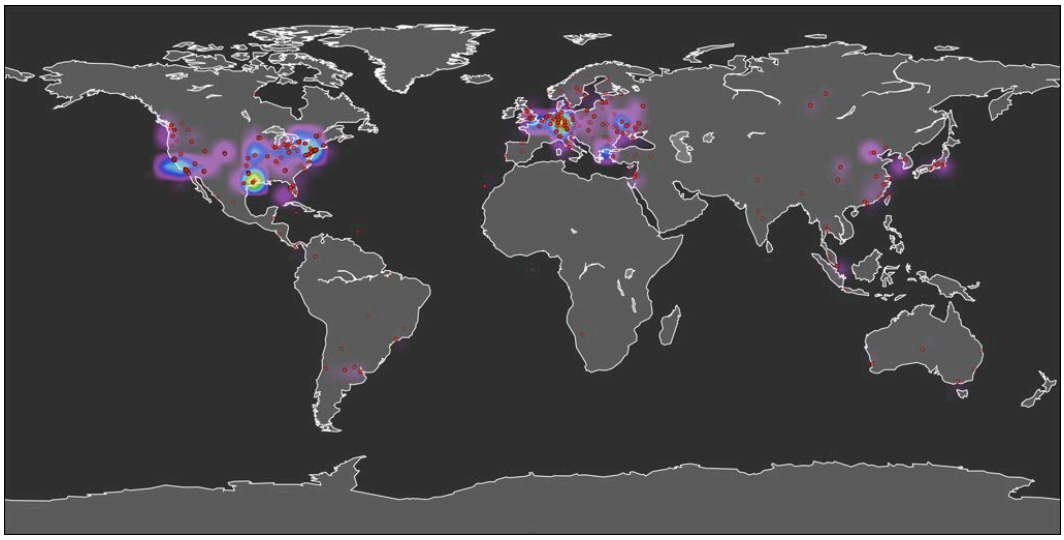


Figure 3.1: Map of rogue AV servers, as monitored by HARMUR over a period of two months, in July and August 2009.

`id-private.com`, and `space.kz`. In some other cases, we also observed that certain ISPs, even though they do not formally offer anonymous domain registration services, are rather lax in their verification of registrant identities and email addresses. For instance, `Namecheap.com` is often associated to weird registrant names such as "Kyle", or "AA".

Regarding the geographical distribution of these servers (Table 3.1), we observed that 53% were in the USA, far more than any other country. The high ranking of the US may be due to the methods used for identifying rogue AV sites, which more easily identified English-language sites than sites marketing scams in other languages. Germany ranked second in this survey, accounting for 11% of the total servers hosting rogue security software identified. Fig. 3.1 graphically depicts the geographical distribution of rogue AV servers, where each red dot represents a distinct server, while the different gradients on the background underline the areas with highest density of deployed servers.

3.2 Selection of Domain Features

We turn now to the application of our attack attribution method to this data set of 5,852 rogue AV websites. As described previously, we want to identify emerging patterns in

Table 3.1: Geographical distribution of rogue AV servers, as monitored by HARMUR over a period of two months, in July and August 2009.

Rank	Country	Percentage
1	United States	53%
2	Germany	11%
3	Ukraine	5%
4	Canada	5%
5	United Kingdom	3%
6	China	3%
7	Turkey	3%
8	Netherlands	2%
9	Italy	2%
10	Russia	1%

the way domains (and servers) are grouped and interconnected with each other, based upon a series of common elements.

In this Section, we describe which *domain features* we found relevant for analyzing rogue AV campaigns. Some illustrative examples of these features are summarized in Table 3.2.

3.2.1 Server IP addresses

Every web site (or web domain) has to be hosted on a publicly available Web server, to which an IP address has been assigned. The mapping between a web site and the server IP address is maintained via the Domain Name System (DNS), a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network. Via specific requests to DNS servers, one can know the IP address of the server hosting a given domain name, as well as the *authoritative nameserver* (i.e., the DNS server responsible for keeping the records of the domain name).

As a result, if cyber-crooks want to distribute their rogueware to innocent victims, they have to *i)* find some hosting web server with a publicly available IP address; and *ii)* register their domain names and let them point to those server IP addresses. Due to the fraudulent aspect of their business, we could a priori believe that cyber-criminals would host rogue AV websites on compromised computers (e.g., on zombie machines that are part of a botnet), as it is often the case with phishing pages, illegal porn and

warez websites. However, our experimental analysis tends to show that a large majority of those rogue domains are hosted by some popular commercial domain registrars and web hosting companies (e.g., GoDaddy, eNom, Tucows, OnlineNIC, etc).

Assuming that cyber-criminals want to make their efforts profitable, they will probably look for a certain type of hosting infrastructure that is not too expensive, somehow "reliable" for them (i.e., offering some oversight regarding the registration of suspicious names, and slow in taking down fraudulent domains), and possibly allowing to automate certain administrative tasks, such as the bulk registration of new web domains. Furthermore, due to the affiliate structure and the social networking aspect of those criminal organizations, we hypothesize that many affiliates belonging to the same community will most likely reuse the same tools in order to quickly create new domains, distribute rogue AV and thus make some profit.

Consequently, our intuition about a rogue AV campaign is that cyber-crooks of a same organization will, at various stages of the campaign, reuse the very same techniques to create and register their domains. They may also choose for a specific domain registrar or web hosting company, for the very same reasons explained here above.

However, to reduce the cost of ownership, most web hosting companies offer some shared hosting on *server farms*, which means that two domains registered during the same campaign can perfectly be hosted on two different servers of the same Web company (e.g., GoDaddy), having thus nearby IP addresses (i.e., located within the same IP subnet). For this reason, we observed that it is sometimes useful to group server IP addresses by Class C or Class B-subnet, such that we can compare the IP subnets on which two rogue sites are located, instead of the exact IP addresses (see for example domains 465709 and 465706 in Table 3.2 on page 50).

It is also worth noting that some rogue AV domains were observed as being hosted on more than one server, which may be an attempt to reduce the effectiveness of mitigation measures such as IP blocking or blacklisting servers, by providing a certain level of redundancy with spare IP addresses being reserved for a given domain. That is, when cyber-crooks detect an attempt of blocking certain IP addresses, they just need to activate a spare IP address and let their rogue domain point to this new address (i.e., by changing the 'A' record of the rogue domain in the nameserver).

In conclusion, the considerations here above lead us to define the following *domain features*, which can be fed to TRIAGE and used to link rogue domains to the same campaign:

- F_{IP}, which represents the IP address(es) of the web server(s) hosting a given rogue domain. The corresponding feature vector is thus a set of server IP addresses;
- $F_{Cl.C}$, which represents the Class C-subnet(s) of the web server(s) hosting a given

rogue domain. The corresponding feature vector is thus a set of Class C-subnets;

- $F_{Cl.B}$, which represents the Class B-subnet(s) of the web server(s) hosting a given rogue domain. The corresponding feature vector is thus a set of Class B-subnets;
- F_{NS}, which represents the IP address(es) of the authoritative nameserver(s) for a given rogue domain. The corresponding feature vector is thus a set of nameserver IP addresses.

Finally, it is important to note that none of these network observables, if considered *alone*, is sufficient to attribute with high confidence two rogue domains to the same campaign. Indeed, using a specific web hosting provider, or pointing to the same web server, does not necessarily mean that the hosted domains are part of the same rogue campaign. In fact, many legitimate web sites are being hosted on the very same servers as those of rogue sites. Additional features are thus needed to bring stronger evidence of two domains likely involved in the same Rogue AV campaign.

3.2.2 Whois information

Whois [22] is a query/response protocol that is widely used for querying databases in order to determine the registrant or assignee of Internet resources, such as a domain name, an IP address block, or an autonomous system number. By performing periodically Whois lookups, HARMUR can retrieve and store some valuable information about each web site being monitored, such as the *registrant* name (usually, an email address), the domain *registrar*, the geolocation of the hosting web server, and the creation date of the domain.

For the very same reasons as those stated here above, we hypothesize that two domains created by the same criminal organization, for the purpose of running a rogue campaign, will have commonalities in one or several *Whois* features (i.e., same registrant address, or same registrar and creation date for domains that are create in bulk using automated tools).

This leads us to define the following site features related to the available Whois information:

- F_{Reg}, which refers to the name or email address of the registrant;
- F_{Rar}, which refers to the name of the Registrar;
- F_{Geo}, which refers to the geolocation of the web server hosting a given domain (i.e., a country);

Site Id	F_{Dom}	F_{IP}	F_{CLC}	F_{CLB}	F_{NS}	F_{Reg}	F_{Reg}	F_{Geo}	F_{Crea}
271665	windowsantivirus2008.com	74.54.82.219, 209.62.20.233	74.54.82	74.54	74.54.82.119	domadmin @privateregistrations.ws	DIRECTI	US	2008-06-04
271621	Xp-2008-Antivirus.com	208.73.210.27, 208.73.210.121	208.73.210	208.73	204.13.161.55, 204.13.160.55	-	-	US	-
272656	malwaredefender2009.com	67.43.237.75, 211.95.73.189	67.43.237	67.43	208.76.62.100,	jsfsl2341@googlemail.com	Registrme Ltd.	CN	2009-03-04
211552	anti-malware-2010.com	74.205.8.7	74.205.8	74.205	75.102.60.66 216.69.185.2, 208.109.255.2	ANTI-MALWARE-2010.COM @domainsbyproxy.com	GODADDY.COM	US	2009-05-31
122287	antivirus360remover.com	174.132.250.194	174.132.250	174.132	207.218.223.162, 207.218.247.135	ANTIVIRUS360REMOVER.COM @domainsbyproxy.com	GODADDY.COM	US	2009-02-22
272539	norton-antivirus-2010.com	74.208.42.60, 74.208.156.41, 82.165.245.27	74.208.42, 74.208.156, 82.165.245	74.208, 82.165	74.208.3.8, 74.208.2.9	proxy1994891 @1and1-private-registration.com	GODADDY.COM	US	2007-07-08
272540	nortonantivirus2010.com	69.64.145.229, 209.249.222.18, 208.116.34.163	69.64.145 209.249.222 208.116.34	69.64 209.249 208.116	209.249.221.130, 72.34.41.47, 74.81.64.51	support@NameCheap.com	ENOM	US	2006-08-13
334096	home-antivirus2010.com	72.52.210.132	72.52.210	72.52	76.73.35.154, 72.52.210.132	blair@8081.ru	ONLINENIC	US	2009-07-14
334091	homeanti-virus-2010.com	72.52.210.130	72.52.210	72.52	76.73.35.155, 72.52.210.130	blair@8081.ru	ONLINENIC	US	2009-07-14
389838	homeav-2010.com	72.52.210.133	72.52.210	72.52	76.73.35.158, 72.52.210.133	tours@infotorrent.ru	ONLINENIC	US	2009-07-14
465709	pc-anti-spyware-2010	174.139.5.50, 209.31.180.235	174.139.5, 209.31.180	174.139, 209.31	174.139.5.50, 209.31.180.235	argue@8081.ru	ONLINENIC	US	2009-07-29
465706	pc-anti-spy2010.com	174.139.243.45 209.31.180.233	174.139.243 209.31.180	174.139 209.31	174.139.243.45, 209.31.180.233	pixie@ml3.ru	ONLINENIC	US	2009-07-29
465710	p-c-anti-spyware-2010.com	174.139.243.46, 209.31.180.234	174.139.243 209.31.180	174.139 209.31	174.139.243.46, 209.31.180.234	kites@e2mail.ru	ONLINENIC	US	2009-07-29

Table 3.2: Network observables used as *domain features* for a set of rogue AV domains and associated web servers.

- F_{Crea}, which refers to the creation date of the domain, as given in the registration data.

As with the previous ones, these features can be used to link rogue domains to the same campaign, but none of them is sufficient by itself to identify a rogue campaign in a reliable fashion.

3.2.3 Domain names

When cyber-crooks want to create dedicated web sites for hosting rogue AV software, we observed that, in many cases, they tend to choose some appealing names that can easily lure users into believing that their web sites are genuine and legitimate. Furthermore, the domain names may also be chosen to be consistent with the "brand name" given to the distributed rogue software, for example `windowsantivirus2008.com`, `xp-2008-antivirus.com`, `malwaredefender2009.com`, or `pcregistrycleaner.com`.

Consequently, a good reason to look at domain names patterns is that rogueware distributors and their affiliates can possibly rely on the same software tools to quickly create new domain names in an automated fashion, e.g., by combining (randomly or with a given logic) different text *tokens* that are commonly used in the names of legitimate anti-malware products (e.g., XP, anti, virus, spyware, malware, registry, home, cleaner, defender, etc).

Identifying patterns and commonalities among domain names may thus give a good indication on the tools or techniques that are being reused by criminals during the same campaign when creating new rogue domains. We denote this feature as F_{Dom}.

3.2.4 Other possible features

Some other network observables could be useful for the identification and analysis of rogue AV campaigns. While we haven't used those features in the multi-criteria fusion, we believe they can bring other interesting viewpoints on groups of domains attributed to the same campaign.

Software version

A first additional feature we may want to consider is the *software version* of the HTTP server hosting the rogue domain. This feature can be obtained either from the HTTP header sent by the server, or sometimes also from Whois servers. Some possible values of server version are, for example, "`Apache/2.2.3 (Red Hat)`", "`Microsoft-IIS/6.0`",

but also "`nginx/0.6.35`", "`lighttpd/1.5.0`", "`LiteSpeed`", or eventually some other not-so-commonly-used software like "`Oversee Turing`" and "`gws`".

As such, two domains hosted on different web servers running the same HTTP software may not indicate anything useful if we consider this feature alone. However, looking at the distribution of software versions for a group of domains that we suspect of being part of the same phenomenon may help us to confirm the results. More precisely, we see two main reasons for this:

- if cyber-criminals are hosting their rogue web site on compromised servers (as they don't want to pay a commercial provider for hosting hundreds of domains), chances are high that they will reuse the very same technique (or exploit) to take advantage of a given vulnerability they discovered in a *specific version* of a server software (e.g., an exploit only targeting Apache/2.2.3 running on Windows platforms).

- cyber-criminals can also decide to hire zombie machines that are part of a botnet to host their server infrastructure. In this case, the available server software will depend on the bot software used to control the compromised machines, and in most cases, the web server is then based on some custom, lightweight server software that can be easily downloaded and installed, in a stealthy manner, on zombie machines with high-speed connections and public IP addresses. Since the bot herder has usually full access to the underlying operating system of the bots, he can even configure the HTTP server software in such a way that standard banners are replaced by a stealthier one (of his own choice).

Threat type

Another useful feature that we could include in the analysis of rogue AV campaigns is the *type of threats* found on each rogue web site, such as possible browser exploits, trojan downloads, obfuscated (malicious) javascripts, or fake codecs installation. This kind of information is available via web site reputation services, such as *Norton SafeWeb*, which is used as analysis feed by HARMUR. Those reputation services usually rely on high-interaction honeypots to effectively detect these client threats.

In our measurements, only a small fraction of the monitored rogue domains also involved the hosting of additional threats, such as browser exploits, malware downloads, etc. However, we hypothesize that using this feature in combination with others may be helpful to identify and characterize malicious domains that are controlled by the same group of criminals, as they will quite likely reuse the same techniques to distribute malware or to compromise new victims.

3.3 Graph-based Clustering

Based on the feature analysis here above, we have selected the following 6 features for the application of the graph-based clustering component: F_{Dom}, F_{IP}, $F_{Cl.C}$, $F_{Cl.B}$, F_{Reg} and F_{NS}.

The other features were not selected mainly for two reasons: *(i)* for certain aspects, the data collected by HARMUR was, at the time of this analysis, either too generic or incomplete to be used as clustering feature (like threat types or Whois information, also because some web sites were not active any more); and *(ii)* features like F_{Geo} and F_{Rar} were too redundant with other selected features, such as the information provided by IP subnets, and the software module for running an aggregation by means of Choquet integral (which can model redundancies and synergies, see Section 2.4) was not yet implemented or fully operational.

One could argue that $F_{Cl.C}$ and $F_{Cl.B}$ are somehow redundant with F_{IP}. However, as explained previously, those features are less specific than F_{IP} but still more precise than F_{Rar}, and can better grasp the fact that rogue domains created during the same campaign can point to different server IP's with nearby addresses (which may belong to the same web provider). Moreover, we note that domains registered through the same Registrar on very close dates are usually hosted on nearby servers (as it is the case with domains 465709, 465706 and 465710 in Table 3.2 on page 50), which further justifies the choice of $F_{Cl.C}$ and $F_{Cl.B}$ as clustering features.

Prior to running the clustering analysis, we need to define appropriate distance metrics for all features.

3.3.1 Distance Metrics

IP addresses

Since feature vectors defined for F_{IP}, $F_{Cl.C}$, $F_{Cl.B}$ and F_{NS} are simply *sets* of IP addresses (or sets of IP subnets), it is relatively easy to calculate a similarity between two sets by using the *Jaccard coefficient* which is defined by following expression:

$$sim(i,j) = \frac{|S_1 \bigcap S_2|}{|S_1 \bigcup S_2|} \tag{3.1}$$

where S_1, S_2 are two sample sets.

For example, looking at domains in Table 3.2 (page 50):

- for domains 465709 and 465706, comparing IP addresses (F_{IP}) with the Jaccard coefficient gives a similarity equal to 0, whereas the same operation for $F_{Cl.C}$ and $F_{Cl.B}$ yields 0.5 and 1 respectively.
- idem for domains 334096 and 334091, where the similarity for F_{IP} is zero, but the IP subnets (Class C and B) are identical.

Registrant names

The most straightforward way of measuring a similarity between two registrant names is simply to check their equality. However, this does not allow us to grasp commonalities in the way registrant names (i.e., email addresses) have been created, e.g.:

- people can use automated tools to create new email accounts, usually offered by specific email providers (such as Gmail, Hotmail, Yahoo, etc), and use them afterwards to automatically register rogue domains;
- we have observed some recurrent patterns in the email addresses of certain registrants, like the use of specific keywords, or string tokens (often borrowed from the domain name being registered), probably also created in an automated fashion;
- a substantial amount of rogue domains have been created using third-party companies offering *domain privacy protection* services (e.g., `domainsbyproxy.com`, `whoisguard.com`, `privateregistrations.ws`, or eNom's "Whois Privacy Protection Service"). Consequently, comparing which specific *email domain* is used by registrants can reflect a certain strategy used by cyber-crooks (in this case, protecting their identity).

These considerations have led us to define a heuristic distance for comparing email addresses used by registrants. More precisely, for two given rogue domains:

(1) we start obviously by checking the equality between the two registrants addresses;

(2) when the value given in (1) is zero, then we further compare the three following sub-features: *email domain*, *username*, *presence of AV-keywords*. The final similarity value is then given by a weighted mean, defined by the following empirical weighting values: [0.2, 0.2, 0.5].

The latest sub-feature refers to the screening of about 30 commonly-used AV-keywords within the email addresses, such as $\{anti, virus, cleaner, remove, malware, spyware, \ldots\}$. When at least 2 different tokens are found, this sub-feature is equal to 1. This simple

heuristic distance proved to be effective in grouping registrants addresses that looked semantically related.

Let us consider a few examples from Table 3.2 (page 50):

- for domains 334096 and 334091, the similarity for F_{Reg} is equal to 1:
- for domains 334096 and 465709, the similarity is equal to 0.2 (same domain only);
- for domains 211552 and 122287, the similarity is equal to 0.7 (same domain and presence of at least two AV-related keywords);

Domain names

Regarding F_{Dom}, we need to catch commonalities between rogue domain names having similar patterns, or common sequences of the very same tokens, which can indicate that the very same tool has been used to dynamically create new names based on a given set of keywords. A commonly used technique for measuring the amount of difference between two sequences (or two strings) is the *Levenshtein distance*, also called the *edit distance*, which is used in various domains such as spell checking, information retrieval, text and web mining, or DNS sequence analysis.

The Levenshtein distance is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. It can be considered as a generalization of the Hamming distance, which is used for strings of the same length and only considers substitution edits.

The Levenshtein distance is zero if and only if the strings are identical, and the upper bound is given by the length of the longer string. Let us consider a few examples of distances between rogue domain names we have observed:

- the distance between `scan4lite.com` and `scan4life.com` is only 1, whereas the distance between `scan4lite.com` and `scan4safe.com` is equal to 3;
- the distance between `gofilescan.com` and `gofullscan.com` (resp. `fast4scan.com`) is 2 (resp. 6);
- in Table 3.2, the distance between `home-antivirus2010.com` and `homeanti-virus-2010.com` (resp. `homeav-2010.com`) is 3 (resp. 8);
- in Table 3.2, the distance between `home-antivirus2010.com` and `pc-anti-spy2010.com` (resp. `anti-malware-2010.com`) is 9 (resp. 12).

Since Levenshtein gives a distance (and not a similarity) metric, we still need to tranform those values into similarities, for example by using equation **??**. Similarly to the calibration procedure performed in Chapter **??** (on page **??**), we need to determine a constant σ that reflects the decreasing rate of the similarity as an exponential function of the distance.

By considering a large number of domain names in our data set, we observed that an edit distance of 5 or 6 was in most cases reflecting an *average similarity* between two rogue domains names, with a sufficient number of common tokens justifying a similarity value around 0.5, whereas a Levenshtein distance above 12 was clearly indicating that the two domains had almost nothing in common in their name schemes (or at least, no significant pattern in common). Based on those observations, we have derived an empirical value for $\sigma = 7$. The similarity values obtained from transforming some Levenshtein distances are showed in Table 3.3, which correctly translates our requirements regarding the measurement of similarities between rogue domain names.

Table 3.3: Transforming Levenshtein distances into similarity values using $\sigma = 7$ in equation 2.3.2.

Levenshtein	0	1	2	3	4	5	6	7	8	9	10	11	12
Similarity	1	0.98	0.92	0.83	0.72	0.60	0.48	0.37	0.27	0.19	0.13	0.08	0.05

Note that some other types of string or text distances could be used to better grasp commonalities among domain names. For example, we could try define a "token-based distance", which computes the number of common tokens between two domain names (based on a predefined list of commonly-used keywords), and normalizes this value by using the Jaccard coefficient (equation 3.1).

Another possiblity would be to rely on more complex distances, such as *semantic matching*, or certain metrics defined in *Latent Semantic Analysis* (LSA), to analyze the possible meaning of each domain name and consequently, to determine whether two domain names are semantically related (i.e., similarly to what a human expert tries to do). However, we leave the study of these options as future work, as the Levenshtein distance has performed very well on this data set, and meaningful clusters have been obtained with this metric.

3.3.2 Cluster Analysis

Based on the previously defined features and distance metrics, we have then applied the graph-based clustering to each individual feature, using the dominant set framework

introduced in Section 2.3.1.

An overview of the clustering results is given in Table 3.4, where we can compare the global performance of site features individually. As usual, the consistency of the clusters can be assessed through the average *compactness* value of all clusters, given by the column $\overline{C_p}$. We have also calculated an average compactness for the first 20 clusters, represented by $C_{p,20}$, as these are usually the most meaningful clusters found by the dominant sets algorithm. The column $\overline{size}$ refers to the average size of the clusters.

First, we can observe that the features related to IP addresses (F_{IP}) or to IP subnets ($F_{Cl.C}$ and $F_{Cl.B}$) provide apparently very compact clusters, i.e., very high C_p (even close to 1). This seems to indicate that rogue AV sites are located in a limited number of IP subnets (between 61 and 73% of all rogue sites could be clustered in only 110 to 192 clusters), and they tend to form very tight clusters. This can be observed in more details in Fig. 3.2 showing the individual compactness values for the first 20 clusters. Furthermore, a large majority of the sites is included in the first 20 clusters, as the evolution of the cluster size seems to show (curve in magenta in Fig. 3.2 (b), (c) and (d)). As we could expect, features related to IP subnets (Class C and B) give fewer clusters than F_{IP}, and they are also larger and more compact.

Table 3.4: Overview of the graph-based clustering results for the rogue AV data set.

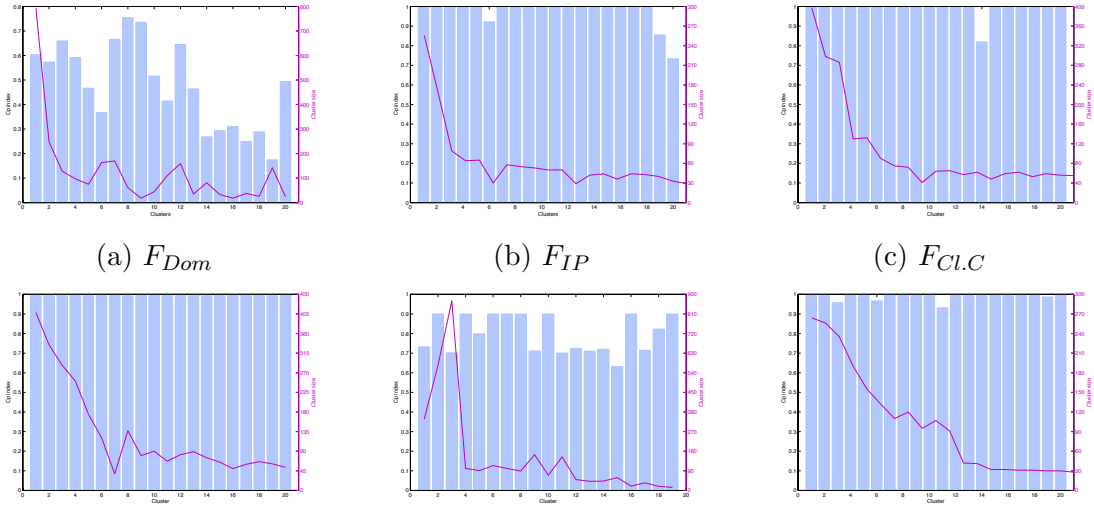
Feature	**Nr clusters**	**Nr sites**	$\overline{\textbf{size}}$	$\overline{C_p}$	$\overline{C_{p,20}}$
F_{Dom}	132	4,117 (70%)	31.2	0.46	0.53
F_{IP}	192	3,559 (61%)	18.5	0.82	0.98
$F_{Cl.C}$	110	3,667 (63%)	33.3	0.97	0.99
$F_{Cl.B}$	122	4,250 (73%)	34.8	0.98	1
F_{Reg}	19	2,937 (50%)	172.2	0.78	0.78
F_{NS}	40	2,529 (43%)	63.2	0.95	0.99

Regarding *domain names* (F_{Dom}), it is quite surprising to see that about 70% of this data set has been clustered based on commonalities in domain name patterns. The average compactness of the clusters is, admitteddly, a bit lower than for IP addresses, but still acceptable (see also Fig. 3.2 (a)). Still, the largest cluster found for F_{Dom} contains about 800 domain names. These results, in addition to some illustrative patterns of domain name clusters given hereafter, seem to confirm the intuition that miscreants are likely using very effective, automated tools to create large amounts of domain names with common patterns, in a bulk fashion.

Even more surprisingly, only 19 clusters of *registrant names* were found, accounting

for 50% of the data set. The mean cluster size for this feature is also the largest one, with (on average) about 170 rogue sites associated with the same registrant(s) ! The individual C_p values of the clusters, showed in Fig. 3.2 (e), indicate that all registant clusters are quite consistent, i.e., there are apparently very good reasons to group those rogue domains based on the registrant names. Here too, we hypothesize that cybercrooks are able to create new email accounts using highly effective, automated tools, and those new email addresses are then used to create and register new domain names quite anonymously. An in-depth analysis of the cluster patterns further confirmed this assumption, as illustrated hereafter with some detailed results.

Finally, looking at clustering results obtained w.r.t. *nameservers* (F_{NS}), we note that rogue domains are apparently not as much correlated than by the other IP-related features. The number of clusters found with this viewpoint is significantly lower (only 40 groups comprising totally 43% of the sites). However, those clusters are still very compact, but also larger than clusters of IP addresses or IP subnets.



(a) F_{Dom} (b) F_{IP} (c) $F_{Cl.C}$

(d) $F_{Cl.B}$ (e) F_{Reg} (f) F_{NS}

Figure 3.2: Compactness values of the clusters (in blue) and their respective sizes (in magenta) for the first 20 clusters found in the Rogue AV data set (5,852 rogue domains).

Some detailed results

To visualize the clusters, starting from the similarity graphs built for the various features, we have created 2-dimensional maps using a dimensionality reduction technique called t-distributed Stochastic Neighbor Embedding, or *t-SNE* [81]. t-SNE is a non-linear dimensionality reduction technique that aims at preserving as much of the significant structure of the high-dimensional data as possible in the low-dimensional map. That is, we can verify that two nearby data points on the 2-dimensional map have highly similar feature vectors, whereas two distant points should have nothing in common. This can be helpful to visualize a high-dimensional data set, but also to assess the consistency of clustering results.

For the sake of illustration, we have considered all rogue sites involved in the largest clusters only, which are the most representative of the kind of results we have obtained. Also, we illustrate some detailed results only for the four features F_{Dom}, F_{IP}, $F_{Cl.B}$ and F_{Reg}. However, very similar visualizations can be obtained for the other two features (F_{NS}, $F_{Cl.C}$), with similar interpretations regarding cluster patterns.

Let us consider Fig. 3.3 (a) (on page 61), on which we can visualize the 20 largest clusters obtained w.r.t. **rogue domain names** (F_{Dom}), and comprising 3,151 rogue sites. On this map, each pixel represents the domain name of a given site, and the pixel color refers to which cluster the site belongs to (notice the numbers around certain points to indicate the cluster id's). The relative interpoint proximities have been mapped to the inter-domain similarities, as calculated with the chosen distance metric (i.e., Levenshtein in this case). As we can observe, the overall structure of the map seems to indicate that there are basically two regions: *(i)* a region with well-separated clusters of domain names (like clusters 1, 11, 20, 7, 12); and *(ii)* a quite large and "messy" zone, where data points are somehow mixed and clusters overlap with each other (like clusters 2, 4, 6, 3, 5, 16, 19, etc).

This aspect can be easily explained by looking at the cluster patterns, like those explicitely given in Fig. 3.4 (a). In this table, some domain name patterns are represented using *regular expressions* for the variable parts of the names, whereas fixed string tokens (common to all domains of the indicated cluster) are highlighted in **bold**. This can help the analyst to understand, very quickly and via a single global picture, the inter-relationships among those domain name patterns. For example, cluster **1** (which contains about 794 sites) is made of domain names that are built with exactly 5 randomly-chosen alphanumeric characters, whereas cluster **11** (containing 110 sites) represents domain names made of 7 to 8 alphanumeric characters (most of them also randomly chosen). This does not mean, *per se*, that the two clusters are due to the same root cause (or the same tool); however, it already explains their mutual proximity on the map.

The same reasoning holds for clusters **7** and **12**, where some common string tokens (e.g., **scan** and **.info**) tend to tie those clusters close to each other (idem with clusters **2** and **8**). Regarding the fuzzy area in which we find clusters **4, 6** and many others as we move towards the top of the map, we observed that those overlapping clusters represent domain names with commonly-used words, made of 4 to 10 alphanumeric characters, and involving many *keywords* usually related to anti-malware or anti-spyware software, which explains why those domains are somehow inter-related with many others. In other words, the variability of those patterns combined with the numerous commonalities among some of them explains why the t-SNE algorithm had difficulties to clearly separate those domain names on a reduced map with only 2 dimensions.

As we could expect from the global clustering results, the two-dimensional maps obtained for the viewpoints related to IP addresses (Fig. 3.3 (b)) and IP subnets (Fig. 3.3 (d)) reflect the strong correlating aspect of those features. In both maps, we have considered the 20 largest clusters, which comprise about 1,269 and 2,589 rogue AV sites for F_{IP} and $F_{Cl.B}$ respectively. Most of the clusters are apparently well-separated. However, we can observe a few overlapping clusters, for example clusters **4** and **16** (in both maps), or clusters **7**, **9**, **10** in the $F_{Cl.B}$ map, which can again be easily explained by the corresponding patterns given in Fig. 3.4 (b),(d). We note also that clusters found w.r.t. $F_{Cl.B}$ are quite obviously much larger than those found with F_{IP}.

Even though IP-related clusters tend to form very tight clusters, we argue that this feature *alone* is probably not sufficient in many cases to identify a rogue AV campaign. In our opinion, all web sites hosted on the same IP subnet (or even on the same web server) are not necessarily created by the very same group of cyber-crooks. There are, apparently, several popular Web providers among those communities (probably for good reasons), and thus the same web server can perfectly host (malicious) domains created by different groups or communities.

Finally, we have represented in Fig. 3.3 (c) the 10 largest clusters obtained with F_{Reg}. Those clusters contain 2,501 rogue sites, and each data point on the map represents here the domain registrant. The patterns corresponding to those clusters are given in Fig. 3.4 (c), where variable parts of the registrants are again represented by regular expressions, and fixed tokens are emphasized in **bold** style. These registrant patterns can explain why we find a very large mixed cluster (1,197 points) in the center of the map, due to two inter-related clusters (**1** and **3**) that are composed exclusively of email addresses from the `gmail.com` domain. Idem with clusters **4**, **5** and **9**, but this time within the email domain `yahoo.com`. As such, observing two domains whose registrants use an email address of the same domain does not mean anything special. However, other clusters are well-separated, showing more interesting patterns like clusters **2** and **8** (`cn@id-private.com` and `cn@space.kz`), or cluster **6** and **17** for which we can see that

(a) Map of rogue domains clusters (F_{Dom})

(b) Map of rogue AV Server IP's (F_{IP})

(c) Map of Registrant clusters (F_{Reg})

(d) Map of IP subnets (Class B) clusters ($F_{Cl.B}$)

Figure 3.3: Visualizing clustering results for the rogue AV data set.

Id.	Cluster pattern
1	[a-z0-9] {5}**.cn**
11	[a-z0-9] {7, 8}**.cn**
7	**scan**[4|6] {lite, live, home, user ...}**.info**
12	{lite, live, home, user ...} [4|6]**scan.info**
20	{assist, beers, cds, cigars, sofas, ...}**online.cn**
2	{any, av, best, easy, fast, go, lite, ...}**scan.com**
4,6	[a-z0-9] {4, 10}**.com**
10	**adware**{2009, clean, alert, bot, pro, ...}**.com**
8	**goscan**{-pro, data, elite, file, gate, lite, ...}**.com**

(a) F_{Dom}

Id.	Cluster pattern
1	84.16.247.12, 89.149.236.145
2	209.44.126.102
7	209.44.126.241
11	69.64.155.119
4	64.191.92.197
16	64.191.92.197, 91.206.231.146
3	195.95.151.174
19	195.95.151.174, 195.95.151.138, 91.212.41.114

(b) F_{IP}

Id.	Cluster pattern
1,3	[a-z0-9]*__@gmail.com__
2,8	cn@id-private.com, cn@space.kz
4,5,9	[a-z0-9]*__@yahoo.com__
6	contact@privacyprotect.org
7	admin@mas2009.com
10	support@NameCheap.com
17	{ *AV-keywords* }__@domainsbyproxy.com__

(c) F_{Reg}

Id.	Cluster pattern
1	84.16, 89.149
3	195.95
5	63.146
7	209.44
9	209.44
10	209.44, 69.64
4	64.191
16	64.191, 91.206
20	210.51, 220.196, 222.73, 91.212

(d) $F_{Cl.B}$

Figure 3.4: Some cluster patterns found for each feature of the rogue AV data set. For clusters containing multiple string patterns (F_{Dom}, F_{Reg}), variable parts are represented with a syntax based on regular expressions, whereas fixed string tokens are highlighted in bold. For F_{Reg}, *AV-keywords* refers to a set of about 30 AV-related keywords, such as `adware`, `anti-`, `malware`, `spyware`, `av360`, `repair`, `tool`, `virus`, `bot`, `registry`, `remove`, ...

domain owner(s) also protect their privacy by using different *whois domain protection* services.

In conclusion, we note that most of the clusters obtained w.r.t. each site feature can reveal interesting and meaningful patterns revealing how those rogue sites have been created and managed. In fact, each feature can be seen as a viewpoint giving a certain *perspective* on the underlying phenomenon (or root cause), which in turn can also highlight interesting aspects of the modus operandi of the miscreants.

However, it becomes difficult (and time-consuming) to combine those viewpoints *manually* when the number of features increases, even when we rely on graphical visualiza-

tion techniques such as those presented here. Furthermore, the fact that rogue sites have been clustered w.r.t. *a given aspect*, or even two clusters that are lying in the same neighbourhood on a given 2D map, does not mean that the rogue sites under scrutiny are *necessarily* due to the same root cause. As explained previously, only one common feature can be merely due to a coincidence, to a commonly-seen pattern, or to a commonly-used technique.

To identify the underlying phenomena in a more systematic and reliable manner, certain clusters are likely to be merged whereas some others may have to be split. To aid the analyst to make such a decision, the multi-criteria aggregation component will be used to effectively combine all these viewpoints, such that the final result models the expectations of an expert regarding the (combination of) features that must be satisfied in order to attribute two rogue sites to a common root cause.

3.4 Multi-Criteria Aggregation

3.4.1 Defining parameters

We are now in a position to combine all site features using an *aggregation function*, with the purpose of identifying rogue AV campaigns whose rogue domains are automatically grouped together based upon common elements likely due to the same root cause.

As a first exploratory approach, we have used two different Ordered Weighted Averaging functions (OWA and Weighted OWA) as aggregation means, which have been described in Section 2.4. However, nothing forbids us from performing the very same analysis using more complex aggregation methods, such as the Choquet integral, if we need to model important synergies or redundancies among features.

Based on the feature analysis performed in Section 3.2, and considering our intuition on rogue AV campaigns, we have defined the following weighting vectors to be used in the (W)OWA aggregation:

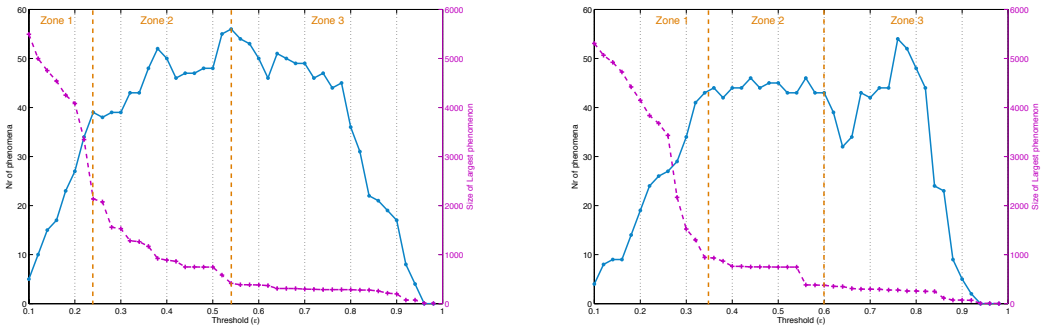
$$\begin{cases} \mathbf{w} &= [\,0.10,\ 0.10,\ 0.20,\ 0.30,\ 0.20,\ 0.10\,] \\ \mathbf{p} &= [\,0.20,\ 0.20,\ 0.15,\ 0.10,\ 0.25,\ 0.10\,] \end{cases}$$

By defining the weighting vector $\mathbf{w}$, we give more importance to criteria starting from the third highest position, which means that the two highest scores will have lower weights (0.10) and thus *at least three* strong correlations will be needed in order to have a global score above 0.3 or 0.4.

Regarding the weighting vector $\mathbf{p}$, we give a little more confidence to the features F_{Dom}, F_{IP} and F_{Reg}. Intuitively, we can reasonably assume that a combination of those specific features will yield a high probability that correlated rogue sites are likely due to

the very same campaign. On the other hand, we give a little less confidence to $F_{Cl.C}$, $F_{Cl.B}$ and F_{NS}, as these features are obviously less specific, and even somehow redundant with some of the previous features.

It is worth reminding that a strong advantage of these agregation techniques is that the analyst does not need to determine beforehand *which* set of features are the most relevant ones in the aggregation.



(a) OWA (b) WOWA

Figure 3.5: Sensitivity analysis of (a) OWA (b) WOWA aggregation techniques; to determine appropriate ranges of values for the threshold ϵ. The regions 1, 2 and 3 indicated on the graphs correspond to the 3 phases described in Fig. **??**. The axis on the right (in magenta) shows the size of the largest phenomenon (or rogue AV campaign) for each threshold value.

The last important parameter to define is the **decision threshold** ϵ, which is used to eliminate unwanted links in the combined graph, and thus also to identify connected components from it. As usual, we can determine an appropriate range of values for ϵ by performing a *sensitivity analysis*, i.e., we let ϵ increase from a very low to high value, and we observe the number of components (or MDCs) that we can find in the resulting graph, as well as the size of the largest MDC. This is illustrated in Fig. 3.5 where we can observe the impact of the threshold on (a) the OWA aggregation method, and (b) on the Weighted OWA. We have indicated on the plots three regions of interest in the determination of ϵ.

In the first region of the plot, edges with a lower weight in the aggregated graph are being removed, and large connected sub-graphs start to split into a number of more meaningful sub-graphs.

In the second region, we can observe a sort of plateau starting around the values $\epsilon = 0.25$ for OWA, and $\epsilon = 0.35$ for WOWA, which seems to indicate some reasonable

values for our decision threshold (as the number of large phenomena becomes stable). Even the size of the largest phenomenon becomes fairly stable at those threshold values (certainly with the WOWA operator).

Increasing further ϵ up to an excessive value can then lead to a significant loss of nodes and edges in the aggregated graph, which means that we also loose a lot of information and thus few semantics can be derived on the phenomena under study from the MDCs. We observe also that an appropriate range of values for ϵ lies usually around k/n, with k the minimum amount of correlated features desired by the analyst to link two events.

3.4.2 Results overview

In Table 3.5, we briefly compare the performance of each aggregation method. Overall, we note that the two sets of phenomena found by the two techniques are consistent with each other, both in terms of size as well as with respect to the composition of each phenomenon. As it could be expected, the WOWA technique performed a little better than OWA, which is due to the use of a second weighting vector $\mathbf{p}$ that models the reliability of site features.

Table 3.5: Comparison of OWA and WOWA aggregation methods for the rogue data set.

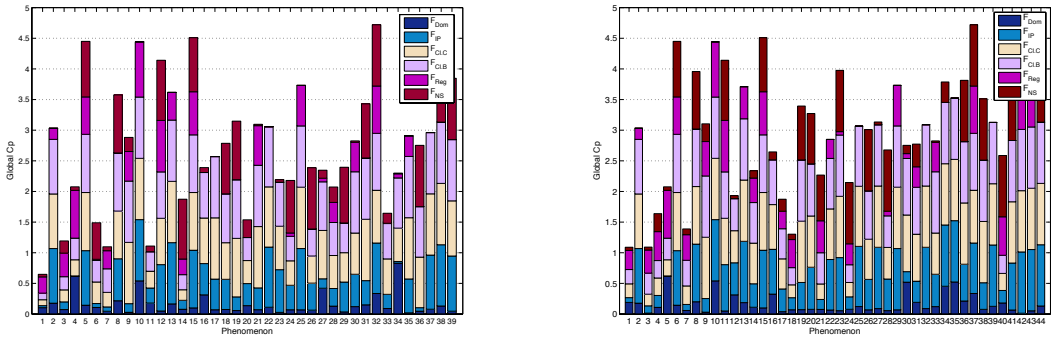
Characteristic	**OWA**	**WOWA**
Threshold ϵ	0.30	0.35
$\lvert\mathcal{P}\rvert$	161	173
$\lvert\mathcal{P}\rvert$, with $\lvert P_i\rvert \geq 10$	39	44
Nr of sites	4,049 (69%)	3,586 (61%)
Average C_p	0.46	0.51

To further evaluate the consistency of the results, we have represented in Fig. 3.6 the global graph compactness C_p for the largest phenomena, as calculated individually for each feature. This interesting view can be used to determine which features tend to group rogue sites together within each phenomenon P_i. First, we note that most of the phenomena have globally high compactness values, except for a few ones (such as P_1 and P_3 found with OWA). A deeper inspection of those phenomena reveals that these are also very large connected components, which explains why they are less compact since they are usually made of several loosely connected subgraphs.

Quite naturally, we see also on Fig. 3.6 that IP-related features contribute the most to the correlation of rogue sites. In many cases, F_{Reg} seems to complete or reinforce those

correlations. It is also interesting to see that some phenomena are correlated by $F_{Cl.C}$ and $F_{Cl.B}$, but not by F_{IP} (such as P_{27} found with OWA), which justifies the selection of those features.

F_{Dom} has in general lower C_p values, but there are still 3 or 4 phenomena in which domain name correlation plays a significant role (like for P_4, P_{10} and P_{34} found by OWA). Finally, we observe that each P_i can be characterized by varying degrees of correlation regarding each feature, but overall there are always at least three different features having a high correlation (except for P_1, the largest phenomenon comprising more than one thousand sites).



(a) OWA

(b) WOWA

Figure 3.6: Evaluation of the results consistency using the global compactness (C_p) of the largest phenomena found using (a) the OWA aggregation; and (b) the WOWA aggregation. Each color refers to the C_p index of each site feature individually.

3.5 Analysis of Rogue AV Campaigns

In this final Section, we present a more in-depth analysis of some illustrative case studies, in order to show the kind of insights we can get into the behavior of so-called *Rogue AV campaigns* (shortly written RC's in the following), which were identified, in a systematic and automated manner, by our multi-criteria clustering technique.

The different RC's that are studied in this Section are summarized in Table 3.6. It is worth mentioning that these experimental results are based on case studies that have been presented in [73, 19].

Table 3.6: High-level characteristics of some typical Rogue AV campaigns (RC's).

RC	# sites	# Reg.	# Registrar	# IP's	# Class B	Timespan	Server countries
3	438	115	6	50	15	03 Oct 2008 - 03 Jun 2009	UA, CN, KY, SG
4	752	4	1	135	7	22 Jun 2008 - 27 Feb 2009	US, DE, BY
5	310	17	1	13	5	17-20 Oct 2008	CN, DE
27	15	3	1	8	4	25 Jun - 14 Jul 2009	US
34	14	3	1	19	2	29 Jul 2009	US

3.5.1 PC-Security and PC-Anti-Spyware campaigns

As a first illustration, we present two relatively simple yet interesting phenomena identified TRIAGE, namely RC 27 (Fig. 3.7) and RC 34 (Fig. 3.8). In those figures, the domain names are shown in light blue, the web server /24 subnets in yellow, nameservers in purple, and the email address of the Registrant in red. Double-edged purple boxes indicate servers with co-located DNS and web servers.

Both RC's are composed of a small number of rogue sites, and these are mostly correlated by server IP addresses and by common patterns in the domain names (notice that this is consistent with the assessment of graph compactness in Fig. 3.6). Note also that all domain names are clearly referring to anti-virus or anti-spyware software "products".

Although the two RC clusters initially appear to be distinct, they present a number of similarities:

- Both clusters use the exact same domain naming scheme (except that one uses "spyware" while the other uses "virus");
- All of the domains in each cluster use the same registrar (OnlineNIC) and are serviced by the same two ISPs;
- The email addresses of all domain registrants are in ".ru" domains;
- The servers were on consecutive IP addresses;

Perhaps even more conclusively, we found that the content of each site was identical, with the exception of one differing image (recall that the site content was not a feature of our clustering system).

In fact, cyber-crooks used for both RC's a single registrar (OnlineNIC[5]) which apparently offers the free use of their registration API/template system. The similarities described here above strongly suggest that the task of registering, creating, and hosting

[5] `http://www.onlinenic.com`

these rogue security software domains was automated and that the same entity may be responsible for both clusters.

Also worth noting is that both clusters are split between two different ISPs, suggesting an attempt to provide some level of redundancy in case a cluster is taken offline by the ISP. Finally, we observed that all of the hosting web servers were located in the US.

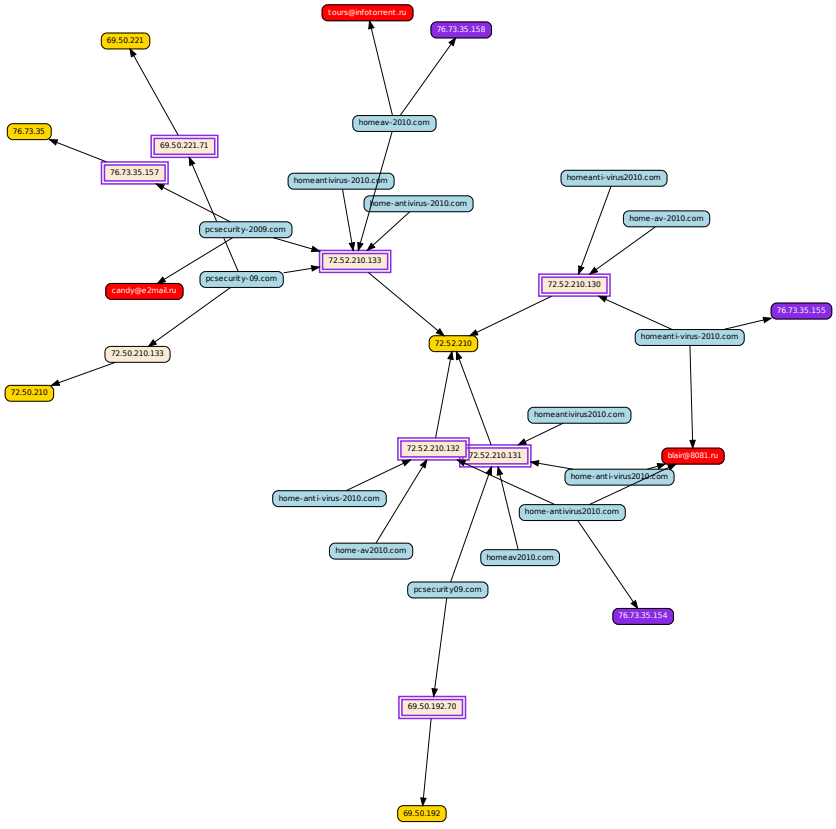


Figure 3.7: RC27: a rogue campaign related to Anti-virus2010.

3.5.2 Two different large-scale campaigns within the .cn TLD

The TRIAGE attribution process has also identified some other clusters that represent more sophisticated campaigns. Two such examples are RC 4 and RC 5, which are two

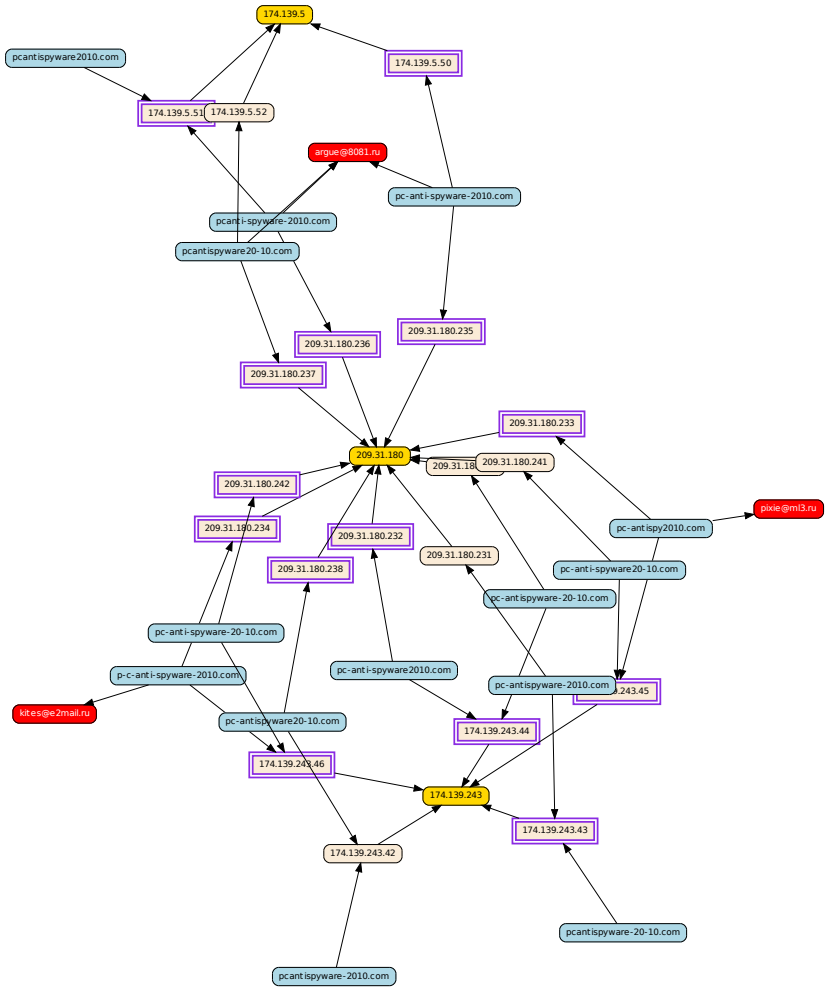


Figure 3.8: RC34: a rogue campaign related to Anti-spyware2010.

different campaigns observed within the **.cn** top-level domain (TLD). Regarding cluster RC 5, about 310 .cn domain names were registered in only three days, as represented in Fig. 3.9 on page 75. The domain names (in blue) point to 13 IP addresses residing in five subnets (yellow) and were registered by a number of Web-based email addresses (red) in three days (purple). The prevalent use of popular Web-based email accounts

(e.g., `yahoo.com`, `gmail.com` and `hotmail.com`) to register these domains is assumed to be because these email services are easily anonymized.

Interestingly, all of the domain names in RC 5 are referring to various popular web categories, such as games, fun, e-cards, casino and even porn sites, but apparently not a single domain name seems to relate to AV software. However, since they have been included in our rogue data set, they were still somehow related to rogue AV. One could think that some of those web sites are possibly "legitimate" ones that have been compromised, so that they could serve rogue AV software. Note also that we found many of these sites were also hosting malware (e.g., trojans, fake codecs). Considering the abnormal pattern showing the registration of all those domains "in bulk", a more likely explanation is that *(i)* cyber-crooks try to diversify their activities, by hosting different types of threats, and *(ii)* they want to optimize the monetization of their web sites by attracting as many users as possible with popular web categories. Finally, all of the domains have been registered at a single Chinese registrar (Era of the Internet Technology), and 97% of the web servers are located in China.

In the next cluster, RC 4, about 750 rogue domains have been registered also in the **.cn** TLD (resolving to 135 IP addresses in 14 subnets), on eight specific dates over a span of eight months (Fig. 3.10 on page 76). However, differently from RC 5, the majority of the IP addresses of the hosting servers (pointed to by these domains) were hosted in the United States, Germany, and Belarus. In fact, no server could be identified as being located in China[6].

Like in RC 5, the same Chinese registrar (Era of the Internet Technology) was used by cyber-crooks for the registration of all domain names. However, differently from RC 5, all of the domain names are composed of exactly 5 alphanumeric characters, apparently chosen in a random fashion, which again indicates the use of automated tools to create those domains. Finally, a noteworthy characteristic of this RC is that the registrant responsible for 76% of the domains makes use of a *whois domain privacy protection* service (`cn@id-private.com`), which is also a commonly observed characteristic in certain rogue campaigns.

3.5.3 An affiliate-based rogue network

Our multi-criteria method has identified RC 3, a rogue network showing an even more complex structure, as represented in Fig. 3.11 on page 77. In this cluster, more than

[6]It should be noted that the .cn top-level domain (i.e., the domain designation for China) has no registration restrictions, and non-Chinese based operators can easily register .cn domain names for a very cheap price.

430 rogue sites (in blue) are forming some sort of "bubbles" that are interconnected by common registrants (in red) and common hosting servers or IP subnets (in yellow). We hypothesize that this weird and complex network of rogue AV websites is likely to reflect the affiliate-based structure used by cyber-crooks to propagate rogue security software. Different reasons support this assumption:

- there is a large number of registrants, and most of them are responsible for a single domain;
- the domains are registered at 6 different registrars, which are quite popular for hosting malicious web sites;
- besides rogue AV names, many other domain names are also associated to other web categories that are often used for hosting dubious or suspicious websites, e.g.: online pharmacy (like `pharmacyeasy.cn`), casino (like `smilecasino.cn`), porn sites (like `hot-girl-sex-tube.com`), and there are even a few examples of typo-squatting web sites (like `disenyworld.com` or `rapidhsare.com`). This indicates a possible *diversification of activities* maybe performed by different affiliates;
- many other types of web threats have been found on a significant number of those sites (almost 50% of them), which can again indicate that affiliates attempt to monetize their domains by serving other malicious purposes as well;
- in this cluster, the numerous commonalities found among subsets of rogue sites indicate that several groups of affiliates are probably reusing the very same techniques, e.g.: creating and registering new domains at the same registrar or ISP (which does not really care about suspicious domain names), reusing the same browser exploits for hosting drive-by downloads, or serving the same malware or trojan (only 8 unique MD5 were found among all malicious binairies hosted on these web sites).

Also worth noting is that this cluster RC 3 has been observed in a span of time of 8 months. However, we observe once again that most of these sites are being registered in large groups during three phases, each one having a timespan of only a few days, which obviously requires a high level of coordination.

Finally, the geographical location of the web servers has also an interesting pattern, with 42% of the servers in Ukraine (UA), and the rest of the servers is spread in China but also in Cayman Islands (KY), Singapore (SG), and a few of them in Latvia (LV). We note again the prevalent use of Web-based email domains for registrants (more than 40% of gmail.com and about 26% of yahoo.com).

3.6 Lessons learned and countermeasures

This study leverages the analysis of real data to shed some light on the characteristics and dynamics of a specific threat landscape, that of rogue security software. We identify the specificities of such threat landscape and their foundations in a particularly favourable market. Such knowledge has direct repercussions on nowadays security practices, and helps underlining weaknesses in currently employed techniques as well as potentials for new research avenues.

Users. Despite of a minor number of cases in which rogue AV domains were observed also in association to other type of threats such as drive-by downloads, the main propagation vector for this type of threat is the psychological impact on the user. The in-depth study of the reasons for the successfulness of the interaction between victims and rogue campaigns is out of the scope of this work, but our analysis clearly shows that users have an important role in the successfulness of rogue AV campaigns. As suggested in [**?**], the cost-benefit trade-off associated to the offering of security services is often badly received by the users, that tend to reject the necessity of performing monetary investments to be shielded from hypothetical security threats. Rogue security software leverages this social reality to its own advantage. Increasing user awareness on the cost implicitly associated to security *may* have an impact on the relatively high conversion rates observed in this study, and *may* impact the return on investment associated to rogue AV campaigns.

Blacklisting is strained. Our study revealed two characteristics of the infrastructure used to spread rogue AV that have important consequences on the effectiveness of countermeasures against this threat, and, specifically, of blacklisting, a technique commonly used to prevent end users from accessing malicious resources.

As described in Section 3.1.2, the rogue AV infrastructure comprises both servers that exclusively host a very large number of rogue AV sites and servers where rogue AV sites coexist with legitimate ones. This situation is a worst case for blacklisting. In fact, IP-based blacklisting (where access to a specific web server IP is blocked) is bound to generate many false positives, thus preventing users from visiting benign sites that happen to be hosted on server IPs that also serve malicious sites. In fact, a naive IP-based blacklisting approach, listing all the servers we identified, would have incorrectly blocked access to 129,476 legitimate web sites. Conversely, domain name-based blacklisting (where access to a specific domain is blocked) is undermined by the easiness with which malicious actors can register large batches of domains. The registration of hundreds of automatically generated domain names observed in the different campaigns is likely to be an active attempt to evade such lists. For example, 77 of the rogue-specific

servers that we tracked were associated with more than twenty different domains, with a maximum of 309 domains associated to a single server.

Taking-down rogue AV campaigns. What would be a good strategy then to effectively fight rogue AV campaigns? Through an analysis of the victim access dataset performed in [20], it appears that taking down payment processing sites could help stop emerging rogue AV campaigns. Indeed, payment sites appeared to be far less in number than other rogue AV sites (we showed in [20] that 7 payment sites supported almost 200 front-end "scanning" sites) and seemed to change less frequently. Furthermore, by disrupting the sites generating revenue, defenders are likely to significantly affect also other parts of the rogue AV operations (*e.g.*, registering new sites and paying for hosting).

DNS-based threat detection. This study has highlighted once more the important role of the DNS infrastructure in Internet threats. Rogue AV campaigns often rely on misleading DNS names to lure victims into trusting their products (*e.g.*, *pcsecurity-2009.com*). Also, we have seen how such campaigns often lead to the automated deployment of large numbers of domains pointing to a few servers and following well-defined patterns in their naming schema. For all these reasons, as already noted in [**?**] for other type of threats, DNS seems to be a promising point of view for the detection of such anomalies.

Finally, even though we have used TRIAGE as a means to attribute known malicious sites to rogue AV campaigns, we envision an extension of these techniques by which new web sites that are likely to be part of a rogue AV campaign could be identified proactively. For example, we may attempt to classify a newly registered domain into one of the rogue AV clusters that we have identified during this study. A positive classification (i.e., the site's network observables are similar to those used in a campaign) would indicate that the site could have been registered by the same individuals responsible for the campaign, and, thus, would warrant further inspection or close monitoring of the domain.

3.7 Summary

In this experimental application, we have performed a longitudinal analysis of the infrastructure and the dynamics associated with an increasingly popular threat, that of rogue security software. By attributing rogue web sites to common root causes based upon a series of common features, we have showed how TRIAGE could provide a unique perspective on how rogue AV campaigns and their server-side components are effectively

organized, created and managed. Starting from nearly 6,000 suspicious domains, we could identify about 40 campaigns that were quite likely coordinated by criminal organizations. An in-depth analysis of some campaigns highlighted the kind of insights we could obtain on the behavior of those organizations, in a systematic and automated manner.

These results can be leveraged in several ways. First, they give a more explanatory description of the rogue AV or client-side threats, in which, for example, individual, disconnected sites are substituted by sets of related sites in which time relationships (e.g., creation dates) are more explicit. Second, campaign-level information reveals the *modus operandi* of the criminals orchestrating the campaign, i.e., how they register the domains, what are their hosting partners, the duration of their efforts, the sophistication of the tools available to them (e.g., to automate the registration of domain names), and the countermeasures they employ against take-down efforts. Finally, the patterns discovered by this TRIAGE analysis could yield means for identifying additional rogue AV sites pro-actively or reactively, for example through a closer monitoring of DNS registration patterns.

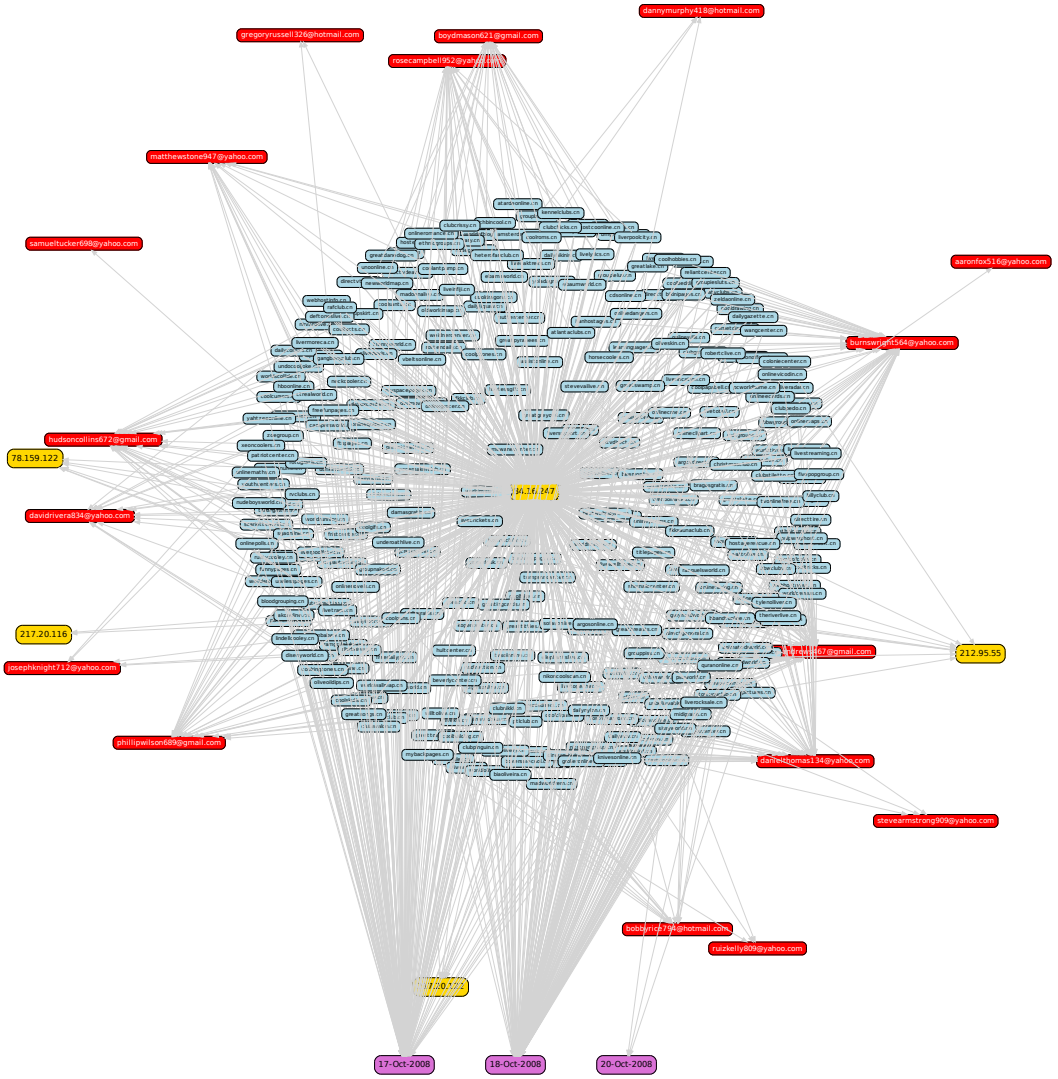


Figure 3.9: RC5: A rogue AV campaign within the .cn TLD.

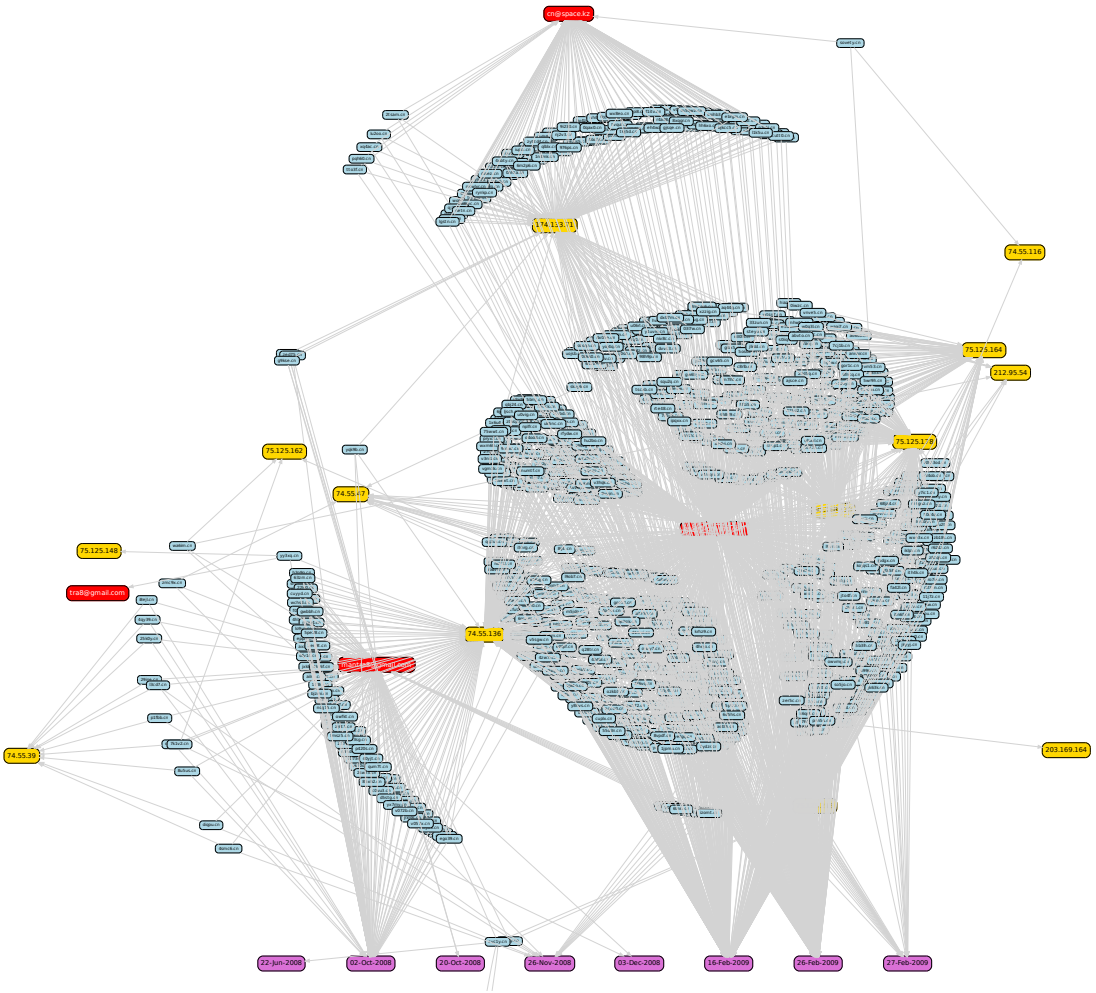


Figure 3.10: RC4: A different rogue AV campaign within the .cn TLD.

Figure 3.11: RC3: An affiliate-based rogue network.

4 Analysis of Allaple Variants

4.1 Introduction

Polymorphic techniques have introduced a new challenge in malware analysis: it is often difficult to discern the instance of a known polymorphic malware from that of a newly encountered malware family, and to evaluate the impact of patching and code sharing among malware writers to prioritize analysis efforts. According to [14], the amount of samples submitted to VirusTotal [82], a popular virus scanning platform, is in the order of millions of samples per month. Such numbers translate into a daily load of approximately 30,000 samples per day. This load can be partially explained by the easiness with which malware writers can generate new code by personalizing existing code bases, or by re-packing the binaries using code obfuscation tools [52, 45, 85]. In addition, malware sample counts are biased by the increasing usage of polymorphic techniques [8].

In order to investigate the feasibility of using TRIAGE in combination with SGNET data, we have decided to focus our attention on a well known phenomenon, already investigated from different perspectives in previous WOMBAT deliverables: the `Allaple` worm outbreak [67, 24]. Discovered in late 2006, `Allaple` was one of the first widespread polymorphic malware: the worm mutates its content at every propagation, in an attempt to evade classical signature-based detectors. In the context of this deliverable, we propose a streamlined process for the analysis and generation of intelligence on a very specific process. In the context of WP4 we have explored several solutions to tackle the analysis and the clustering of `Allaple`-related polymorphic malware samples:

- In D16 (D4.2) we have shown how the clustering of behavioral profiles obtained through the execution of malware in sandboxed environments could be of great help to the problem of malware polymorphism. More specifically, the work in [7] showed how similar methodologies could successfully group the execution of Allaple samples into two main behavioral groups associated to two known variants of this worm. In the very same paper, we have once more underlined the large amount of inconsistencies associated to standard AV labeling (as originally reported in [14] and [3]).

- In D17 (D4.4) we have presented the results obtained by applying a highly scalable and simple technique to cluster malware samples according to their structural characteristics, called "feature-based" malware clustering. We have shown how the current level of sophistication of polymorphic engines is such to allow the identification of different malware recompilations by simply looking at the content of specific portions of the PE headers.

- In D18 (D4.6) we have described the contextual information generated by the SGNET honeypot deployment for each collected malware sample. We have shown how SGNET can provide highly valuable insights on the population infected by a specific malware variant, as well as detailed information on the structure of its propagation strategy.

- In D21 (D4.7) we have underlined the complementarity of all the previously described approaches at generating a more complete picture of the malware landscape. We have presented in [40] an empirical study showing the importance of factoring together multiple approaches and multiple malware clustering techniques to 1) detect the limitations of a specific clustering approach; and 2) generating additional semantics for the malware analyst that would not be apparent when considering each approach in isolation.

SGNET Dataset

The SGNET deployment was described in detail in D13 (D3.3) [42, 39]. In contrast to other malware collections, SGNET focuses on the collection of detailed information on code injection attacks and on the sources responsible for these attacks. This is made possible through a layered approach, that focuses on the emulation of each stage of a code injection attack by building upon a model initially introduced in [21] called EGPM. The EGPM model structures each injection attack into four distinct phases:

- **Exploit (ϵ).** The set of network bytes being mapped onto data which is used for conditional control flow decisions. This consists in the set of client requests that the attacker needs to perform in order to lead the vulnerable service to the failure point.
- **Bogus control data (γ).** The set of network bytes being mapped into control data which hijacks the control flow trace and redirects it to somewhere else.
- **Payload (π).** The set of network bytes to which the attacker redirects the control flow through the usage of ϵ and γ.

- **Malware (μ).** The binary content uploaded to the victim through the execution of π, and that allows the attacker to run more sophisticated operations that would be impossible in the limited space normally available to the payload π.

On top of code injection information, the SGNET dataset is structured to contain a variety of information on the context of a code injection event, on the location of the attacking source, its configuration (obtained through passive OS fingerprinting) and on the location of the victim honeypot targeted at a specific point in time. The interested reader can find in D18 (D4.6) a comprehensive description of the information collected in the SGNET dataset.

In the context of this experiment, we have focused on a peculiar propagation characteristic that is known to be associated to the spread of the `Allaple` worm: the shellcode is designed to force the victim system to open a TCP port on a specific port (9988) over which the malware sample is uploaded. We have default selected a very specific shellcode behavior, and looked at all occurrences of such behavior since the very beginning of the SGNET deployment. This led to the identification of 10,162 SGNET events (i.e., *code injections*) related to the `Allaple` propagation in the period from September 13, 2007 until December 31, 2009.

4.2 Selection of Malware Features

Table 4.1 summarizes the features taken into consideration for the analysis. Mirroring the results obtained in the context of WP4, we take into consideration three main classes of information: information on the propagation context, information on the malware structure, and information on the malware behavior as generated by running each collected malware sample in the Anubis sandbox for a period of 4 minutes. The rightmost column in Table 4.1 represents the cardinality of each feature, that is the number of distinct values associated to all the code injection attacks taken into consideration in this context.

By looking in depth at each feature, we can infer some insights on their ultimate usefulness at understanding the phenomenon under analysis. From this set of features, we have thus selected a subset of malware-related features (both static and dynamic) to perform an multi-criteria analysis with TRIAGE. Recall that the purpose of such an analysis is to see if TRIAGE can help an analyst to distinguish the various code variants belonging to the same polymorphic malware family, but also to better understand the structures of those variants and their inter-relationships (e.g., commonalities and differences between variants due to patching, recompilation with different packers, adding of new functionalities, etc). Hence, a preliminary analysis of the features has guided us

Contextual features		
Event timestamp	Time of occurrence of the code injection attack.	10,160
Source address	IP address of the attacking source.	7,200
Source AS number	Autonomous system number of the attacking source (when available).	848
Source country code	Mapping of each attacking source to a country of origin by means of the Maxmind database.	110
Destination address	Address of the honeypot that was targeted by the attack.	50
Source/destination address overlap	The set of shared bits between source and destination address. The feature can be effective to detect bugs in the random number generator of the malware sample, or in detecting localized propagation strategies.	-
ScriptGen exploit classification	By leveraging the protocol learning techniques used in SGNET [43] we can classify the network interaction and discern network interactions likely to be associated to the activity of different exploit implementations.	39
Static malware features		
MD5 hash	The MD5 hash of the malware sample content. It is worth noting that the full hash content can be used to identify non-polymorphic malware samples, that do not mutate the binary content upon propagation.	7,379
File size	The size in bytes of the malware sample file.	409
File type	The file type as classified by libmagic.	3
PE section names	The name of the different sections defined in the file header according to the Portable Executable (PE) format (when applicable).	20
PE DLL imports	Name of the DLL files to be imported as specified in the Import Address Table (IAT) in the PE format.	2
PE kernel32 symbols	List of symbols to be imported from kernel32.dll as specified in the IAT.	6,386
PE linker/OS version, machine type	Version number for the linker and OS, as well as machine type, as specified in the PE header.	3/1/1
Dynamic malware features		
Anubis execution summary	Set of high level operations performed by the malware upon execution in the Anubis sandbox. For instance, the set contains the flag *IRCBOT* if the malware sample is seen generating IRC traffic towards one or more destinations throughout its execution.	25
Anubis syscall types	Set of types of system calls triggered by the malware sample during execution. For instance, the type *registry* is associated to any system call invocation related to Windows registry manipulations.	13
Anubis syscall targets	Extension of the previous feature meant to include also the full name of the target of the system call operation performed by the malware sample. For instance, a Windows registry operation is associated here to the type *registry*, as well as to the name of the registry entry being affected by the modification.	1,165

Table 4.1: SGNET features available for the Allaple analysis (right column represents cardinality)

to select certain characteristics instead of others, based on their expressiveness or their "potential" to highlight meaningful similarities and differences between code variants.

4.2.1 Static malware features

As already suggested here above, certain malware features related to the static analysis of the binaries may obviously reveal some interesting information. For each code injection, we have in particular selected the following features to be included in this TRIAGE analysis:

- F_{mw_md5}: the MD5 hash of the malware sample, which can help to identify non-polymorphic malware samples.
- F_{mw_size}: the size of the malware sample (in bytes).
- $F_{pe\_sections}$: the set of section names defined in the Portable Executable (PE) file header. For example:

  `.text\x00\x00\x00|rdata\x00\x00\x00|.data\x00\x00\x00`

- F_{pe_linker}: the version number of the linker used for the compilation of the malware sample (e.g., `92`).
- $F_{pe\_kernel32}$: the set of symbols that are imported from `kernel32.dll`, as specified in the Import Address Table (IAT). For example:

  (`GetProcAddress`, `LoadLibraryA`, `CreateActCtxA`, `GetComputerNameA`)

4.2.2 Dynamic malware features

There are mainly three behavioral features that are of particular interest for the analysis of the Allaple variants:

- $F_{bh\_optypes}$: the set of high-level system call types triggered by the malware sample during execution in the Anubis sandbox. For example:

  `network|service|process|section|random|sync|thread|registry`

- $F_{bh\_summary}$: a sequence of high-level behaviors performed by the malware upon execution in the Anubis sandbox. For example:

  `addressscan|file_modification_destruction|all_reg_activities`

- $F_{bh\_fullops}$: an extension of *bh_optypes* that includes the type as well as the full name of the target of the syscall operation performed by the malware sample. For example (for conciseness, some of the entries have been omitted):

```
file|C:\WINDOWS\system32\shell32.dll,
file|C:\WINDOWS\system32\urdvxc.exe,...,
network|AddressScanner_TCP,network|IcmpScanner,...,
process|C:\WINDOWS\system32\urdvxc.exe,...,
registry|HKLM\SYSTEM\CURRENTCONTROLSET\SERVICES\TCPIP\PARAMETERS\WINSOCK,...
```

4.2.3 Other features

Exploit

The *exploit* information, despite being associated to multiple FSM traversals, has limited usefulness in this specific context. An in-depth analysis of the traversals showed that they were all associated to the same vulnerability, namely the ASN.1 vulnerability (MS04-007) exploited on TCP port 139. The existence of multiple traversals is biased by the fact that, in this specific exploit, the IP address of the destination is part of the application-level payload. This tricks the ScriptGen algorithm (that approaches application-level payloads in a completely protocol-agnostic way) to generate traversals that match exploits targeting only specific address ranges. The exploit information is therefore not considered in this specific analysis.

Anti-virus signatures

Each malware sample collected in SGNET is automatically submitted every day to VirusTotal [82] for a certain period of time, according to a given scheduling policy. This means that we have at our disposal the AV signatures of most of the samples considered for this analysis. These signatures could have been included as additional features in the MCDa analysis. However, many research groups (including previous work carried out in the project) have pointed out the unreliability of AV labels at consistently grouping malware variants [7, 14, 3].

Nevertheless, we have kept these AV signatures as "labeling features", as a baseline to analyze our experimental results and to compare the structure of the Multidimensional Clusters (MDCs) given by TRIAGE with the labels given by various well-known AV products, as obtained from VirusTotal [82].

Propagation context

Finally, our analysis also revealed that *contextual features* (such as the origins of infected machines, the targeted sensors or the timestamps of the events) were apparently not sufficiently discriminant to be included in the MCDA analysis. That is, code injection attacks related to the Allaple propagation and shellcode behavior were apparently always coming from the very same networks, and were observed by the very same sensors, and this almost continuously in the whole analysis timeframe. For this reason, we decided to use in a first-stage analysis only the most relevant *code features* (static and dynamic) to try to highlight the different malware variants. Then, we used the contextual features in a second-stage analysis, to find out if we could get another useful perspective on the propagation context of those Allaple variants.

4.3 Multi-Criteria Analysis

4.3.1 Graph-based Clustering

Based on the feature analysis here above, we have performed a graph-based clustering of the 8 code-related features described in previous Section. Regarding distance metrics, we have used only two different distances:

(i) a simple equality for all feature vectors made of a generic value (or an ordered sequence of values), i.e, for:

$$F_{mw\_md5}, F_{mw\_size}, F_{pe\_sections}, F_{pe\_linker}, F_{bh\_optypes}, F_{bh\_summary}, F_{bh\_fullops}$$

(ii) the *Jaccard* distance (equation 3.1) for the feature vectors of $F_{pe\_kernel32}$, which are *sets* of values.

An overview of the clustering results is given in Table 4.2, where we can compare the underlying structure of each malware feature separately. Quite obviously, all features represented by generic values and involving a very simple distance metric (i.e., the simple equality) are very straightforward to cluster, since it boils down to executing simple queries directly on the TRIAGE database. For $F_{per\_kernel32}$, however, we have used again the dominant sets algorithm, which is more appropriate in this case as similarity values within the edge-weighted graph involve continuous values in the interval $[0, 1]$.

In Table 4.2, it is interesting to see that apparently not all malware samples are polymorphic ones. Regarding F_{mw_md5}, there is indeed one big cluster of 1,284 samples having the very same MD5 hash. Besides that cluster, there are also a couple of other

Table 4.2: Overview of the clusters by feature for the Allaple data set.

Feature	Nr clusters	Nr samples	$\overline{\text{size}}$	Max. size	$\overline{C_p}$
F_{mw_md5}	71	2,471 (24%)	34.8	1,284	1.0
F_{mw_size}	141	9,729 (95%)	69	2,395	1.0
$F_{pe\_sections}$	15	10,147 (99%)	676.5	7,578	1.0
F_{pe_linker}	3	10,155 (99%)	3,385	8,510	1.0
$F_{pe\_kernel32}$	1	3,719 (36%)	3,719	3,719	1.0
$F_{bh\_optypes}$	8	7,715 (75%)	964.3	4,755	1.0
$F_{bh\_fullops}$	156	6,440 (63%)	41.3	1,388	1.0
$F_{bh\_summary}$	21	7,693 (75%)	366.3	3,442	1.0

ones of moderate size (in the order of few hundreds); however, all other MD5 clusters are rather small in size.

With respect to F_{mw_size}, there seem to be some very popular binary sizes (the largest cluster contains not less than 2,395 samples of the same size), which is quite surprising for a data set comprising a large amount of polymorphic malware samples. The two most popular binary sizes are 57,344 and 57,856 bytes. For the section names defined in the Portable Executable ($F_{pe\_sections}$), we can see that there are only a few different patterns, with one particular section pattern that encompass about 75% of all data set samples (`.text\x00\x00\x00|rdata\x00\x00\x00|.data\x00\x00\x00`).

About the linker versions used to compile the malware binaries, we see that only three different linkers seem to be used for this malware data set. More than 85% of the samples have been compiled using linker version `92`, about 14% with linker version `96`, and only a few ones with a less popular linker (version `140`).

The structure of feature $F_{pe\_kernel32}$ is somehow different and quite interesting: TRIAGE has found only one large cluster of samples having the same set of symbols imported from `kernel32.dll` (i.e., `GetProcAddress`, `LoadLibraryA`). All other malware samples have apparently randomized sets of symbols, which is probably due to the polymorphic nature of the code.

Regarding the behavioral features, we can see that $F_{bh\_optypes}$ and $F_{bh\_summary}$ have fairly similar structures: between 34 and 47% of the samples have apparently exactly the same behavior when executed in the Anubis sandbox. The most popular patterns for these high-level behaviours are given here below, together with their cardinality. A deeper analysis of all other clusters revealed that most of these summarized behaviours are fairly similar, with minor modifications in the sequences of high-level operations; however, they might still be useful to discriminate different malware variants.

$F_{bh\_optypes}$

network|service|process|section|random|sync|thread|registry|file|time	4,755

$F_{bh\_summary}$

addressscan|file_modification_destruction|all_reg_activities|win_dir_copy	3,442
addressscan|file_modification_destruction|all_reg_activities	2,382

Finally, when we consider more detailed behavioral information, with the full name of the target of the syscall operation performed by the malware ($F_{bh\_fullops}$), we can see that this feature generates much more clusters with diverse patterns. Even though the high-level behavior is usually quite similar, differences can be detected when looking at lower level patterns. While some of these differences are normal artefacts generated by the differing execution contexts, others may be active attempts of the malware samples to somehow randomize their behavior (e.g., choosing random propagation targets).

We can conclude from this simple cluster analysis of the various features that some of them (F_{mw_md5}, $F_{pe\_kernel32}$) may reveal some hints on the polymorphic or non-polymorphic nature of the malware samples, whereas some others (like F_{mw_size}, F_{pe_linker}, $F_{pe\_sections}$ $F_{bh\_optypes}$, $F_{bh\_summary}$) do not seem to be of straightforward help in this specific case. For this reason, it might be useful to try to combine all features by relying on the *multi-criteria fusion* component of TRIAGE, with the purpose of identifying multi-dimensional clusters reflecting different code variants.

4.3.2 MCDA Aggregation

Even though a cluster analysis of individual features may give some interesting viewpoints and reveal informative patterns, we still lack the global picture, and it might be even more interesting to try to "connect all dots".

In this Section, we report on the experiment performed with the aggregation of the eight malware features (static and dynamic) of all `Allaple` samples. As described before, the objective of such a MCDA analysis is to help separate the various code variants (polymorphic, non-polymorphic, and possible other families like bots or trojans) from the set of 10,162 malware samples. We have used two different aggregation functions and have compared their performance.

First, we have defined an OWA aggregation function (defined in Section 2.4.4) that reflects our expectations regarding the minimum number of features required to attribute samples to a given malware variant with high confidence:

$$\mathbf{w} = [0, 0, 0.10, 0.20, 0.40, 0.20, 0.10, 0]$$

By defining such an OWA weighting vector, we completely remove the influence of the two highest scores, no matter *which* features they are related to. We start then to give some importance to the third highest score, with the highest weights given to the fourth and fifth highest positions. This means that at least four strong correlations will be needed in order to have a global score above 0.5. Recall that we do not need to specify in advance which combination(s) of four malware features are required to link malware samples.

The second method we have used to model the aggregation of all features is a fuzzy integral based on Choquet. Defining a complete fuzzy measure that models all possible coalitions of features requires 2^n values, which in this case with $n = 8$ is already prohibitive. As described in Section 2.4.5, the best compromise between complexity and richness of the aggregation model consists to define a 2-additive fuzzy measure, where the analyst only has to define the overall importance factors for every feature separately (i.e., the Shapley values, as defined in 2.7), and, if needed, define also some interactions among pairs of features (such as redundancies or synergies) to enrich the model. As a result, the complexity of the sub-model represented by such a 2-additive measure is reduced to the definition of at most $n(n+1)/2$ weighting factors.

To build our 2-additive fuzzy measure, we have first defined the following *importance factors*:

F_{mw_md5}	F_{mw_size}	$F_{pe\_sections}$	F_{pe_linker}	$F_{pe\_kernel32}$	$F_{bh\_optypes}$	$F_{bh\_fullops}$	$F_{bh\_summary}$
0.15	0.14	0.06	0.12	0.15	0.06	0.20	0.12

To enrich the model, we have then added certain *interactions* among following pairs of features:

$(F_{mw\_md5}, F_{pe\_kernel32})$	$(F_{mw\_md5}, F_{bh\_fullops})$	$(F_{bh\_fullops}, F_{bh\_summary})$	$(F_{bh\_optypes}, F_{bh\_summary})$
0.06	0.06	0.048	-0.06

These interaction values were automatically computed by TRIAGE so as to enforce the additivity and monotonicity conditions (as given by expression 2.22); the analyst just needs to set a relative amount of synergy or redundancy between pairs of features. In this case, we have set 20% of synergy for the first three pairs here above, and 50% of redundancy for the pair $(F_{bh\_optypes}, F_{bh\_summary})$. Then, to compute the Choquet integral starting from those importance factors and interaction values, we have used the formula given by expression 2.21.

The last important parameter to set is the decision threshold ϵ, which is used to remove unwanted edges in the aggregated graph, and then to identify the multidimensional clusters (MDCs) via the connected components algorithm. A sensitivity analysis is required for ϵ to determine the best ranges of values according to the number of MDCs, total percentage of clustered samples and the distribution of cluster sizes obtained with various values of the decision threshold.

The result of the sensitivity analysis for the 2-additive Choquet aggregation is illustrated in Fig. 4.1, where we can see that values of ϵ starting from 0.44 up to 0.6 seem to be appropriate, according to the number of MDCs and total amount of samples that are clustered. We have chosen a decision threshold of 0.46 in order to keep as many malware samples as possible, without grouping too many of them within the same MDC.

In Fig. 4.2, we can observe that OWA aggregation gives apparently a stable, but somehow weird behavior. There are indeed only two large plateaus where all indicators (nr of MDCs, etc) are very stable. For OWA, threshold values taken for the first plateau (i.e., between 0.40 and 0.70) seem to be an appropriate range for ϵ. We have thus selected 0.50 for our analysis.

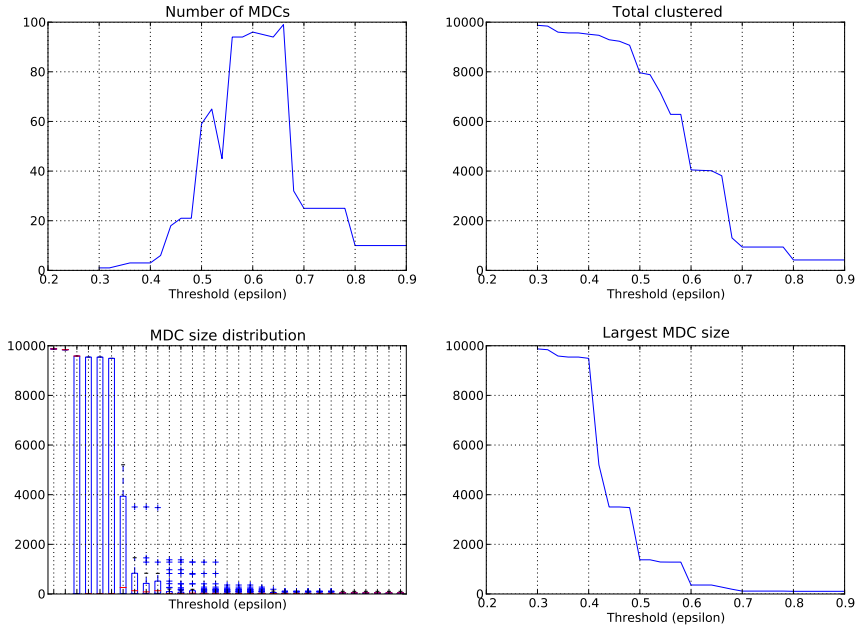


Figure 4.1: Sensitivity analysis of the decision threshold ϵ for the Choquet aggregation process.

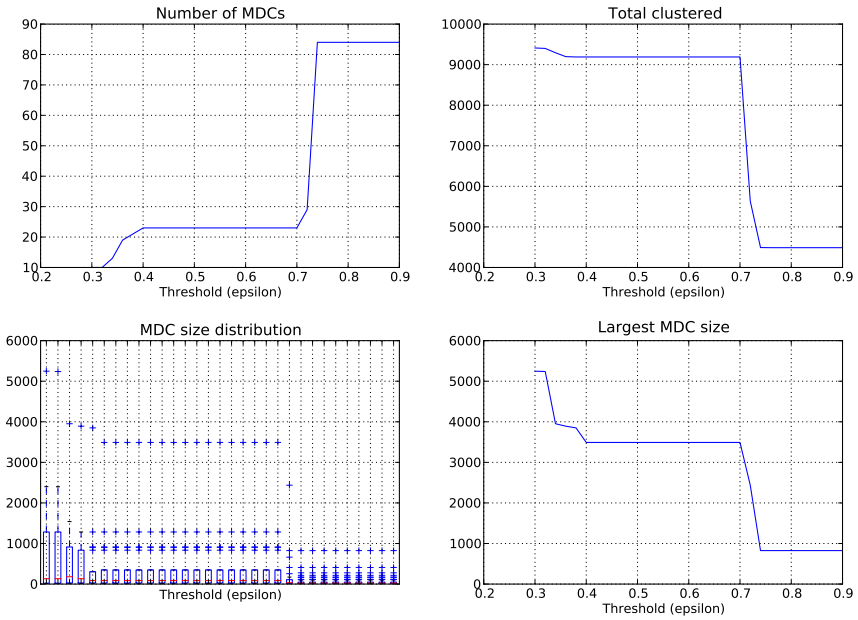


Figure 4.2: Sensitivity analysis of the decision threshold ϵ for the OWA aggregation process.

In Table 4.3, we compare the global performance of OWA and Choquet aggregations. Overall, both methods give fairly similar results in terms of MDCs, average compactness (C_p) and total number of samples clustered in MDCs. Note, however, that we haven't included too many redundant features in this analysis; otherwise, Choquet would definitively offer a more effective and flexible aggregation method than OWA. For the rest of the analysis, we will now focus on the 21 MDCs (which we will also call malware *variants*) found with the Choquet aggregation.

We turn now to the evaluation of the consistency of individual MDCs. In Fig. 4.3, we have represented the global graph compactness of the MDCs (and each color in the bar chart refers to the average compactness index C_p of a single feature). We observe that all MDCs have on average *at least 4 features* with a high C_p, which seems to be consistent with the constraints we have previously modelled regarding aggregation parameters. Most MDCs have indeed a global compactness value between 4 and 5 (the max. value is 8). In the same chart, based on the C_p of `mw_md5`, it is very easy to see which MDCs are quite likely made of polymorphic malware variants, and which ones are likely to contain non-polymorphic samples (i.e., all MDCs with a high C_p for `mw_md5`).

For example, all malware samples of MDCs 1481, 1487, 1488, 1489 and 1493-96 seem to have the same MD5 (or a very limited number of MD5's). Not surprisingly, for all these MDCs a high correlation w.r.t. `mw_md5` involves also a high correlation for `mw_size`, `pe_sections`, `pe_linker` and `pe_kernel32` (i.e., non-polymorphic samples).

Regarding the polymorphic MDCs, it is worth noting that the involved samples have still apparently very similar global behaviors in a sandbox, since most of these MDCs have high C_p values for the features `bh_summary` and `bh_optypes`. Finally, we note that feature `bh_fullops` seems to be the less correlating feature.

Table 4.3: Comparison of OWA and Choquet aggregation methods for the Allaple data set.

Characteristic	OWA	Choquet
Threshold ϵ	0.50	0.46
Nr of MDCs	23	21
Total clustered	9,189 (90%)	9,231 (91%)
Largest MDC	3,491	3,505
Average C_p	0.59	0.58

4.4 Analysis of the variants

By analyzing in-depth the 21 variants (or MDC's) given by TRIAGE, we could easily identify four different families of malware variants:

1. `[Allaple.b / Rahack.W]`: this malware family contains only purely polymorphic samples, grouping 5 MDC's accounting totally for 4,207 samples (41% of the dataset). Quite obviously, all `mw_md5` and `pe_kernel32` are unique values. The `pe_linker` used to compile those samples is mostly version `92` (rarely `140`), and there are many different `mw_size` values (with still some popular binary sizes shared by groups of samples). Regarding the behavioral features, this malware family is characterized by a unique high-level behavior for more than 98% of the samples, and this variant behavior is apparently the only one involving the operation `win_dir_copy`.

   To illustrate this malware variant, Table 4.4 gives an overview of the main characteristics of the samples of MDC 1479, along with the set of signatures given by two well-known anti-virus products (AV brands have been anonymized). As we can see,

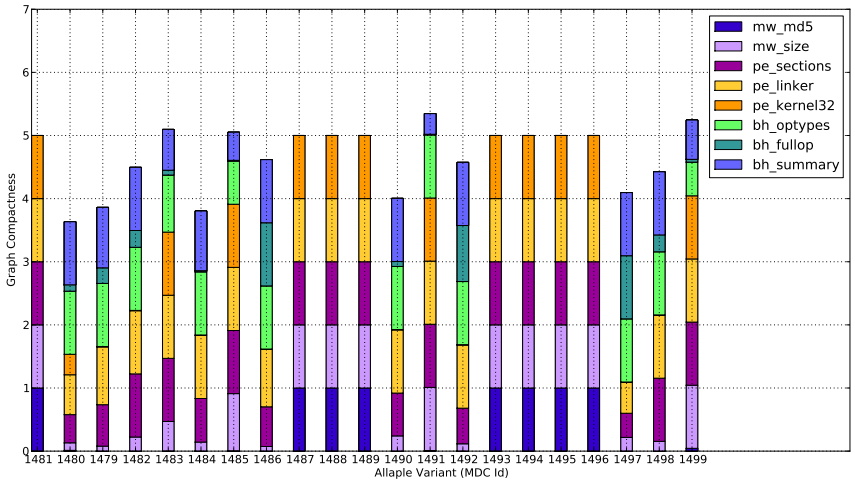


Figure 4.3: Evaluation of the MDCs found by aggregating all malware features using a Choquet integral. Each color in the bar chart refers to the average compactness index C_p of each individual feature.

AV_2 gave the same signature (`Rahack.W`) for almost 100% of the samples while the other one failed to detect approximately one third of them. Finally, Fig. 4.5 illustrates the polymorphic patterns of this malware variant by representing on a graph (with radial layout) all relationships between features `mw_md5`, `pe_kernel32` and `mw_size`.

2. `[Allaple.d/e / Rahack.H]`: this malware family contains a mix of polymorphic and non-polymorphic samples, grouping 11 different MDC's accounting for a total of 2,869 samples (28% of the dataset). In this case, there are not as many `mw_md5` values than the number of code injections, and the `pe_kernel32` values have the same pattern for about 50% of the samples. The `pe_linker` is not only version `92`, but sometimes also `96` (in 25% of the cases). All samples have again the very same high-level behavior (but still a different one from the behavior of the first variant called `Allaple.b` here above).

   To illustrate this variant, Table 4.5 gives an overview of the main characteristics of the samples of MDC 1480, along with the set of signatures given by anti-virus products. As we can see, `AV_2` gave the same signature (`Rahack.H`) to 75% of

Table 4.4: Characteristics of Allaple variant 1479 `[Allaple.b, RaHack.W]` (3,505 samples). (Note: *** means a polymorphic feature)

Feature	C_p	Distinct	Patterns
ex_date	-	527	2007-09-15 → 2009-08-11
mw_md5	0.0	3,505	***
mw_size	0.079	175	***
pe_sections	0.656	12	.text\x00\x00\x00|rdata\x00\x00\x00|.data\x00\x00\x00\x00: 80%, others: 20%
pe_linker	0.915	2	92: 96%, 140: 4%
pe_kernel32	0.0	3,505	***
bh_optypes	1.0	1	network|service|process|section|random|sync|thread|registry|file|time: 100%
bh_fullops	0.247	123	[REMOVED]
bh_summary	0.958	2	addressscan|file_modification_destruction|all_reg_activities|win_dir_copy: 98%
			addressscan|file_modification_destruction|all_reg_activities|win_dir_copy|auto_start: 2%
AV_1	-	6	Net-Worm.Win32.Allaple.e: 11%, Net-Worm.Win32.Allaple.b: 78%, others: 11%
AV_2	-	2	W32.Rahack.W: 100.0%, W95.Drill.18624: 0.0%

the samples, with 25% of the samples attributed to various other signatures. For `AV_1`, 66% of the samples had the signature `Allaple.e`, a few of them had also the signature `Allaple.d`, and the rest of the samples received quite diverse signatures, with many of them referring to `Rbot` or `Virut` signatures. An in-depth analysis of all those samples revealed that it was indeed sometimes difficult to distinguish samples of this malware family (`Allaple.d/e`) from samples belonging to a bot or backdoor family (i.e., the fourth variant described here under).

Finally, Fig. 4.7 gives a graphical representation of all samples attributed to variant 1480 according to features `mw_md5`, `pe_kernel32` and `mw_size`. On this graph, we can clearly distinguish the non-polymorphic samples (in the middle of the graph) from polymorphic ones characterized by many different values of `mw_md5`, `pe_kernel32` (large circles on the sides), but still grouped apparently through some popular `mw_size`.

3. `[Allaple.a / Rahack.H]`: this malware family is another polymorphic variant (w.r.t mw_md5 and pe_kernel32), which is almost identical to `Allaple.e` from a behavioral viewpoint. However, there are some subtile differences regarding the static features `pe_linker`, and `pe_sections`. Table 4.6 gives an overview of the characteristics of this Allaple variant: observe that the linker version is `96` for all samples, and note the different bytes `\x1a` present in the section data. This variant contains only 76 samples.

4. `[Backdoor.Trojan | Backdoor.Win32.Rbot.bni]`: this malware family is very different from previous ones, since all samples seem to refer here either to a back-

Table 4.5: Characteristics of Allaple variant 1480 `[Allaple.d/e, RaHack.H]` (1,455 samples). (Note: *** means a polymorphic feature)

Feature	C_p	Distinct	Patterns
ex_date	-	482	2007-09-26 → 2009-12-30
mw_md5	0.012	836	***
mw_size	0.119	55	59904: 12%, 67584: 1%, 90112: 1%, 89600: 2%, 85504: 5%, 61440: 2%, 78336: 21% 60928: 1%, 88064: 3%, 57344: 17%, others: 11%, 57856: 5%, 50176: 15%, 86016: 2%
pe_sections	0.446	4	.text\x00\x00\x00|rdata\x00\x00\x00|.data\x00\x00\x00: 62%, others: 38%
pe_linker	0.632	2	92: 76%, 96: 24%
pe_kernel32	0.324	630	(GetProcAddress, LoadLibraryA): 57%, others: 43%
bh_optypes	1.0	1	network|thread|process|section|random|sync|registry|file|time: 100%
bh_fullops	0.101	189	[REMOVED]
bh_summary	0.999	2	addressscan|file_modification_destruction|all_reg_activities: 99.9%, addressscan|file_modification_destruction|all_reg_activities|auto_start: 0.1%
AV_1	-	11	Net-Worm.Win32.Allaple.e: 66%, Virus.Win32.Virut.n: 10% Backdoor.Win32.Rbot.bni: 8%, others: 16%
AV_2	-	4	W32.Rahack.H: 75%, others: 25%

Table 4.6: Characteristics of Allaple variant 1490 `[Allaple.a, RaHack.H]` (76 samples). (Note: *** means a polymorphic feature)

Feature	C_p	Distinct	Patterns
ex_date	-	59	2007-10-04 → 2009-11-29
mw_md5	0.0	76	***
mw_size	0.241	16	82432: 8%, 57856: 47%, 90112: 8%, 61440: 7%, others: 30%
pe_sections	0.679	2	.text\x00\x00\x00|rdata\x00\x1a|.data\x00\x00\x00: 80%, .text\x00\x00\x00|rdata\x00\x1a|.data\x00\x00\x00|.rrdata\x00: 20%
pe_linker	0.632	2	96: 100%
pe_kernel32	0.0	76	***
bh_optypes	1.0	1	network|thread|process|section|random|sync|registry|file|time: 100%
bh_fullops	0.078	38	[REMOVED]
bh_summary	1.0	1	addressscan|file_modification_destruction|all_reg_activities: 100%
AV_1	-	4	Net-Worm.Win32.Allaple.a: 74%, others: 26%
AV_2	-	1	W32.Rahack.H: 100%

door or a bot family according to AV labels. We could identify four similar MDC's that apparently contain only these backdoor or bot-related samples, instead of pure `Allaple` worms. Most of those samples are clearly non-polymorphic ones and share the same `pe_kernel32` pattern (`GetProcAddress`, `LoadLibraryA`). They also had the very same high-level behavior regarding `bh_summary` and `bh_optypes`, which are also different from the behavioral patterns of all previous variants since they

usually involve high-level operations such as `ircbot` or `internet_settings` (which are not present in the behavioral profiles of `Allaple` variants).

Table 4.7 shows the main characteristics of MDC 1481, a large MDC containing 1,284 samples having the very same `mw_md5` and `mw_size` values, and attributed to this family of backdoor/trojan samples. Interestingly, for this specific malware, we couldn't get any behavioral information from the Anubis sandbox. To illustrate the different kind of patterns of this non-polymorphic malware family, we have represented in Fig. 4.6 the MDC 1483, also attributed to this `Backdoor.Trojan` family and containing 519 malware samples.

Table 4.7: Characteristics of Allaple variant 1481 `[Backdoor.Trojan]` (1,284 samples).

Feature	C_p	Distinct	Patterns
ex_date	-	457	2007-09-13 → 2009-12-30
mw_md5	1.0	1	3875b6257d4d21d51ec13247ee4c1cdb: 100%
mw_size	1.0	1	57344: 100%
pe_sections	1.0	1	.text\x00\x00\x00|rdata\x00\x00\x00|.data\x00\x00\x00: 100%
pe_linker	1.0	1	92: 100%
pe_kernel32	1.0	1	(GetProcAddress, LoadLibraryA): 100%
bh_optypes	0.0		-
bh_fullops	0.0		-
bh_summary	0.0		-
AV_1	-		Backdoor.Win32.Rbot.bni: 100%
AV_2	-		Backdoor.Trojan: 100%

Final note on contextual features

In Section 4.2, we have presented all available features at our disposal in this SGNET dataset. Besides static and dynamic features, we have also introduced certain contextual features, such as source or destination IP addresses, ASN, etc. (see Table 4.1). However, we have also explained that we were not so confident on the discriminant value of these contextual features for this specific analysis.

To verify this assumption, we have analyzed the 21 variants (MDC's) obtained from the aggregation of the eight code-related features according to those contextual features. After analysis, we were unable to find any meaningful patterns regarding the origins, the destinations or the timing of all these `Allaple`-related code injections observed by

SGNET sensors. To illustrate this point, Fig. 4.4 represents the evolution of source IP addresses (grouped by /8 subnets) as a function of time (by day), for different malware variants identified by TRIAGE. From this figure, it is clear that code injections related to the `Allaple` propagation are apparently always coming from the very same group of networks, with no significant differences between the different variants. The same conclusions hold for source ASN and targeted sensors. In other words, for this specific malware analysis, contextual features were not helpful to discriminate code variants or malware families. This is justified in most cases by the random propagation pattern associated to the `Allaple` worm, that we have seen to constitute the vast majority of this specific dataset. Still, we have seen in the analysis that a minority of other types of malware shares the same propagation vector, but exposes very different behavior (e.g. connecting to C&C channels). Contextual information has been instrumental in [40] to understand the long-term behavior of coordinated hosts, but due to the very small number of instances in this specific dataset, this information did not turn out to be useful here.

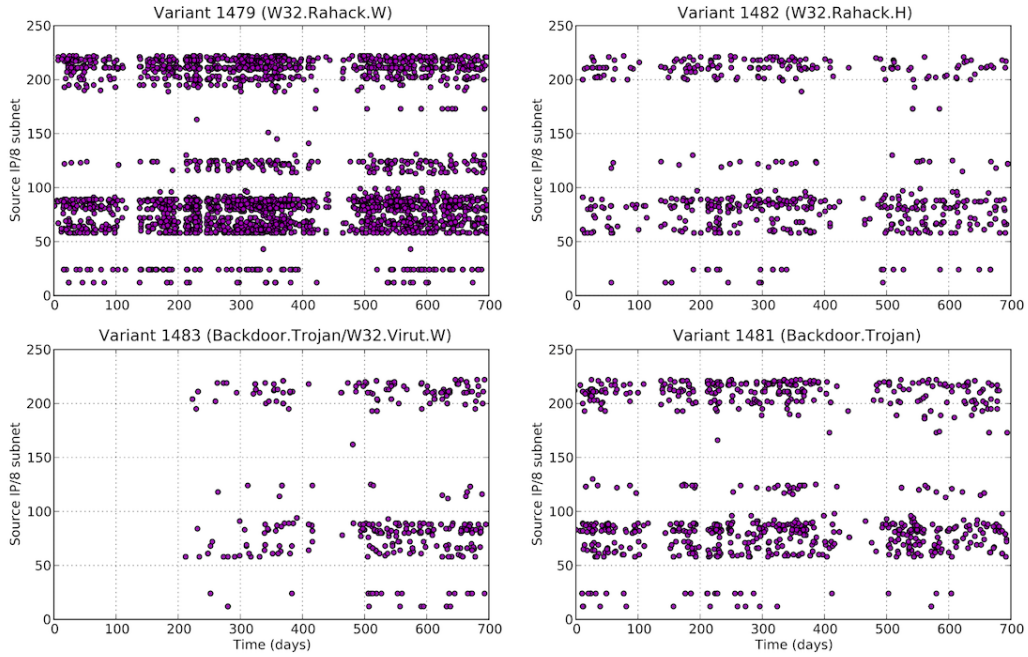


Figure 4.4: Evolution of Source IP addresses (grouped by /8 subnets) for different Allaple variants.

4.5 Summary

In this Chapter, we have demonstrated the use of TRIAGE to perform a multi-criteria analysis of a malware data set comprising 10,162 samples related to the `Allaple` propagation scheme and collected by SGNET for a period of 2 years. The purpose of such an analysis was to understand the *root causes* behind the propagation of all those samples by identifying the various malware variants found by TRIAGE.

By aggregating eight different features related to the structure of the code (i.e., static features) and the behavior of the malware samples (i.e., dynamic features) when executed in ANUBIS, we were able to infer that this large malware data set could in fact be summarized by only three or four different malware families. In particular, TRIAGE was able to separate *polymorphic* versus *non-polymorphic* samples, as well as pure `Allaple` worm samples from the `Backdoor/Trojan` samples that share the very same propagation vector, but exhibit very different behavior from `Allaple`.

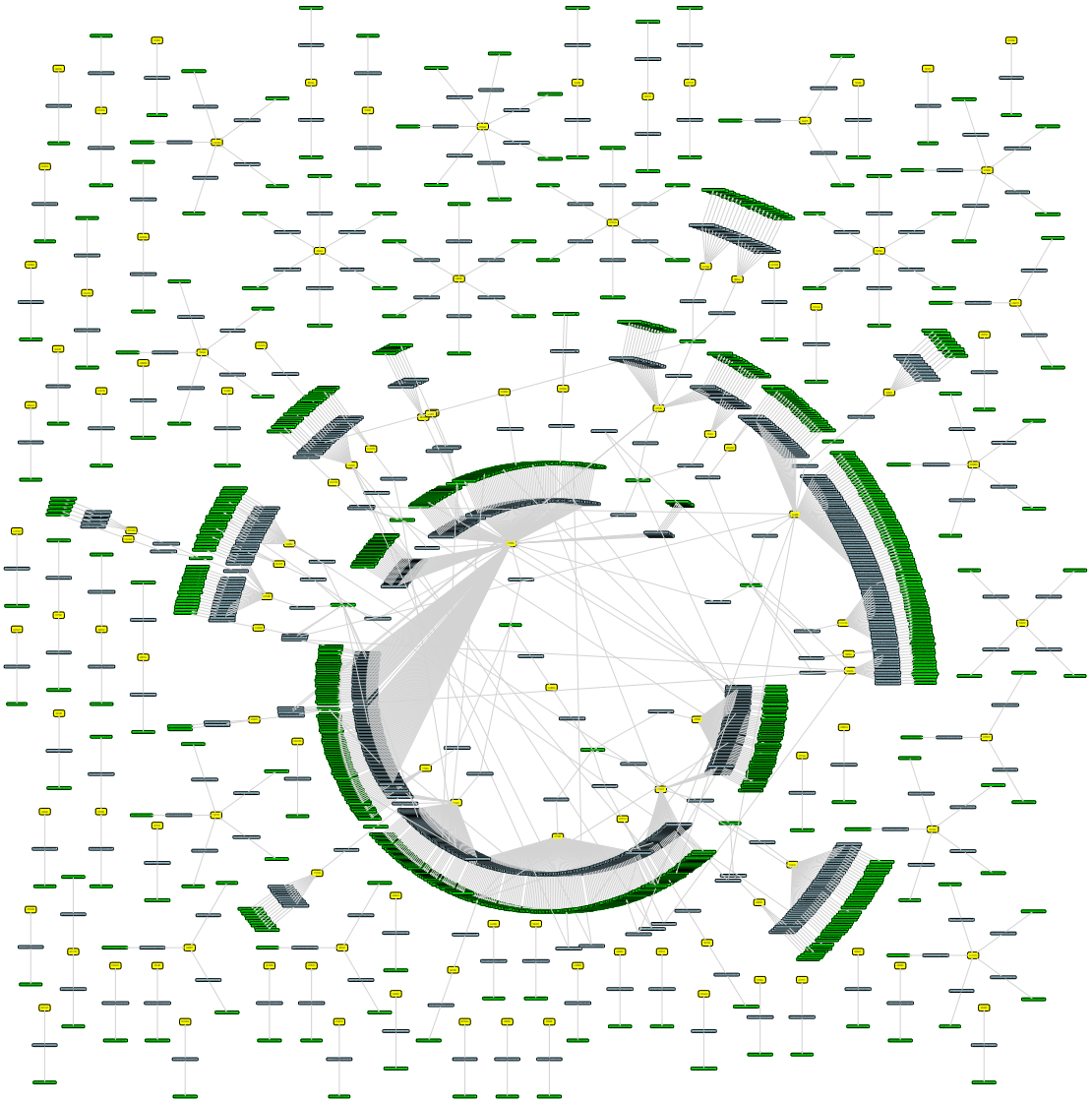


Figure 4.5: Allaple variant 1479 [`Allaple.b`|`Rahack.W`]. Purely polymorphic variant w.r.t. `mw_md5` (nodes in blue) and `pe_kernel32` (nodes in green). Interestingly, there are still some very popular binary sizes (i.e., `mw_size`, the nodes in yellow) shared by large groups of polymorphic samples.

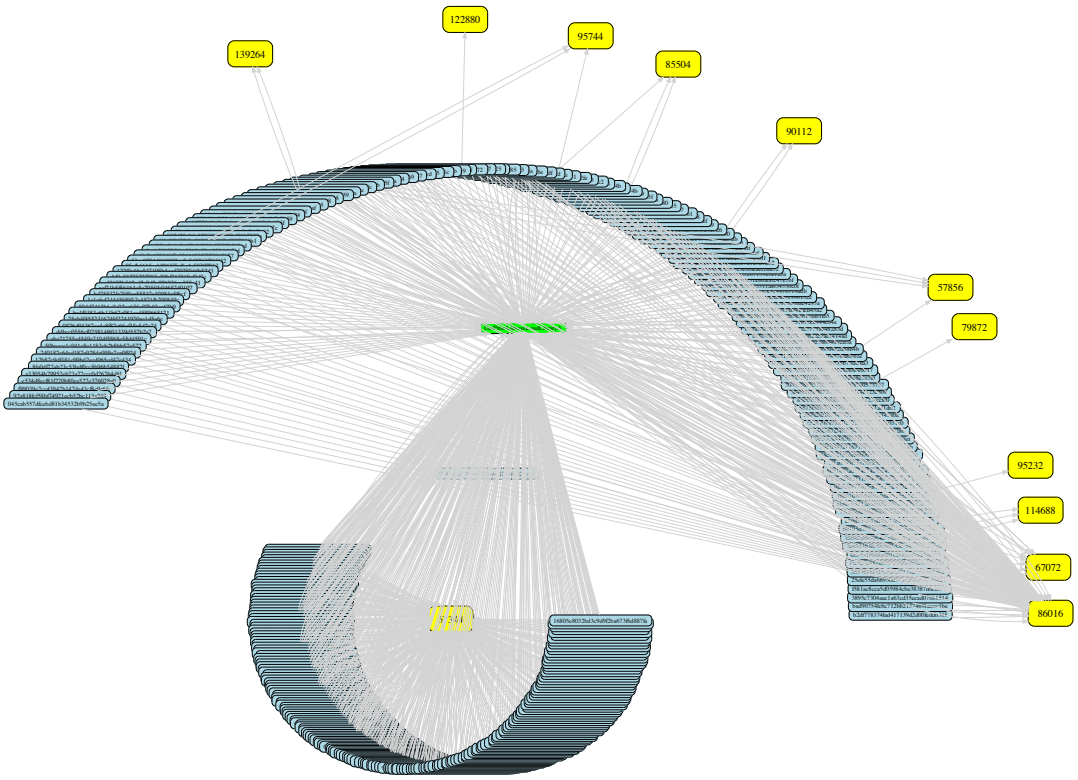


Figure 4.6: Allaple variant 1483 [`Backdoor.Trojan`|`Virus.Win32.Virut.av`|`Backdoor.Win32.Rbot.adqd`]. A non-polymorphic Backdoor variant, characterized by a constant pe_kernel32 value (`GetProcAddress`,`LoadLibraryA` – green node), and a few different binary sizes (nodes in yellow) with a popular size of 57,344 bytes. Interestingly, this Allaple variant comprises 519 code injections, but only 328 distinct MD5 hashes (nodes in blue). Note the very different patterns of this graph compared to pure polymorphic variants (Fig. 4.5 and 4.7).

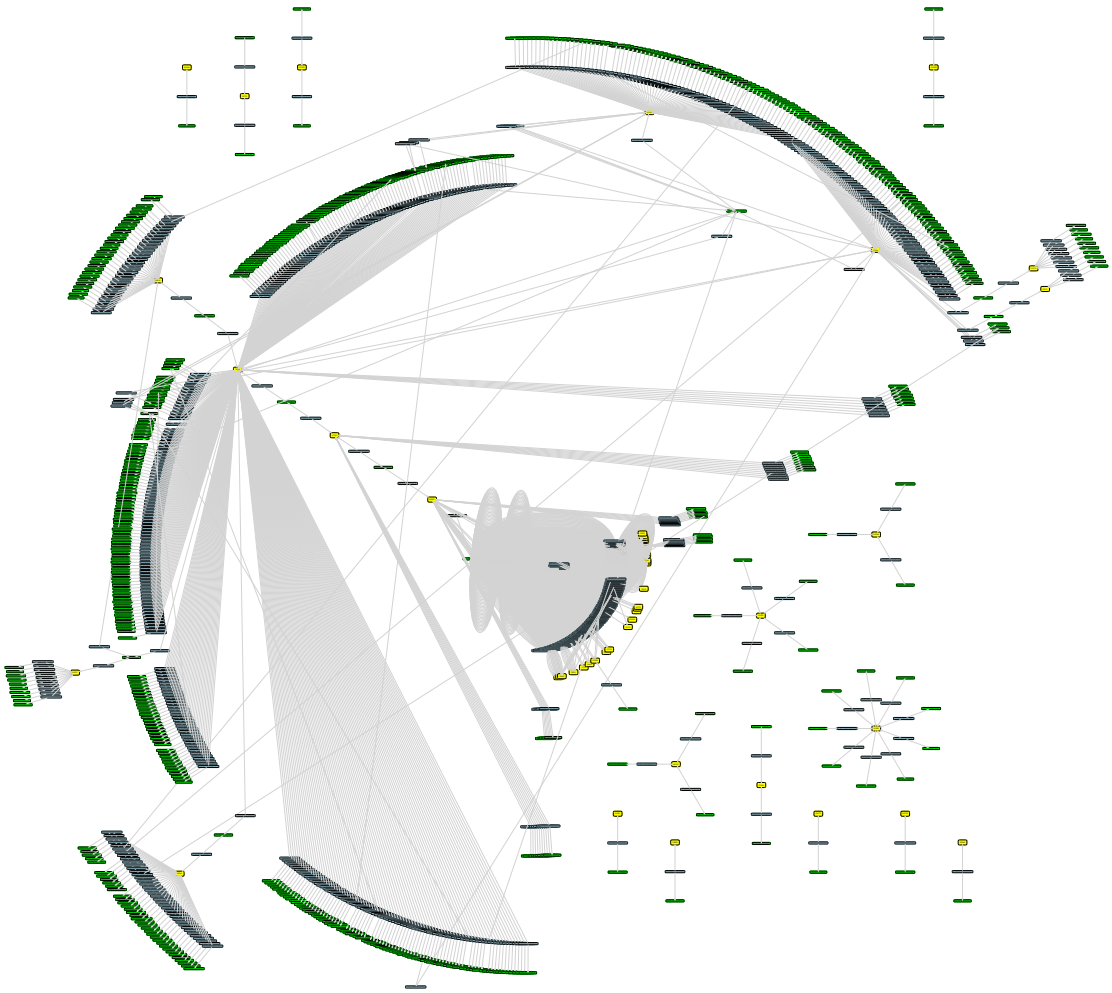


Figure 4.7: Allaple variant 1480 [`Allaple.d/e`|`Rahack.H`]. A mix of two other variants, which contains both non-polymorphic samples (in the center of the graph) and polymorphic ones (large circles on the sides). Legend: blue nodes: `mw_md5`, green nodes: `pe_kernel32`, yellow nodes: `mw_size`. Interestingly, there are still some very popular binary sizes (nodes in yellow) shared by large groups of polymorphic samples. Between 25 and 35% of these samples have AV signatures linked to `Backdoor.Trojan`, `Virut.n` or `Win32.Rbot`, which shows the difficulty to distinguish backdoors/bots variants from `Allaple.d/e` samples (the non-polymorphic samples of this group are most probably linked to backdoors). Note also that the behavioral profiles of all these samples were quite different from variant 1479 (`Allaple.b`).

5 Analysis of Spam Botnets

5.1 Introduction

Many illegal and profitable malicious activities in the Internet are enabled by gaining control over *botnets* [5, 18, 60, 4] comprising thousands or sometimes even millions of infected machines, of which many belong to innocent home users that are even not aware of it. Today it is well-known that the worldwide spam problem is largely due to a few large groups of compromised computers, called *spam botnets*, which are under the control of cyber criminal organizations. According to the 2010 Annual Security Report of MessageLabs [50], the annual average spam rate was 89.1% of all intercepted messages (an increase of 1.4% compared with 2009), with approximately 88.2% of this spam volume originating only from spam botnets. As analyzed by SecureWorks [68], in 2008 the top botnets were collectively able of sending over 100 billion spams per day. In 2010, this spam sending capability was estimated to 71 billion spams per day [50], a slight decrease compared to 2009 [49]. However, the spam volume sent by those botnets is still significant today and continues to pose a major threat to our digital economy.

In this Chapter, we show how TRIAGE was used to analyze the *spam campaigns* that are performed through spam botnets. In particular, we are interested in knowing more about the behaviors of those spamming bots, e.g.: what are the *modus operandi* of spammers who control those botnets and how do they run their campaigns? Can we highlight some inter-relationships among *different* spam botnets, like shared origins or spam campaigns run in parallel by several botnets? Are there "specialized" botnets that tend to be used to launch specific kinds of spam campaign? And finally, can we observe other hidden specificities or patterns that might not appear directly from the raw spam messages sent by those botnets?

Quite obviously, answers to such questions are very important, as they may not only improve our understanding of spam botnets and their "ecosystem", but they may also emphasize the strategic or economic decisions made by organizations controlling them. Such intelligence can help us improve our defence mechanisms, and could also ultimately help us to design countermeasures for defeating the operations and activities of spam botnets.

In the following Sections, we start by giving a brief overview of the spam botnets

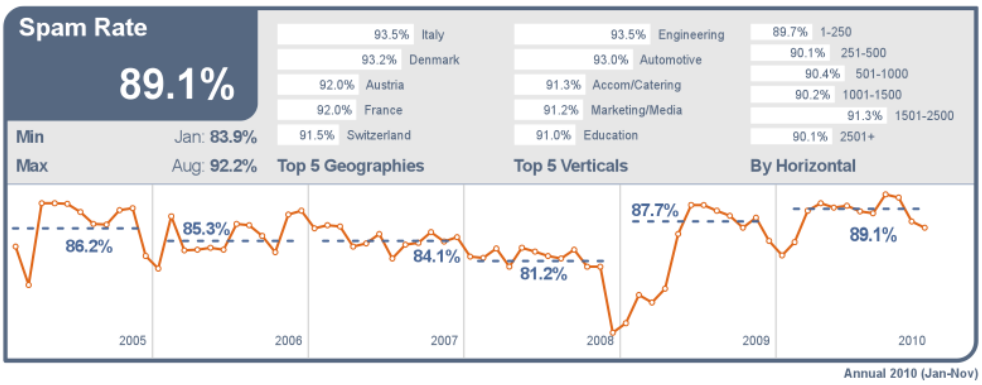


Figure 5.1: Global spam rate observed by Message Labs [50] over the latest years (until end of 2010).

ecosystem as it is known today. Then, we describe the Message Labs data set that was at our disposal for this TRIAGE analysis. In Sections 5.2 and 5.3, we provide details on how we have applied our multi-criteria analysis tool to extract intelligence from this spam dataset and identify various spam campaigns run by different botnets. Finally, to illustrate the results, we further analyze some of the spam campaigns in Section 5.4. We conclude and summarize our findings in Section 5.5.

5.1.1 Spam Botnets Ecosystem

Spam is a very important business, and thus in the latest years, cyber criminals have found new and innovative ways to attack computers and businesses via sophisticated forms of malware that have continued to infect millions of machines, forming today the well-known (spam) botnets. A *botnet* is a collection of zombie machines controlled by cyber criminals using a particular strain of malware for each botnet. Many botnets are used to send spam, some others (like the Zeus Trojan) are developed and maintained to conduct financial fraud – these are also referred to as *banking trojans*, see the BANOMAD early warning system developed within WOMBAT and described in deliverable D23 (D5.3) for more information about this kind of trojans.

Since a few years, we observe that botnets generally account for between 80-90% of all spam sent globally, as pointed out in the annual intelligence report published by MessageLabs [50]. Fig. 5.1 shows that the global spam rate observed by this company at the end of 2010 was still 89.1%, despite an important drop in 2008 due to the disconnection of *McColo*, a rogue ISP based in California who was hosting many botnets C&C servers.

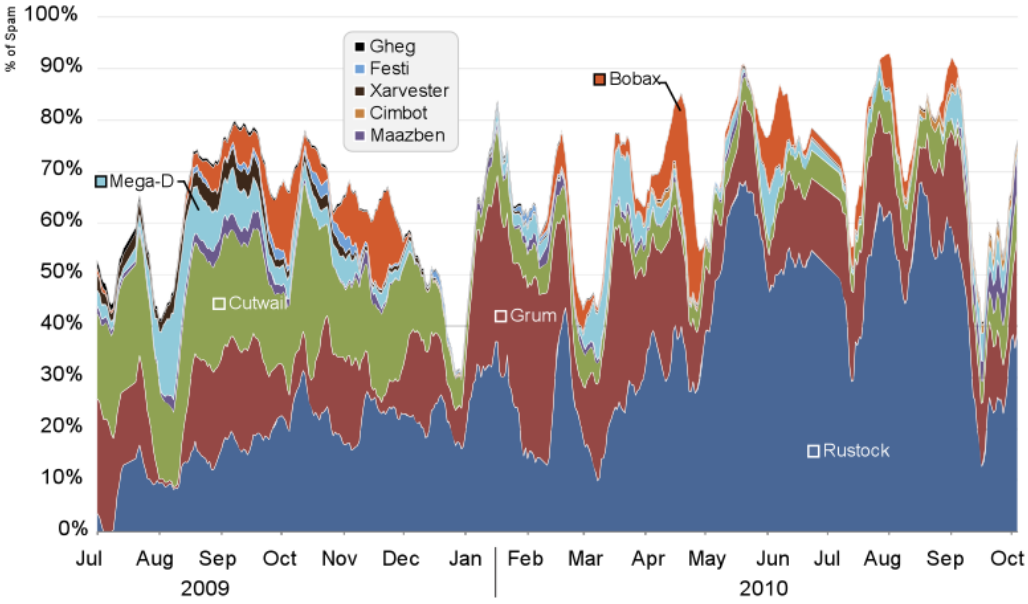


Figure 5.2: Volume of spam sent by the largest spam botnets, as observed by Message Labs [50].

Since the McColo takedown, spam botnets have unfortunately recovered very quickly: in Fig. 5.2, we can see that in 2010 the proportion of spam sent from botnets was still accounting for approximately 88% of the global spam volume. On the same figure, we can also observe which botnets are mainly responsible for this huge volume of spam. `Rustock` seems to be one of the most active botnets since a couple of years, with a spam volume of 47% attributed only to this botnet at the end of 2010. Fig. 5.3 provides some global characteristics of the most important botnets, together with a breakdown of spam by categories for 2009 compared to 2010. As one can see, the top spam categories in 2010 continued to be related to pharmaceutical products, casino/gambling and selling counterfeit products (such as replica watches, etc). Interestingly, two new categories have also appeared in 2010 in the top 5, namely unsolicited newsletters and scam related to fake *job offers* or *mules*.

botnet	% of spam	spam/day	spam/ bot/ min	Est. botnet size	Country of Infection
Rustock	47.5%	44.1 billion	145	1100k to 1700k	USA (17%), Brazil (7%), India (7%)
Grum	8.5%	7.9 billion	91	310k to 470k	Russia (12%), India (8%), Vietnam (8%)
Cutwail	6.3%	5.9 billion	40	560k to 840k	India (17%), Russia (16%), Ukraine (8%)
Maazben	5.2%	4.8 billion	37	510k to 770k	Russia (11%), India (10%), Brazil (7%)
Mega-D	2.3%	2.1 billion	105	80k to 120k	Russia (15%), Ukraine (14%), Brazil (7%)
Cimbot	2.1%	1.9 billion	185	32k to 48k	Italy (27%), Spain (25%), France (14%)
Bobax	1.2%	1.1 billion	18	250k to 370k	India (32%), Russia (25%), Ukraine (9%)
Xarvester	0.5%	501 million	109	17k to 25k	Italy (15%), UK (10%), Poland (8%)
Festi	0.1%	96 million	127	8k to 12k	Vietnam (24%), Indonesia (21%), India
Gheg	0.1%	49.8 million	20	8k to 12k	Spain (12%), Indonesia (10%), India (10%)
Other, smaller Botnets	0.5%	26.8 million	22	220k to 340k	
UnNamed Botnets	2.9%	2.6 billion	21	490k to 740k	
Total BotnetSpam	**77.0%**	**71.1 billion**	**77**	**3500k to 5400k**	**India (9%), Rusia (9%), USA (7%)**
Non-botnet spam	**23.0%**	**21.8 billion**			
Grand Total		**92.9 billion**			

Category: End of 2009	% spam
Pharmaceutical	64.0%
Watches	14.3%
Weight Loss	4.9%
Casino/Gambling	4.9%
Unsolicited Newsletters	2.5%
Malware	2.5%
Degrees	2.0%
Jobs/Mules	1.0%
Phishing	1.0%
Scams/Fraud	0.5%
Sexual/Dating	0.5%
Unknown/Other	0.5%
Degrees/Diplomas	0.5%
Missing Persons	0.5%
Software	0.5%

Category: End of 2010	% spam
Pharmaceutical	64.2%
Unsolicited Newsletters	9.3%
Casino/Gambling	7.0%
Watches	6.5%
Jobs/Mules	3.3%
Sexual/Dating	3.3%
Software	1.4%
Degrees/Diplomas	0.9%
Phishing	0.9%
Missing Persons	0.5%
Malware	0.5%
Weight Loss	0.5%
Unknown/Other	0.5%
Mobile Phones	0.5%
Scams/Fraud	0.5%

Figure 5.3: *(Left)* Some global characteristics of the largest spam botnets (October 2010). (*Right*) Breakdown of spam by categories (*source*: Message Labs Intelligence [50]).

5.1.2 MessageLabs Dataset

As interesting as the figures and statistics here above may be, we are still lacking more strategic information on the activities of those spam botnets, in order to try answering the aforementioned questions on spam campaigns operations and botnet activities.

For this reason, we have considered a TRIAGE analysis of a very large spam data set provided by Message Labs. As part of their continued business, the MLI team (Message Labs Intelligence) sets up and maintains spamtraps all around the world. All email traffic sent to these spamtrap domains is analyzed by honeypots that extract different features from the emails (including the message content, sender's IP address, name of the sending bot if available from CBL [16] rules, embedded URIs, "From" domain being used, etc). The spamtrap traffic is used by Message Labs for two different purposes: (i) it serves as training data for a learning module that attempts to tune and update spam detection rules, so as to reach very high detection rates for the filtering modules used to protect MessageLabs client's email traffic, and (ii) the spamtrap traffic is sampled on a daily basis, with about 10,000 random samples stored every day in a separate database, which serves as a baseline for the statistical analysis of spam and botnet traffic.

Bot-related features		
Bot signature	Name of the spam bot, as obtained from CBL signatures [16].	76
From domain	The email domain used as "From" field in the spam email header.	84,246
OS details	Name and type of OS of the sending machine, obtained from P0f (passive OS fingerprinting).	92
Origin-related features		
Country (code)	Geographical origins of the spam sender (city, lat./long., region, continent are also available).	220
Host name	Name of the machine having sent the spam message (obtained by reverse DNS query).	11,768
Source IP	The IP address of the spamming machine.	648,638
Source Class A	The source IP aggregated by its /8, 16, /24 subnet	169
Class B		16,314
Class C		331,208
Campaign-related features		
Subject line	The "subject" field of the spam message.	596,932
Day	The day on which the spam message has been received.	92
URI domains	The set of domains of the URIs embedded in the body of the message (if any).	107,469
Charset	The character set used by the sender to encode the message (e.g., us-ascii).	48
Language	The language used for the spam content, as detected by various language detection modules.	17
Message size	The size of the message (in bytes).	27,777
Attachment	The set of names of any files attached to the spam message.	18,886

Table 5.1: SPAM features available in the MessageLabs spam dataset (right column is the cardinality)

The spam data provided by MLI was collected by worldwide distributed spamtraps in a 3-months period spanning from October, 1st 2010 until January 1st, 2011. From this traffic, about 1 million spam messages were randomly and uniformly chosen on the whole period. All spam features available in the dataset are described in Table 5.1.

5.1.3 Preprocessing messages into spam events

The 3-months dataset provided by MLI contains 923,293 spam messages and about 20 different features. As pointed out in Chapter 2, the current TRIAGE approach relies on a graph-based representation, which requires the computation of pairwise distances (or similarities) between all objects. Although this operation could be easily parallelized in the future, the current implementation does not allow to process graphs or matrices of size 1 million-by-1 million. However, most security data sets are made of events that

can easily be grouped according to certain common features, such as the "type" of event (e.g., type of exploit or a given TCP/UDP port for network attacks, a given type or family of malware for malicious binaries, etc).

Consequently, to circumvent the scalability issue due to this $\mathcal{O}(N^2)$ complexity, we have preprocessed the spam data set to reduce the number of messages, taking care also of not loosing too much semantics or mixing messages that are quite likely caused by different phenomena. Instead of processing each spam message individually in TRIAGE, we have created *spam events* by grouping all messages sent by the different bots on a given day, and with a given set of keywords in their *subjects* (which should somehow reflect the various campaigns run at different points in time). More precisely, regarding the subject lines of all messages, we have applied the following steps:

- we have first extracted the *most common words* or tokens used in all spam subjects, by simply counting word occurrences;
- for each message, we have then identified those keywords and sorted them by length, assuming that the longest words have usually a more important meaning;
- finally, we have created groups of subjects and assigned messages to those groups by simply looking at messages having at least x words/tokens in common (with $x \in \{1, \ldots, 4\}$).

As a result, we could summarize the ML spam data set into 3,801 spam events comprising at least 30 messages on a given day. These 3,801 spam events account totally for 629,460 messages (70%) of the initial data set. An example of such *spam event* is given here under:

Event Id	Bot Id	Bot name	Subject group	Subject keywords	Date	Nr of msg
13-3990-84	13	`Rustock`	3990	`pfizer`, `now`, `%`	2010-12-24	1,559

5.2 Selection of Spam Features

From Table 5.1, we have selected the most relevant features for performing a multi-criteria analysis, and we have created feature vectors for all 3,801 spam events. These feature vectors are described here after.

5.2.1 Bot-related features

Bot-related features give interesting information on the type of bot that have sent spam messages. The bot signature is obtained by applying CBL [16] rules to the raw network

traffic, since many bots have specific particularities (mostly at the SMTP layer) that can be used to recognize a given bot implementation. However, we haven't used this feature in our analysis for two reasons: i) we have already used them to group spam messages into *spam events* according to the bot signature; and ii) we wanted to verify how accurate those signatures are (or at least if TRIAGE could confirm them), and if it can eventually help to classify messages with an *unknown* signature, by aggregating several other features. Note that about 216,456 messages (23%) didn't match any CBL rule and had thus an *unclassified* bot name.

The next two features characterizing a bot were instead included in the MCDA analysis:

- F_{From}: is a *set* containing the most frequent "From domains", i.e., domains used as `From` field in the email header. For example, for the spam event given here under (13-3990-84), the set of from domains is:

  {`telesp.net.br, iam.net.ma, gaoland.net, embarqhsd.net, ...`}

- F_{P0f}: represents the *distribution* of OS names obtained from the passive OS fingerprinting. For example:

OS name	Count	Percent.
`2000 SP2+, XP SP1+ (seldom 98)`	639	41.0
`2000 SP4, XP SP1+`	669	43.0
`XP/2000 (RFC1323+, w+, tstamp-)`	106	7.0
`XP SP1+, 2000 SP3` (2)	31	2.0
others	114	7.0

5.2.2 Origin-related features

Assuming that each botnet could be characterized by its own army of infected machines, some features related to the origins of spam sending machines may definitively bring some relevant information on the campaigns and activities of each botnet. As a result, we have selected the following features for our analysis:

- F_{geo}: represents the *distribution* of countries of origin of the spam sending machines, obtained by mapping IP addresses to corresponding countries. For example:

Country code	Count	Percent.
US	390	25.0
GB	117	8.0
IN	105	7.0
VN	109	7.0
RU	75	5.0
KR	66	4.0
BR	65	4.0
CA	42	3.0
others	590	38.0

- F_{Host}: represents the *set* of most frequent host names of sending machines, obtained by reverse DNS lookups. Only the TLD and first sub-domain are being considered here, for example:

  {`fpt.vn, airtelbroadband.in, telesp.net.br, hinet.net, shawcable.net ...`}

Note, however, that this type of feature *alone* would certainly not be sufficient to distinguish the various botnets, e.g., due to popular countries or networks that are more infested than others, or simply the fact that certain machines could be infected by multiple bots. Other features related to the origins (IP addresses, Class A-subnets, etc) were not used in this MCDA analysis.

5.2.3 Campaign-related features

To analyze the activities of botnets and the dynamics of spam campaigns run through them, we have also included the following features:

- $F_{Subject}$: represents the *set* of keywords extracted from the subject lines of messages contained in a spam event. For example:

  {`reorders, discount, viagra50/100mg, qua1itymedications`}

- F_{Day}: is the observation day for all messages of a spam event (e.g., 2010-11-27).

- F_{URIs}: represents the *set* of most frequent URIs embedded in the messages of a spam event. For example:

  {`dmediczh.ru, diemedic.ru, hadmedic.ru, didmedic.ru, ...`}

- $F_{Charset}$: represents the distribution of character sets used to encode message content. For example:

Character set	Count	Percent.
us-ascii	982	100.0

- F_{Hour}: represents the distribution of spam messages by hour of the day for a given spam event, which might be useful to find temporal patterns in spam campaigns operations (e.g., the same preferred time frames in the day or in the night for sending messages). For example:

Hour	Count	Percent.
0	297	19.0
2	220	14.0
3	255	16.0
5	142	9.0
7	206	13.0
8	128	8.0
others	311	20.0

The distribution of message sizes and languages were not used in the MCDA analysis. As showed by the cluster analysis in the next Section, those features were not considered as sufficiently discriminant to be helpful in this case for the analysis of spam campaigns.

5.3 Multi-Criteria Analysis

5.3.1 Graph-based clustering

Based on the feature analysis here above, we have performed a graph-based clustering of the 9 spam features described in previous Section. Regarding distance metrics, we have used three different distances, according to the type of feature vector:

1. the *Jaccard* coefficient (equation 3.1) for all feature vectors representing sets of values, i.e., $F_{From}, F_{Host}, F_{Subject}, F_{URIs}$;

2. the *Jensen-Shannon* divergence (equation 2.6), mapped to a similarity in $[0, 1]$ using the transformation of Shepard given in 2.3.2, for all feature vectors representing distributions, i.e., $F_{P0f}, F_{geo}, F_{Charset}, F_{Hour}$;

3. a linear function that maps date differences (expressed in days) to the interval

$[0, 1]$, according to various segments defined by parameters a, b, c:

$$F(d_{ij}, a, b, c) = \begin{cases} 1 & \text{if } d_{ij} \leq a \\ 1 - \frac{d_{ij}-a}{2(b-a)}, & \text{if } a \leq d_{ij} \leq b \\ \frac{1}{2} - \frac{d_{ij}-b}{2(c-b)}, & \text{if } b \leq d_{ij} \leq c \\ 0 & \text{if } d_{ij} \geq c \end{cases} \tag{5.1}$$

where d_{ij} is the date difference in days between two spam events i, j. For this analysis, the parameters of the function a, b, c have been set to 0, 2, 3, respectively.

Before aggregating all features, we have performed a preliminary cluster analysis to have a better idea of the underlying structure of spam events with respect to each feature individually, which might help also to determine their ultimate ability to discriminate different phenomena related to spam campaigns.

An overview of clustering results for bot features is given in Fig. 5.4. The figure shows for each feature the number of clusters (found by the dominant sets algorithm), the total number of events being clustered, the maximum and average cluster size, and the average graph compactness C_p, which gives a hint on how compact the clusters are. The rightmost column of the Table indicates which features have been selected for the MCDA aggregation (in the next step). Not surprisingly, clusters for F_{bot} are perfectly compact, but recall that we did not use that feature for the MCDA analysis (only for labelling purposes). The largest cluster contains 1,537 spam events which are attributed to the `Lethic` botnet.

Regarding F_{From}, only 26% of the spam events could be clustered, and clusters are on average rather small, whereas clusters for F_{P0f} are much larger and not so numerous (only 5 different patterns of OS distributions were found). Not surprisingly, the largest cluster of operating systems has following pattern: `2000 SP4, XP SP1+`: 62%, `2000 SP2+, XP SP1+ (seldom 98)`: 22%, and the remaining 16% refer to variants of `XP/2000`. Most other clusters have very similar OS patterns, except the shapes of the distributions that are different.

The clustering results for the origin-related features are given in Fig. 5.5 (note that the distributions of Class A-subnets - F_{classA} - have not been considered in the MCDA aggregation). In that figure, observe that the average compactness values are not very high for features F_{geo}, F_{Host}, and the total amount of clustered spam events is rather low, which seems to indicate that spam events have in general very diverse origins, leading to distributions of countries/hosts that are difficult to group, at least from a statistical viewpoint (i.e., according to Jensen-Shannon divergence). However, the cluster patterns for F_{geo} are still quite different from each other, with distributions of countries mapped

Feature	Type	Nr of clusters	Nr of events	Max. Size	Avg Size	Avg Cp	Aggreg.
F_{bot}	val	22	3,783 (99%)	1,537	199	1.0	
F_{From}	set	37	1,011 (26%)	330	27	0.69	✔
F_{POf}	distri	5	3,154 (83%)	2,267	630	0.77	✔

Figure 5.4: Cluster analysis of spam bot-related features.

to quite different regions of the globe. For example, the global distributions of the two largest geographical clusters are respectively:

(i) US: 15%, BR: 7%, IN: 6%, VN: 6%, GB: 5%;

(ii) RU: 14%, VN: 15%, IN: 11%, BR: 5%, UA: 8%, ID: 8%, PK: 5%, KZ: 3%.

Feature	Type	Nr of clusters	Nr of events	Max. Size	Avg Size	Avg Cp	Aggreg.
F_{Geo}	distri	90	1,368 (36%)	92	15	0.37	✔
F_{Host}	distri	22	1,969 (52%)	381	89	0.25	✔
F_{ClassA}	distri	38	869 (22%)	105	23	0.39	

Figure 5.5: Cluster analysis of features related to spam origins.

Next, Fig. 5.6 gives an overview of clustering results for the campaign-related features. $F_{subject}$ is quite likely a very interesting characteristic, since spam campaigns are usually targeting specific businesses or categories. From Fig. 5.6, it appears that clusters of subject keywords are on average not so large but still very compact (with an average C_p close to 1), and about 59% of the data set could be grouped into subject clusters. Not surprisingly, most of these subject clusters have keyword patterns referring to the pharmaceutical business (with keywords like `qua1itymedications`, `pills`, `pfizer`, `viagra`, `hydrocodone`, `vicodin`, `noprescription`, ... and all possible spelling variations of those terms) and more generally, to advertisements for product discounts (with keywords like `rolex`, `shipping`, `discount`, `better prices`, `...% off`, etc). Some of those clusters also reflect the different other categories mentioned previously in Fig. 5.3.

Clustering with respect to F_{day} was apparently quite straightforward. This feature can be useful to link spam events that might be part of the same large-scale campaign when they occurred roughly in the same time period. Similarly, the distribution of spam

sent by hour – i.e., F_{Hour} – can help to link spam events that are possibly related to a same campaign when they exhibit similar-looking hour patterns, either on a same date or maybe on consecutive days. This makes the assumption that the bot owner could decide to run his campaign during preferred hours of the day (or in the night), which can also be influenced by also the region of the world that is being targeted.

Even though there are only 5 different patterns with respect to $F_{charset}$ (i.e., the distributions of character sets used for encoding spam content), these clusters are still very compact and might indicate some bias either in certain bot implementations, or in spam campaigns operations. For example, campaigns targeting russian-speaking people will most likely have their content encoded using the `koi-8r` character set. Whereas campaigns or bots that are mostly used to target European people will probably use character sets like `utf-8` or `iso-8859-1`. Finally, to target people living in North-America, the character set `us-ascii` is the most likely to be used. So it looks appropriate to include this feature in the MCDA aggregation, even though this characteristic is probably not very useful if considered separately from all other spam features.

In conclusion, as interesting as these clustering results may be, we have at this point no real information on which botnet(s) are used for which kind of campaigns, and how, i.e., what are the different economic models used by the various botnets. In the next Section, we will thus aggregate all features using the MCDA approach.

Feature	Type	Nr of clusters	Nr of events	Max. Size	Avg Size	Avg Cp	Aggr.
$F_{subject}$	set	129	2,248 (59%)	84	18	0.92	✔
F_{day}	date	50	3,084 (81%)	156	62	0.96	✔
F_{URIs}	set	90	850 (22%)	51	10	0.82	✔
$F_{Charset}$	distri	5	2,798 (74%)	1,764	560	0.84	✔
F_{Lang}	distri	1	3,321 (87%)	3,321	3,321	0.99	
$F_{MsgSize}$	distri	12	65 (1%)	11	5	0.48	
F_{Hour}	distri	58	2,005 (53%)	293	34	0.41	✔
F_{Attach}	set	5	35 (<1%)	11	7	0.60	

Figure 5.6: Cluster analysis of campaign-related spam features.

5.3.2 MCDA aggregation

Similarly to the two previous applications of TRIAGE in Chapters 3 and 4, we may want now to "connect all dots" by aggregating all viewpoints given by the 9 spam features selected here above. Remind that the objective is to get better insights into the spam botnets ecosystem and spam campaigns dynamics.

As a first exploratory approach, we have applied the OWA operator, which is the most straightforward and easy-to-use method to compute the agregation. We have defined following weighting vector to reflect our expectations on the minimum amount of features required to link groups of spam events to a same spam campaign run by a given type of bots:

$$\mathbf{w} = [0, 0, 0.10, 0.20, 0.40, 0.20, 0.10, 0, 0]$$

By aggregating all similarity scores with this OWA vector, we completely ignore the influence of the two highest scores, no matter *which* features they are related to. We start then to give some importance to the third highest score, with the highest weights given to the fourth and fifth highest positions. In other words, at least four strong correlations will be needed in order to have a global score above a decision threshold of 0.5. Recall that we do not need to specify in advance which combination(s) of four spam features (out of nine) are required to link two spam events together.

Quite similarly to what we did in previous analysis (chapter 4), we also tried to model our domain knowledge on spam botnets and their campaigns using a fuzzy integral based on Choquet (as defined in Section 2.4.5) to aggregate all spam features. Here again, it would be too tedious and error-prone to define a fuzzy measure involving 9 criteria manually, as 2^9 combinations must be defined in that case. Fortunately, it is much easier to define a 2-additive fuzzy measure, without compromising the richness and flexibility of our aggregation model. Recall that the analyst only has to define the overall *importance factors* for each feature separately (i.e., the Shapley values defined in 2.7), and then to add some *interactions* among pairs of features (such as redundancies or synergies) if he wants to enrich the model. As a result, the complexity of the sub-model represented by such a 2-additive measure is reduced to the definition of *at most* $n(n+1)/2$ weighting factors. In most cases, an analyst does not need to define all interactions among pairs of features; the ones that are not defined will simply be considered as independent (i.e, no interaction).

To define our 2-additive fuzzy measure, we have first defined the following *importance factors*:

F_{P0f}	F_{Geo}	$F_{Subject}$	F_{Day}	$F_{Charset}$	F_{Hour}	F_{From}	F_{Host}	F_{URIs}
0.05	0.05	0.20	0.15	0.10	0.10	0.15	0.10	0.10

To enrich the model, we have then added certain *interactions* among following pairs of features:

(F_{Geo}, F_{Host})	$(F_{Subject}, F_{Day})$	$(F_{Subject}, F_{Hour})$	$(F_{Subject}, F_{From})$	(F_{Day}, F_{URIs})
- 0.05	0.06	0.04	0.15	0.10

Once again, these interaction values were automatically computed by TRIAGE to enforce the additivity and monotonicity conditions given in expression 2.22. The analyst simply needs to set a *relative* amount of synergy or redundancy between pairs of features. In this case, we have set between 20 and 50% of synergy for the last four pairs here above (with positive indices), and 50% of redundancy for the pair (F_{Geo}, F_{Host}). Then, to compute the Choquet integral starting from those importance factors and interaction values, we have used the formula given by expression 2.21.

As usual, we need to set a value for the decision threshold ϵ, which is used to remove unwanted edges in the aggregated graph, and finally to identify the multidimensional clusters (MDCs) via the connected components algorithm. A sensitivity analysis is required for ϵ to determine the best ranges of values according to the number of MDCs, total percentage of clustered samples and the distribution of cluster sizes obtained with various values of the decision threshold.

The result of the sensitivity analysis for the OWA aggregation is illustrated in Fig. 5.7, where we can see that values of ϵ starting from 0.30 up to 0.40 seem to be appropriate, according to the number of MDCs and total amount of samples that are clustered. We have chosen here a decision threshold of 0.32 in order to have as many spam events as possible, without grouping too many of them within the same MDC.

In Fig. 5.8, we can observe that the 2-additive Choquet aggregation gives fairly similar results, although the number of MDCs is higher than with OWA. In this analysis, we have also selected 0.32 as decision threshold for the Choquet aggregation, so that we don't loose too many spam events by removing edges.

In Table 5.2, we compare the global performance of OWA and Choquet aggregations. Overall, both methods give fairly similar results in terms of MDCs, average compactness (C_p) and total number of samples clustered in MDCs. Since we have chosen for a lower decision threshold ϵ, we can observe that the average compactness (computed over of all MDCs) also reflects this choice. Considering the aggregation parameters that we have set, such a value for ϵ also means that we have lowered the number of strongly correlated features required to link spam events together in the same MDC (which, in this case, will be closer to 3 instead of 4). In the rest of the analysis, we will now focus on the MDCs given by the OWA aggregation only.

To evaluate the consistency of individual MDCs, Fig. 4.3 represents the global graph compactness of MDCs (where each color in the bar chart refers to the average compactness index C_p of a single feature). Besides the two first MDCs, we observe that they all

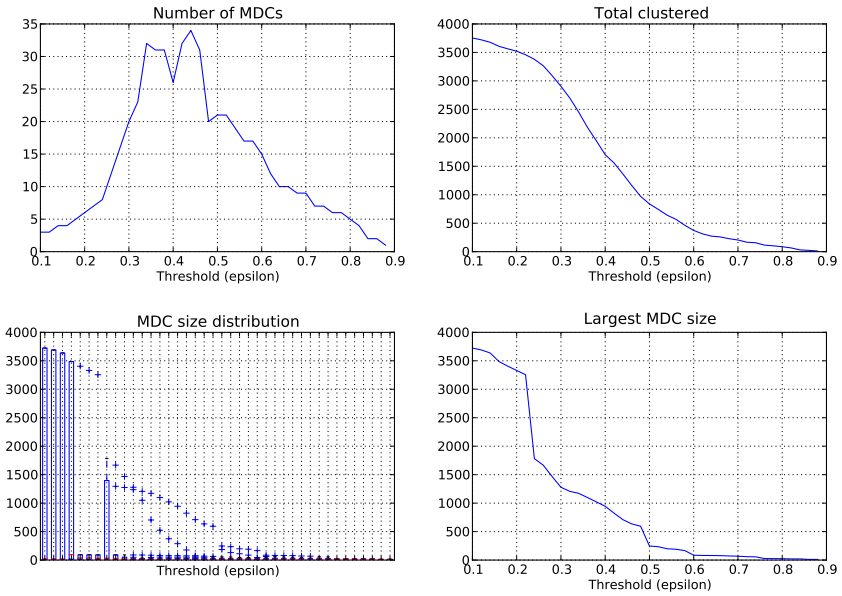


Figure 5.7: Sensitivity analysis of the decision threshold ϵ for the **OWA** aggregation.

Table 5.2: Comparison of OWA and Choquet aggregation methods for the ML spam data set.

Characteristic	**OWA**	**Choquet**
Threshold ϵ	0.32	0.32
Nr of MDCs	23	39
Total clustered	2,699 (71%)	2,798 (74%)
Largest MDC	1,207	1,207
Average C_p	0.36	0.35

have on average *at least 3 features* with a high C_p, and have globally a total compactness value above 3. This seems to be consistent with the constraints we have previously modelled regarding aggregation parameters.

In the same chart, it is easy to see that F_{P0f}, $F_{Charset}$ (and to a lesser extent, F_{Day}) have always very high C_p values, and are thus strongly correlating features for all MDCs. Note, however, that the precise patterns of MDCs w.r.t those features are still likely to

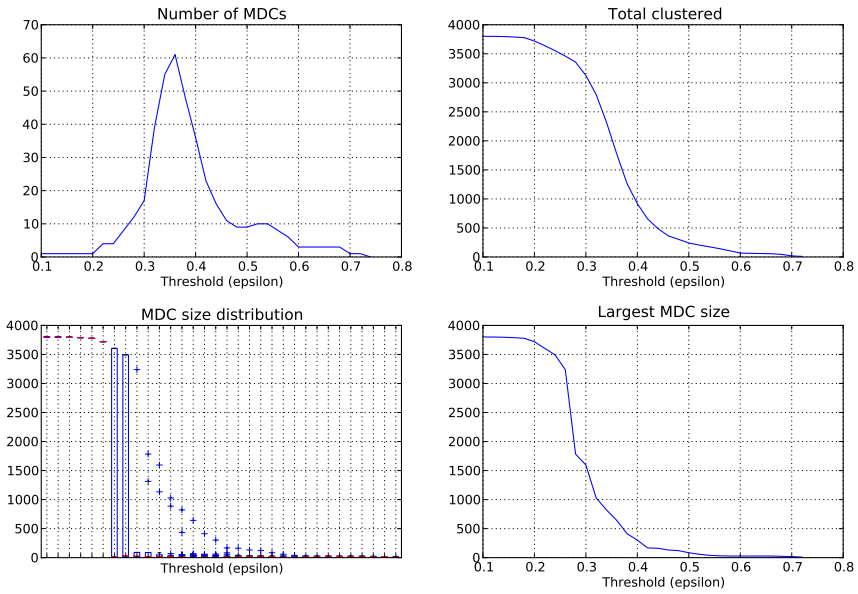


Figure 5.8: Sensitivity analysis of the decision threshold ϵ for the **Choquet** aggregation.

be different from one MDC to another. On the other hand, the origin-related features (F_{Geo}, F_{Host}) and F_{From} have usually much lower compactness values. We can also observe the light interdependence we were somehow expecting between F_{Host} and F_{Geo}, i.e., the presence of F_{Host} as correlating feature involves usually F_{Geo} to have also high C_p scores.

It is quite striking to see that many MDCs have highly similar patterns regarding the combination of C_p values, like MDCs identified by ID's 4 to 9 and 11 to 23, whereas MDCs 1, 2, 3 and 10 have a very different composition of features. Regarding F_{URIs}, it seems that embedded URIs are a strong characteristic starting from MDC 4 (i.e., for smaller campaigns), except for MDC 10.

Finally, the two first MDCs may not appear a priori as meaningful as the other ones, because of their rather low value of compactness for most features. However, since these are also the two largest MDCs, this low global C_p value may be skewed by the connected component algorithms that might have identified, for each feature, loosely coupled subgraphs within the aggregated graph (due to a "chaining effect"), which form sorts of weakly interconnected "bubbles" of spam events within each single-feature graph.

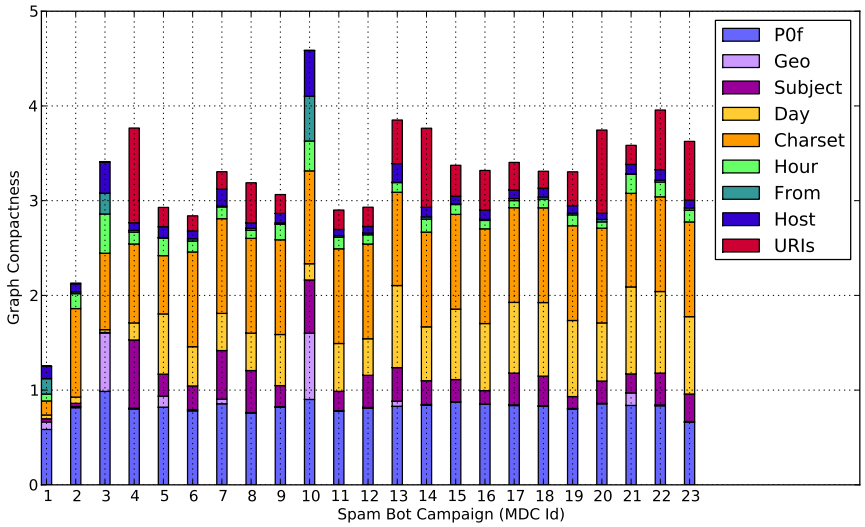


Figure 5.9: Evaluation of the MDCs found by aggregating all spam and bot features using the OWA operator. Each color in the bar chart refers to the average compactness index C_p of each individual feature.

5.4 Analysis of Spam Bot Campaigns

To illustrate the kind of insights we can automatically obtain from such an MCDA analysis, we provide in this Section a more in-depth analysis of multi-dimensional clusters (MDCs) representing various spam campaigns performed by different botnets. Fig. 5.10 gives an overview of the characteristics of the 10 largest MDCs found by TRIAGE. Interestingly, besides the two first MDCs, each campaign is attributed to a single botnet, which already shows that the classification performed by TRIAGE seems to be consistent with the classification obtained from CBL rules. However, recall that those rules were not used in the MCDA analysis, and thus we already show that it is perfectly possible to classify spam using our multi-criteria approach, by combining different spam features such as those used for this analysis.

MD Cluster	Botnet(s)	Nr Events	Nr Msg	Nr dates	Duration (days)	Tot. Cp
1	**Rustock** (42%), Cutwail (6%), **Unclassified** (36%), Grum (12%), MegaD (4%), DonBot (<1%)	1,207	451,729	89	92	1.25
2	**Lethic** (67%), **Maazben** (26%), Unclassified (7%)	1,050	56,135	86	92	2.13
3	Xarvester	89	45,146	82	92	3.41
4	Lethic	35	1,535	17	17	3.76
5	Bagle	33	1,665	7	22	2.93
6	Lethic	29	1,150	7	7	2.84
7	Grum	24	1,348	6	7	3.31
8	Lethic	23	824	8	8	3.19
9	Lethic	21	923	5	6	3.06
10	Cutwail	20	1,316	17	17	4.59

Figure 5.10: Overview of the 10 largest Multi-Dimensional Clusters (MDCs) representing various spam campaigns run by different botnets.

Finding 1: Botnets inter-relationships

The first MDC contains a mix of spam events that are largely attributed to `Rustock` (42%), but also to `Grum` (12%), `Cutwail` (6 %) and `Mega-D` (4 %), whereas MDC 2 seems to indicate some interconnection between `Lethic` and `Maazben`. As explained in previous Section, the two largest MDCs result from a *chaining* effect among different compact subgroups of spam events that are weakly interconnected by a number of common features (which also explains their low overall compactness value). However, it also indicates that certain spam botnets are tightly interconnected, and they may run very similar spam campaigns in the same period of time. For example, by looking at the bot distribution of MDC 1, although `Rustock` has the largest share of spam events, it seems difficult to distinguish campaigns run by this botnet from the ones run by other botnets like `Cutwail`, `Grum` or `Mega-D`, simply because these other botnets may sometimes launch campaigns with the very same subject keywords, using the same character sets, etc. These different botnets are also sharing a large number of hosts that may be infected by several bots and are thus running campaigns launched by different botnets in parallel. Another likely explanation is that people controlling those botnets are ei-

ther collaborating (e.g., "load balancing" the various spam campaigns on the different botnets they are controlling) or they are working for the same client groups.

To illustrate this point, Fig. 5.13 represents all relationships among spam events of MDC 1 (light blue nodes in the graph) with respect to F_{Bot} (red nodes) and $F_{Subject}$ (purple nodes). This figure clearly illustrates the "bubble effect" underlined here above, where spam events attributed to different botnets are still interconnected by common sets of subject keywords (like `rolex.com`, `% off`, `for you` in the middle of the graph that connect `Rustock` to `Cutwail` and unknown bots). Similarly, in Fig. 5.14 representing MDC 2, we can observe the interconnections between `Lethic` and `Maazben` (the two red nodes inside the large circle), and some interconnections with unclassified bots as well (the small sector outside the circle), with respect to the subject keywords used in the campaigns run by those botnets.

However, it is important to note that, because of the multi-criteria aggregation, subjects keywords are not the only feature that "glues" those spam events together. When looking at other dimensions (like bot origins, URIs, From domains and charsets), we can observe very similar patterns. In other words, most of the "gluing events" that tend to interconnect botnets have at least a number of other features in common. For example, Fig 5.15 illustrates all relationships between spam events of MDC 1 with respect to F_{Host} (i.e., the host names of the machines from which the bots are sending spam). In this graph with a radial layout, we can also observe that a significant number of host names (represented by green nodes) are shared by unclassified bots and `Rustock`, whereas a few number of spam events (in light blue) attributed to `Cutwail`, `Grum` and `Mega-D` are also pointing to host names used by `Rustock`. We have also observed the same phenomenon for feature F_{Geo} (countries of origin).

Finding 2: Attribution of Unclassified Spambots

Another interesting finding of this analysis is that TRIAGE can help to attribute spam events having an *unclassified* bot signature to a known botnet. This can be visualized from previous graphs, in which many unclassified bots are sharing a number of features with spam events of `Rustock` (like subject keywords in Fig. 5.13) or `Maazben` (Fig. 5.14). Note that, in those graphs, we have represented only one or two different features; however, we could easily identify, within an MDC comprising unclassified bots , which spam events with an "unclassified" bot signature share at least three or four features with `Rustock`, `Maazben`, or any other bot having a specific CBL signature.

Finding 3: Dynamics of Spam Campaigns

By looking at other MDCs of the Table in Fig. 5.10, we could easily infer some interesting information regarding the dynamics of spam campaigns run by different botnets.

Campaigns performed by `Rustock` are apparently *long-lived* and somehow stable, whereas campaigns run by `Lethic` and `Maazben` are instead rather *short-lived* (they last on average 7 days), and have a sort of *polymorphic* behavior with respect to various features (e.g., a different set of disposable URIs is used every day, and the set of From domains is also constantly changing). This polymorphic behavior of `Lethic` can be inferred from the large number of small and short-lived campaigns found by TRIAGE and represented in Fig. 5.10. This is further illustrated in Fig. 5.11 and Fig. 5.12 for the MDC 6, where we can clearly see in the first figure the use of different sets of URIs (nodes in cyan), and in the second figure, the use of different sets of From domains (nodes in orange) on each day of this 7 days-campaign (represented by pink nodes). All other MDCs attributed to `Lethic` and `Maazben` in Fig. 5.10 have very similar patterns. It is also interesting to note that most URIs used by these botnets have a `.ru` top-level domain.

Finding 4: "Specialized" Botnets and Other Specificities

Specialized botnets

By looking at the specific subject patterns found in the MDCs representing various bot campaigns, we have also observed that most botnets seem to be specialized in a specific type of activity, i.e., each botnet is mostly distributing a certain spam category. For example:

- `Lethic` and `Maazben` are mostly concerned with spam campaigns relating to pharmaceutical products. Some examples of frequently observed keywords are related to topics like `viagra50/100mg, qua1itymedications, hydrocodone, lorazepam, codeine, herbal, p**enlargement, pills, pharmacy,` .... However, Maazben is sometimes involved also in other campaigns for selling replica watches (see Fig. 5.14 for more details on interconnections between those two botnets).

- the `Rustock/Cutwail` botnet family seems to be more frequently involved in campaigns for the advertisement of any kind of sales or discounts on various products (with keywords like `r0lex, % off, discount, chaper today, sales, suberb`, ... Note also the intentional typos in the words, probably as an attempt to fool anti-spam filters). Rustock can also advertise some very targeted pharmaceutical products, such as `viagra` and `pfizer`.

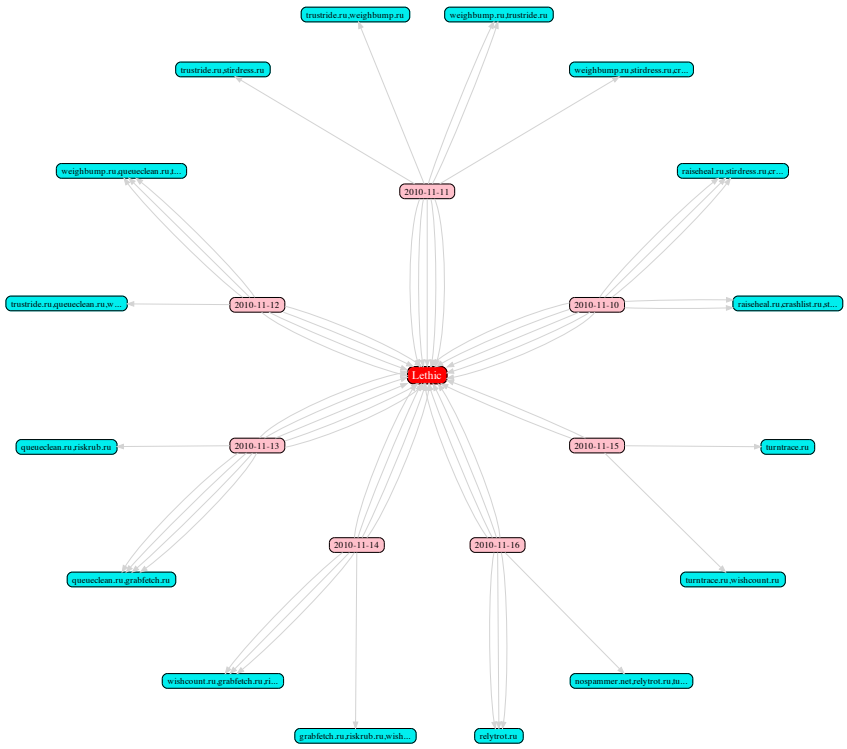


Figure 5.11: Graph with radial layout representing MDC 6 (`Lethic` campaign) with respect to *embedded URIs* (nodes in cyan) and the observation dates (nodes in pink). In this 7-days spam campaign, we see that different sets of disposable URIs are being used every day during the campaign, which illustrates the typical "polymorphic" behavior of this botnet.

- `MegaD` and `DonBot` are often involved in campaigns relating to job offers (the *Jobs/Mules* spam category mentioned in Fig. 5.3). Examples of keywords are `administrator, manager, international, company, vacancy`, ...

Xarvester

The MCDA results have also highlighted interesting specificities for a botnet called `Xarvester`. All spam events attributed to this botnet (based on CBL rules) have also been grouped by TRIAGE in a single cluster (MDC3). The reasons for this may be found in the specificities of the MDC, which are also visible from the compactness values of

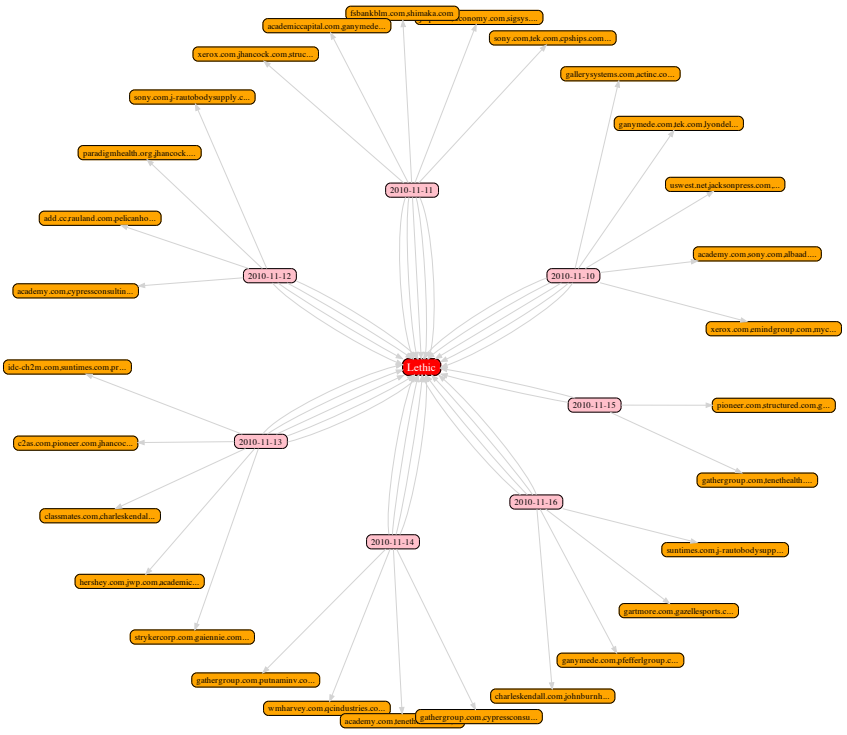


Figure 5.12: Graph with radial layout representing MDC 6 (`Lethic` campaign) with respect to the "*From domains*" used to send spam (nodes in orange) and the observation dates (nodes in pink). Different sets of domains are being used every day to send spam, which illustrates once again the "polymorphic" behavior of `Lethic`.

individual features in Fig. 5.9. First, we observe that the spam sent by this botnet is encoded with a very particular character set, namely `windows-1251`, which was not seen as being used by any other botnet. Regarding the origins of the bots, Xarvester has also some specific patterns of countries and host names, with a significant number of bots located in Bulgaria (11%), Great Britain (13%), Italy (13%), Greece (5%) and Poland (5%). Regarding the subject keywords, 92% of spam sent by `Xarvester` had junk content or non printable characters in the subject lines (probably because of a weird character encoding). Perhaps even more conclusively, most `Xarvester` bots (92%) have the same unique `P0f` OS signature, referring to a `2.5-7 (2)` Solaris or Linux OS kernel (opposed to the various Windows OS signatures usually seen for most other botnets). Finally,

76% of spam sent by Xarvester bots had apparently no URI embedded in the message content, and the other 24% had URI's pointing to a limited number of domains located in the `.ru` TLD.

Use of shortened URLs

Some other insights that we could easily obtain from the MCDA results are related to the `Bagle` botnet, which is represented by MDC 5 in Fig. 5.10. The specific patterns of this MDC that explain its creation by the MCDA algorithm are the following ones:

- a specific distribution of OS names for the bots, i.e.: 2000 SP4, XP SP1+ (71%) and 2000 SP2+, XP SP1+ (seldom 98) (14%);
- a specific distribution of countries of origins of the bots, i.e.: Russia (30%), Ukraine (8%), India (6%), Colombia (6%), Romania (5%) and Venezuela (5%);
- a specific character set used to encode spam messages, i.e.: `iso-8859-1` (100%);
- some specific keywords in the subjects, with the two most frequent ones being `prescript` and `medications`.

Another extremely interesting characteristic of this botnet campaign appears when looking at the distribution of URIs embedded in the messages. Most of those URIs make use of *shortened URL's* services, the most frequently used ones being `bit.ly`, `tinyurl.com`, `miniurl.com`, `vl.am` and `j.mp`. This phenomenon was also reported by MessageLabs in their intelligence report [50] starting from late August. However, the botnets responsible for this rise in the proportion of spam that uses short URLs were in that time `Cutwail` and `Grum`, whereas we have only observed this characteristic for `Bagle` starting from December 10, 2010 until December 31, 2010. Among the shortened URL's embedded in this `Bagle` spam campaign, we also found a significant portion or URIs pointing to `retwt.me`, which refers to *TweetMeme*, a service that aggregates all the popular links on *Twitter* to determine and publish the most popular ones. This specific behavior was not observed for any other botnet within our analysis period.

5.5 Summary

In this Chapter, we have showed how TRIAGE was used to analyze the *spam campaigns* performed through various spam botnets.

Thanks to the MCDA approach implemented in TRIAGE, we have demonstrated how we could gain insights into the *spam botnet* ecosystem, and how we could have a better

understanding of the dynamics of spam campaigns performed through them. In particular, we have highlighted some interesting relationships among families of bots (such as the `Rustock/Grum/Cutwail` family, or the `Lethic/Maazben` family), as well as several other specificities of certain botnets compared to others. We have also pointed out that there were some significant differences in the strategic behavior and the economic models adopted by people controlling those different botnets.

Finally, we have also showed that TRIAGE could be used to help classify spam messages sent by bots with *unknown signature* thanks to the unique combination of multiple spam features as modelled by the multi-criteria aggregation process, and without relying on third-party signatures or external rules.

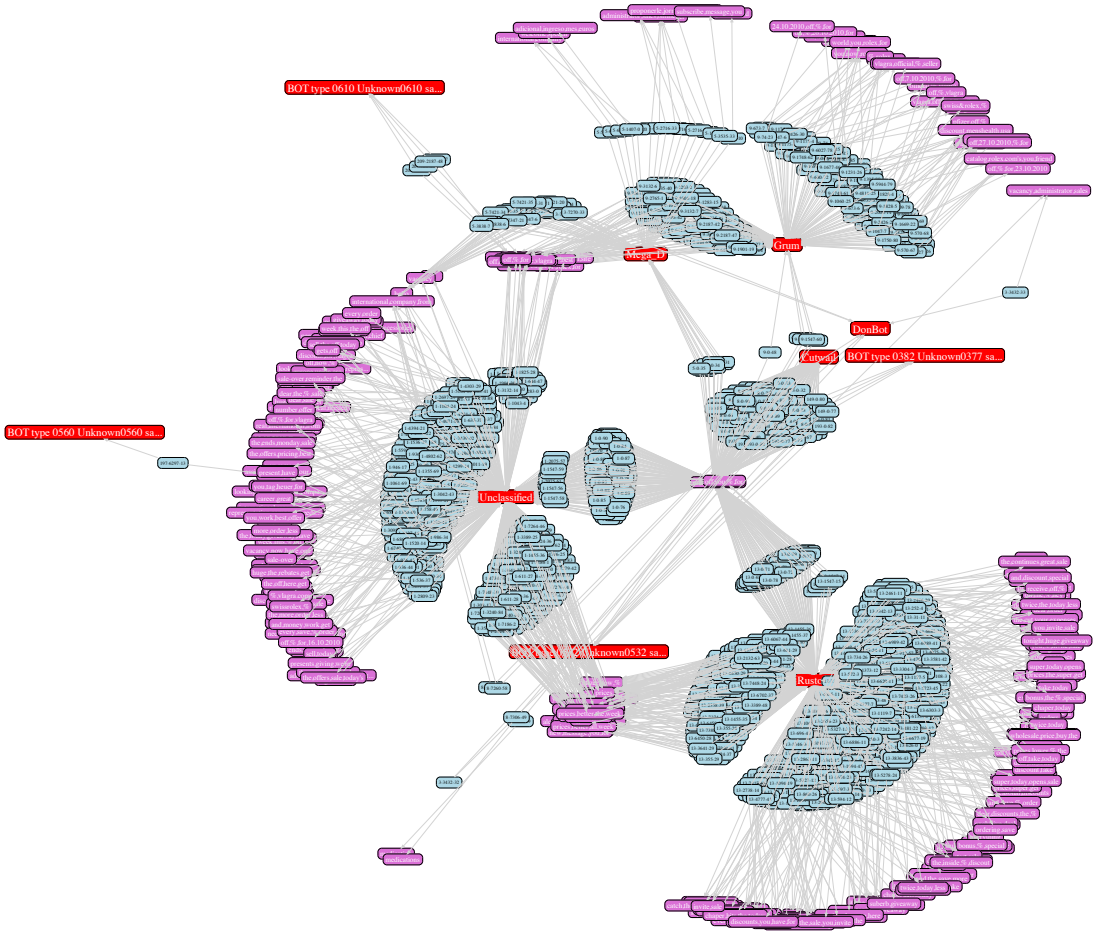


Figure 5.13: Graph with force-directed layout representing all relationships between spam events (nodes in light blue) belonging to MDC 1 (`Rustock/Grum/Cutwail/MegaD`) regarding the subject keywords (purple nodes).

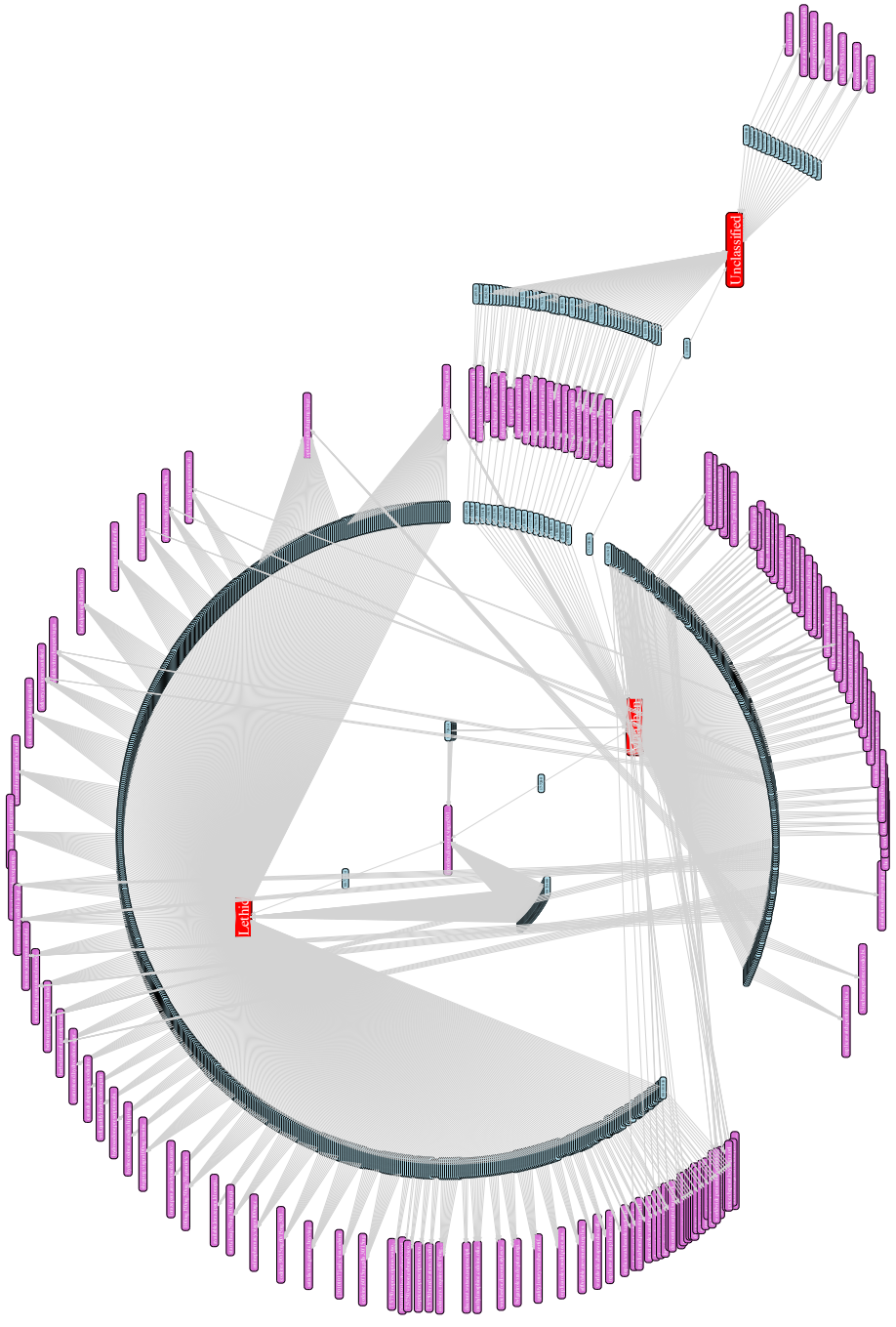


Figure 5.14: Graph with radial layout representing all relationships between spam events (nodes in light blue) belonging to MDC 2 (`Lethic/Maazben`) regarding the subject keywords (purple nodes).

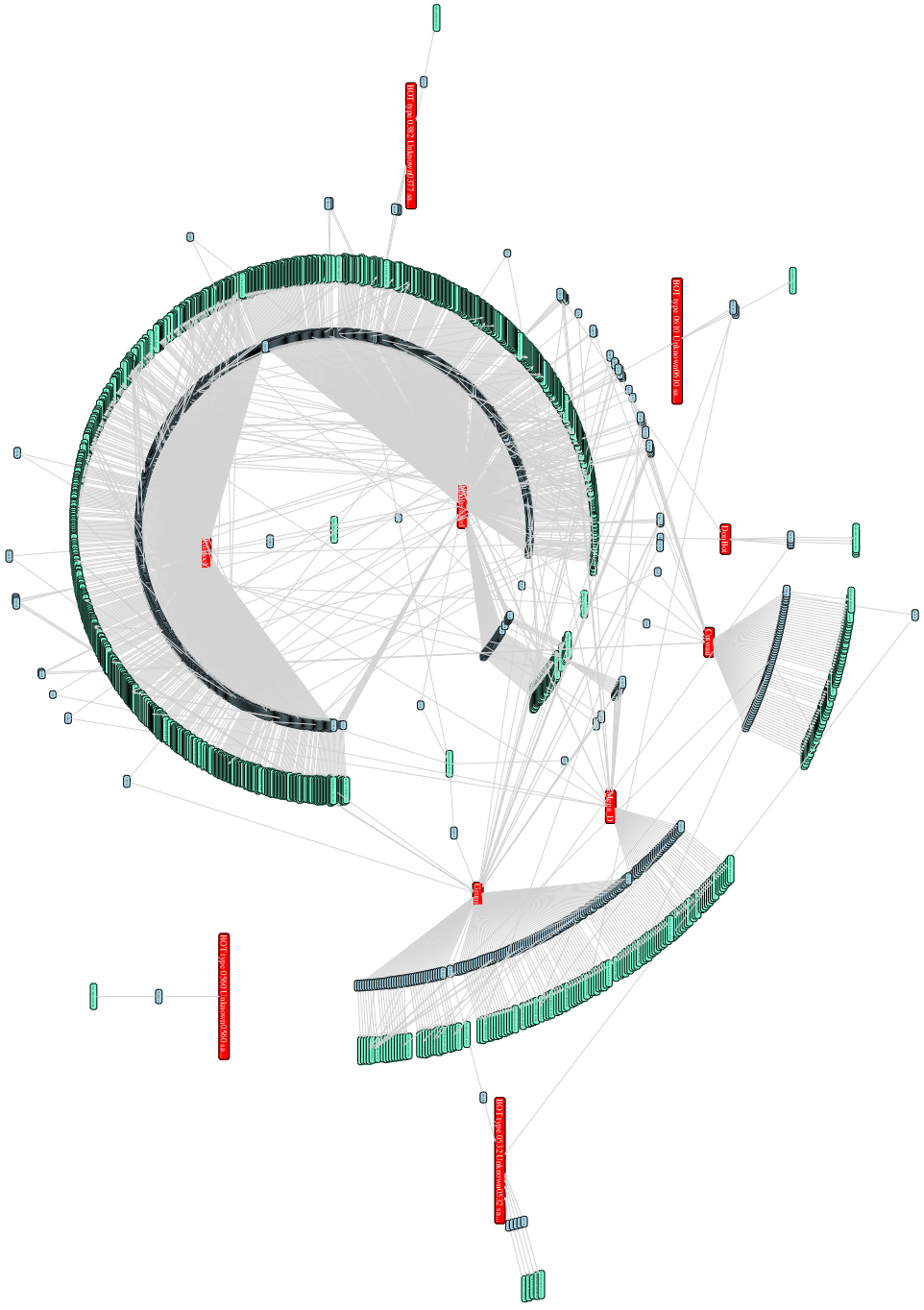


Figure 5.15: Graph with radial layout representing all relationships between spam events (nodes in light blue) belonging to MDC 1 (`Rustock/Grum/Cutwail/MegaD`) regarding the the host names of spamming machines (green nodes).

6 Conclusion

In this final deliverable for Workpackage 5 (*Threats Intelligence*), we have offered an extensive description of all experiments carried out with respect to root cause analysis techniques.

The R&D efforts carried out in WP5 have produced TRIAGE, a generic, multi-criteria software analysis framework for intelligence and root cause analysis in cyber security. As extensively described in Chapter 2, TRIAGE relies on a novel combination of graph-based analysis with a data fusion process inspired by Multi-Criteria Decision Analysis (MCDA).

We have then described how this framework was successfully applied to various WOMBAT datasets to perform intelligence analyses, by taking advantage of several structural and contextual features of various data sets developed by the different partners. These experiments have enabled us to get insights into the underlying root phenomena that have likely caused many security events observed by sensors deployed by WOMBAT partners. In particular, we have described in Chapter 3 how TRIAGE was used to analyze Rogue AV campaigns and the modus operandi of people organizing them. In Chapter 4, we have then described how we have applied the framework to a completely different data set, made of SGNET code injections, with the purpose of analyzing different malware variants attributed to the *Allaple* worm propagation scheme.

Finally, in Chapter 5, we have described another experiment performed on a large spam data set obtained from *Message Labs* (now *Symantec.Cloud*), for which we have used TRIAGE to analyze spam botnets and their ecosystem to get a better under understanding of how those botnets are used by spammers, and how they organize and coordinate spam campaigns. It is worth mentioning that we are considering a possible technology transfer of TRIAGE to *Symantec.Cloud*, who is interested in carrying out regular intelligence analyses of their spam data sets, and may also consider the integration of TRIAGE to their *Skeptic*® spam filtering technology, as a way to improve the various heuristics used in their spam analysis system.

Perspectives

As demonstrated throughout this deliverable, TRIAGE has enabled us to get new insights into the underlying root phenomena that have likely caused many security events

observed by various sensors (e.g., honeypots, crawlers, spamtraps, sandboxes, etc). However, the development of the TRIAGE framework has also opened a number of interesting new challenges that could be investigated in future research.

As a first new research axis, we could envision and develop new clustering techniques to improve the *scalability* of the framework. To further address this scalability issue due to the $\mathcal{O}(N^2)$ complexity of graph-based representations, we could also rely on parallel algorithms, such as the `MapReduce` paradigm.

To further improve the *data fusion* process done by TRIAGE, we could also consider other fusion methods, such as methods based on probabilistic theories (e.g., Dempster-Shafer theory of evidence or belief networks), or new classes of aggregation functions and fuzzy logic approaches.

Next, we could also add an *anomaly detection* capability to the TRIAGE framework to be warned as soon as possible when TRIAGE observes new kinds of attack phenomena that are significantly different from previous ones. This would also enable us to be quickly alerted of significant changes in the behaviors of attackers.

Finally, we have showed throughout this document how different visualization techniques, such as graph-based visualizations or dimensionality reduction, could help us to achieve a better *situational awareness* in network and information security. However, because of the large number of dimensions, it is not always obvious to the analyst to see directly *why* security events have been attributed to the same phenomenon. In a system where interactive visual analytics is combined with data fusion and attack attribution algorithms, the analyst would be better equipped to gain insight into attack phenomena. This means that we need to tightly couple network security algorithms with directly integrated visual analysis methods in the future. Further improvements of the scalability of both the data analysis algorithms and visualizations are also necessary to eventually reach this goal.

We hope that all those different aspects of improvement will be largely addressed in the new EU-FP7 project called VIS-SENSE (`http://www.vis-sense.eu`).

Bibliography

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. pages 420–434. 2001.

[2] J. G. Auguston and J. Minker. An analysis of some graph theoretical clustering techniques. *ACM*, 1970.

[3] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID'07)*, September 2007.

[4] P. Barford and V. Yegneswaran. *An Inside Look at Botnets*. Advances in Information Security. Springer, 2006.

[5] D. Barroso. Botnets - The Silent Threat. In *European Network and Information Security Agency (ENISA)*, November 2007.

[6] M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Process.*, 18(4):349–369, 1989.

[7] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, Behavior-Based Malware Clustering. In *16th Annual Network& Distributed System Security Symposium*, 2009.

[8] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel. Insights into current malware behavior. In *LEET'09: 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 21, 2009, Boston, MA, USA*, Apr 2009.

[9] G. Beliakov. Shape preserving splines in constructing wowa operators: comment on paper by v. torra in fuzzy sets and systems 113 (2000) 389-396. *Fuzzy Sets Syst.*, 121(3):549–550, 2001.

[10] G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Berlin, New York, 2007.

[11] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[12] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc. 35, 99–109*, 1943.

[13] F. Boutin and M. Hascot. Cluster validity indices for graph partitioning. In *IV'04, 4th International Conference on Information Visualization, 2004,*, 2004.

[14] J. Canto, M. Dacier, E. Kirda, and C. Leita. Large scale malware collection: Lessons learned. In *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*, 2008.

[15] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1953.

[16] Composite Blocking List. `http://cbl.abuseat.org`, 2011.

[17] Computer Crime Research Center. Cybercrime is an organized and sophisticated business. Available online at `http://www.crime-research.org/news/20.02.2006/1835/`, Feb 2006.

[18] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005 Workshop)*, Cambridge, MA, July 2005.

[19] M. Cova, C. Leita, O. Thonnard, A. D. Keromytis, and M. Dacier. Gone rogue: An analysis of rogue security software campaigns (invited paper). In *Proceedings of the 5th European Conference on Computer Network Defense (EC2ND)*, November 2009.

[20] M. Cova, C. Leita, O. Thonnard, A. D. Keromytis, and M. Dacier. An analysis of rogue av campaigns. In *Proceedings of the 13th international conference on Recent advances in intrusion detection*, RAID'10, pages 442–463, Berlin, Heidelberg, 2010. Springer-Verlag.

[21] J. R. Crandall, Z. Su, and S. F. Wu. On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In *12th ACM conference on Computer and Communications Security*, pages 235–248. ACM Press New York, NY, USA, 2005.

[22] L. Daigle. WHOIS protocol specification. RFC3912, September 2004.

[23] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis: Fourth Edition*. Hodder Arnold, 2001.

[24] F-Secure. Malware information pages: Allaple.a, `http://www.f-secure.com/v-descs/allaplea.shtml`, December 2006.

[25] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. pages 31–, June-2 July 2004.

[26] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89:445–456, 1996.

[27] M. Grabisch. Alternative representations of discrete fuzzy measures for decision making. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 5(5):587–607, 1997.

[28] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets Syst.*, 92(2):167–189, 1997.

[29] M. Grabisch. The interaction and möbius representations of fuzzy measures on finites spaces, k-additive measures: a survey. *Fuzzy Measures and Integrals. Theory and Applications, M. Grabisch, T. Murofushi, and M. Sugeno (eds)*, 124(1):70–93, 2000.

[30] M. Grabisch and C. Labreuche. A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 2009.

[31] M. Grabisch, T. Murofushi, M. Sugeno, and J. Kacprzyk. *Fuzzy Measures and Integrals. Theory and Applications*. Physica Verlag, Berlin, 2000.

[32] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.

[33] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.

[34] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[35] M. Jakobsson and Z. Ramzan. *Crimeware: Understanding New Attacks and Defenses*. Addison-Wesley, 2008.

[36] W. P. Jones and G. W. Furnas. Pictures of relevance: a geometric analysis of similarity measures. *J. Am. Soc. Inf. Sci.*, 38(6):420–442, 1987.

[37] B. Krebs. Massive profits fueling rogue antivirus market. Washington Post, published online at `http://voices.washingtonpost.com/securityfix/2009/03/obscene_profits_fuel_rogue_ant.html`, March 2009.

[38] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951.

[39] C. Leita. *Automated protocol learning for the observation of malicious threats.* PhD thesis, Université de Nice-Sophia Antipolis, December 2008.

[40] C. Leita, U. Bayer, and E. Kirda. Exploiting diverse observation perspectives to get insights on the malware landscape. In *DSN 2010, 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2010.

[41] C. Leita and M. Cova. HARMUR: Storing and analyzing historic data on malicious domains. In *The BADGERS workshop, Building Analysis Datasets and Gathering Experience Returns for Security*, April 2011.

[42] C. Leita and M. Dacier. SGNET: a worldwide deployable framework to support the analysis of malware threat models. In *7th European Dependable Computing Conference (EDCC 2008)*, May 2008.

[43] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *21st Annual Computer Security Applications Conference*, December 2005.

[44] J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991.

[45] C. Linn and S. Debray. Obfuscation of executable code to improve resistance to static disassembly. In *10th ACM conference on Computer and communications security*, pages 290–299. ACM New York, NY, USA, 2003.

[46] J. Mao and A. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *Neural Networks, IEEE Transactions on*, 7(1):16–29, Jan 1996.

[47] J. Marichal. Tolerant or intolerant character of interacting criteria in aggregation by the choquet integral. *European Journal of Operational Research*, 155(3):771–791, 2004.

[48] W. L. Martinez and A. R. Martinez. *Exploratory Data Analysis with MATLAB*. Chapman & Hall/CRC, 2004.

[49] MessageLabs. Messagelabs Intelligence: 2009 Annual Security Report. Available online at `http://www.messagelabs.com/intelligence.aspx`, Dec 2009.

[50] MessageLabs. Messagelabs Intelligence: 2010 Annual Security Report. Available online at `http://www.messagelabs.com/intelligence.aspx`, Dec 2010.

[51] P. Miranda, M. Grabisch, and P. Gil. p-symmetric fuzzy measures. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(supplement):105–123, 2002.

[52] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *23rd Annual Computer Security Applications Conference (ACSAC)*, 2007.

[53] T. Murofushi and S. Soneda. Techniques for reading fuzzy measures (iii): Interaction index. In *Proceedings of the 9th Fuzzy Systems Symposium, Sapporo, Japan*, pages 693–696, 1993.

[54] Y. Narukawa and V. Torra. Fuzzy measure and probability distributions: Distorted probabilities. *IEEE T. Fuzzy Systems*, 13(5):617–629, 2005.

[55] PandaLabs. The Business of Rogueware. Analysis of a new style of online fraud. PandaLabs Reports, available online at `http://www.pandasecurity.com/homeusers/security-info/tools/reports/`, July 2009.

[56] PandaLabs. Profitability of rogue antimalware. PandaLabs Bulletins, available online at `http://www.pandasecurity.com/homeusers/security-info/tools/reports/`, 2009.

[57] M. Pavan. *A New Graph-Theoretic Approach to Clustering, with Applications to Computer Vision*. PhD thesis, Università di Bologna, 2004.

[58] 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.

[59] V. V. Raghavan and C. T. Yu. A comparison of the stability characteristics of some graph theoretic clustering methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-3(4):393–402, July 1981.

[60] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[61] V. Reding. Enhanced information security in software and services. What role for government, security providers and users? (speech). European Information Security Awareness Day, available online at `http://ec.europa.eu/commission_barroso/reding/docs/speeches/security_20070227.pdf`, February 2007.

[62] G.-C. Rota. On the foundations of combinatorial theory I. Theory of Möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2:340–368, 1964. (MR 30#4688).

[63] B. Schneier. Organized cybercrime. Available online at `http://www.schneier.com/blog/archives/2006/09/organized_cyber.html`, Sep 2006.

[64] L. Shapley. A value for n-person games. In H. Kuhn and A. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, Princeton, NJ, 1953.

[65] R. N. Shepard. Multidimensional scaling, tree fitting, and clustering. *Science*, 210:390–398, 1980.

[66] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[67] D. Smith. Allaple worm (ISC diary), `http://isc.sans.org/diary.html?storyid=2451`.

[68] J. Stewart. Top Spam Botnets Exposed. Malware Research, SecureWorks, published online at `http://www.secureworks.com/research/threats/topbotnets/`, April 2008.

[69] B. Stone-Gross, C. Kruegel, K. C. Almeroth, A. Moser, and E. Kirda. FIRE: FInding Rogue nEtworks. In *Twenty-Fifth Annual Computer Security Applications Conference (ACSAC 2009), Honolulu, Hawaii, 7-11 December 2009*, pages 231–240, 2009.

[70] M. Sugeno. *Theory of fuzzy integrals and its applications.* PhD thesis, Tokyo Institute of Technology, 1974.

[71] Symantec Corporation. Symantec Internet Security Threat Report. Available online at `http://www.symantec.com/business/theme.jsp?themeid=threatreport`.

[72] Symantec Corporation. Symantec Report on the Underground Economy. Available online at `http://www.symantec.com/business/theme.jsp?themeid=threatreport`, November 2008.

[73] Symantec Corporation. Symantec Report on Rogue Security Software. Available online at `http://www.symantec.com/business/theme.jsp?themeid=threatreport`, October 2009.

[74] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison-Wesley, 2005.

[75] O. Thonnard. *A multi-criteria clustering approach to support attack attribution in cyberspace.* PhD thesis, École Doctorale d'Informatique, Télécommunications et Électronique de Paris, March 2010.

[76] V. Torra. Weighted owa operators for synthesis of information. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 2, pages 966–971 vol.2, Sep 1996.

[77] V. Torra. The weighted OWA operator. *Int. Journal of Intelligent Systems*, 12(2):153–166, 1997.

[78] V. Torra. The WOWA operator and the interpolation function W*: Chen and otto's interpolation method revisited. *Fuzzy Sets Syst.*, 113(3):389–396, 2000.

[79] V. Torra and Y. Narukawa. *Modeling Decisions: Information Fusion and Aggregation Operators.* Springer, Berlin, 2007.

[80] J. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[81] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.

[82] VirusTotal. `www.virustotal.com`, 2007.

[83] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1101–1113, 1993.

[84] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision-making. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.

[85] T. Yetiser. Polymorphic viruses - implementation, detection, and protection , `http://vx.netlux.org/lib/ayt01.html`.

[86] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, 1971.