What's your major threat? On the differences between the network behavior of targeted and commodity malware

Enrico Mariconti, Jeremiah Onaolapo, Gordon Ross, and Gianluca Stringhini

University College London
e.mariconti@cs.ucl.ac.uk, j.onaolapo@cs.ucl.ac.uk,
g.ross@ucl.ac.uk, g.stringhini@ucl.ac.uk

***Abstract*—This work uses statistical classification techniques to learn about the different network behavior patterns demonstrated by *targeted malware* and *generic malware*. Targeted malware is a recent type of threat, involving bespoke software that has been created to target a specific victim. It is considered a more dangerous threat than generic malware, because a targeted attack can cause more serious damage to the victim. Our work aims to automatically distinguish between the network activity generated by the two types of malware, which then allows samples of malware to be classified as being either targeted or generic. For a network administrator, such knowledge can be important because it assists to understand which threats require particular attention. Because a network administrator usually manages more than an alarm simultaneously, the aim of the work is particularly relevant. We set up a sandbox and infected virtual machines with malware, recording all resulting malware activity on the network. Using the network packets produced by the malware samples, we extract features to classify their behavior. Before performing classification, we carefully analyze the features and the dataset to study all their details and gain a deeper understanding of the malware under study. Our use of statistical classifiers is shown to give excellent results in some cases, where we achieved an accuracy of almost 96% in distinguishing between the two types of malware. We can conclude that the network behaviors of the two types of malicious code are very different.**

I. Introduction

Malware is a threat that continually requires new defenses and mitigation strategies. The number of malware samples is currently increasing: in 2014, a new malware sample was released by attackers every 4 seconds and this trend continues to increase exponentially [14]. Different families of malware have different goals: some of them may try to damage the infected machine or other devices connected to it, while others may try to use the infected machine as part of an attack against a third entity. Some of the threats are also more critical than others for a network defender: for example, in a corporate environment a ransomware that is infecting and encrypting the hard disks of the infected machines is usually much more immediately dangerous than a malware that sends spam emails. These differences can be useful o those defenders that are managing the network alarms when it is needed to choose which threat to handle first.

In the past, attackers created malware to infect as many machines as possible, however more recently a new trend has emerged: the creation of software designed to target a specific victim by using a certain vulnerability in the victim's system. One of the most famous examples is Stuxnet [23], which exploited some zero-day vulnerabilities of the systems to destroy centrifuges in Iranian nuclear enrichment facilities. Allegedly this attack caused the postponement of the Iranian nuclear program. The Stuxnet example is clear evidence of how dangerous and costly a targeted attack can be for the victims. In [7], Chien and O'Gorman analyze a targeted attack on the chemistry industry sector and underline how frequent these attacks are, and which goals drive these attack campaigns.

Ideally, targeted attacks should be correctly detected and promptly countered by the network defense system, but systems are often under attack by multiple malware threats at the same time. Our goal is to analyze the network behavior of malware samples to identify differences that allow a system to recognize whether a specific attack is targeted or not. Our aim is to verify whether it is possible to distinguish targeted and generic malware using statistical analysis on their network behavior. If multiple malware samples are active on a network, but the system is able to recognize a targeted threat, it is possible to prioritize cleanup choices to limit the damages to the network.

We created a sandbox environment where malware samples produced network traffic that we statistically analyzed by extracting features, which were then used for classification. We carried out an analysis and selection of features through distribution studies, Cramer Von Mises tests and the use of Principal Component Analysis (PCA) following meticulous procedures to validate and justify the choices that can impact the classification results. By using different statistical classifiers and two mechanisms of k-fold cross validation we reached 96% accuracy in the classification of the network behaviors of the two malware types.

In summary, this paper makes the following contributions:

- We set up a sandbox that is able to run and monitor up to 1,200 malware samples a day to look for relevant malware network packets. The sandbox is able to work autonomously for an entire day respecting the safety of users by following the guidelines of previous studies such as [28] and [20].

- We systematically approach the statistical phase by studying the different features through distribution graphs, Cramer Von Mises tests, Histograms, and PCA before validating the findings of the statistical classifiers with k-fold cross validation. We perform a statistical classification among more than one hundred samples per type of malware reaching high values in the accuracy of the identification. The procedure we follow as statistical analysis justifies every experimental choice and can be used as a standard for future work.
- We suggest that our classification framework allows the two types of malware to be identified, and that this approach increases the capability of network defenders to respond to major threats when their systems are under different attacks.

II. Methodology

In this section we explain the procedures applied to all the different stages of the work. We start by explaining the configuration of the sandbox in detail and the operations that allowed us to infect Virtual Machines (VMs) by injecting malware samples in Sections II-A, II-B, and II-C. Section II-E explains the procedures that we applied to study the features (section II-D), and the dataset (section III-A), while section II-F shows which statistical classifiers, and which validation methods we used.

A. The experiment infrastructure

We set up a sandbox infrastructure (Figure 1) that is able to run 50 VMs at a time in the host-only environment created using VirtualBox on Linux. To increase the efficiency of our sandbox (reduce the number of evasion techniques a malware can successfully use to identify the virtual environment), we modified some basic parameters that malware samples often check to detect virtual environments. The main ones are the MAC address, the user name in the machine, and parameters such as the RAM and hard disk size. Each virtual machine downloads and executes a different malware sample from the webserver. A VM is able to request only one sample per test.

The webserver sends malware to the VMs after receiving a request from them. To avoid any double request (two malware samples operating from the same machine) and to track which malware produced specific network packets, the webserver records in a file the combination of VM's MAC address and malware's hash. Another service simulated in the host machine is the mailserver. The mailserver is part of the security measures taken into account to avoid risks coming from our infrastructure; the router redirected all the SMTP packets to the mailserver. A router manages all the connections. It uses IPTables and the VirtualBox options to allow the connection of the VMs to and from the Internet, the webserver, and the redirection to the mailserver. The router also manages the maximum bit rate of the connection for security restrictions.

Malware samples often try to recognize a virtual environment to avoid being studied, in case they detect the virtual environment, the sample would not try to do anything. Researchers have to modify settings to block the attempts a sample makes to evade. To maximize the efficiency of our environment, we used a tool called Pafish that indicates which can be the VM characteristics that will allow a malware sample to recognize it.

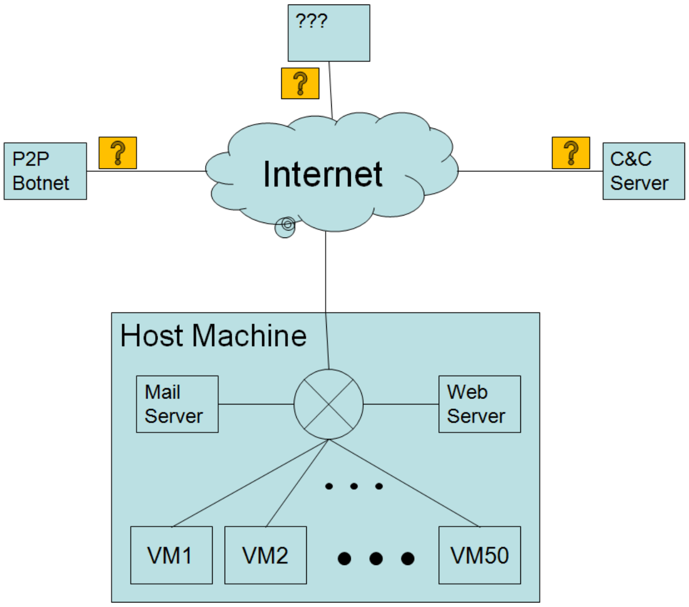

Fig. 1: Sandbox infrastructure: the host machine simulates a network of VMs, a webserver managing the malware distribution, a mailserver to redirect possible spam campaigns from the infected VMs, and a router that allows the connection among the different internal machines and with the Internet.

B. Security restrictions and ethical constraints

While the environment has some similarities with Botlab [20], the security restrictions applied to the environment follow the guidelines presented in [28]. The main restrictions that we operated are the limitations in time and rate, and the redirection of the e-mails. A sandbox that is connected to the Internet is allowed to communicate with Command and Control (C&C) servers or to operate attacks, therefore it is necessary to limit the damage it can do.

The rate limit is a good countermeasure to Denial of Service attacks: 50 VMs sharing a limited amount of bandwidth (1Mb per second) do not allow one of them to perpetrate an efficient DoS attack. The redirection of all the SMTP packets (ports 25, 443, 2525, 3535, 10024, 10025) prevents the effectiveness of all the possible malicious mail campaigns because instead of targeting the victims, all those network packets arrived to our mailserver. This restriction can also be a limitation in studying some of the samples because some of them might communicate with remote attackers using these ports. Another limitation is the time limit; it is due to the need of avoiding long and advanced attacks against people starting from the virtual environment; many attacks are coordinated from C&C servers, setting up instructions to let the infected machines operate for hours against the victims. Our VMs lifetime was 15 minutes.

These types of experiments do not involve sensitive user data and the application of the mentioned restrictions help to ensure that malware studies do not help cybercriminals by actively participating in malicious activity. An active and effective participation by our infected machines would have been reported to law enforcement agencies.

C. The experiment timeline

The experiment is divided in 24 tests of almost one hour each. Each test had a tcpdump instruction to record the network packets into a pcap file, and a different webserver managing different malware samples. The test starts by activating the tcpdump process, the webserver, and the mailserver services.

After that, the VMs are created and started. In each VM, a few scripts are executed: the IP address is immediately set up by looking at the assigned MAC address; after a certain timeout the VM sends a packet to the webserver to ask for the malware sample. Once the download is finished, the VM executes the sample.

The tests run 50 VMs at a time and last for 55 minutes; each experiment runs 24 tests therefore 1,200 malware samples run each day. All the pcap files collected during the tests have to be filtered from system packets and divided (by using the MAC address) in order to recognise which malware produced which traffic. To match malware samples and MAC addresses, we use the file produced by the webserver of each test with all the combinations malware hash-MAC address to assign the pcap file to the correct malware.

We take into consideration only the pcap files containing at least one answer from a destination; the presence of several UDP or ICMP packets is still accepted while the use of DNS packets without answers is not. From the pcap files we extract the features explained in the following section to analyze their efficiency with the samples' values and use the statistical classifiers for the identification of the samples.

D. Features

Previous work already used feature extraction and statistical classification for malware detection, for example [16]. In this work we tried to mix features at a different level using some commonly used features and some others that were not used in malware classification before, such as the Markov Chains probabilities of transition between network protocols.

We define as *flow* the set of packets having the same IP addresses, TCP/UDP ports (source and destination can be swapped in addresses and ports), and same protocol. Some of the features are flow oriented: they are calculated on each flow of the block (the entire pcap file related to the sample) and then converted to the block value (for instance a mean value will be the average among all the mean flow values). By converting the features from flow to block it has been possible to use block and flow features together.

The features can be divided into three families: network features, application features, and behavioral features. The network features look at parameters related to the transport and network levels while the application features extract parameters related to the highest level of the stack. The behavioral features look at parameters that are extracted from the mentioned levels, but they are trying to abstract some concepts from the recorded packets.

The network features are listed in Table I. These features are oriented to the flow, which means that it has been necessary to convert the list of values related to the flows to a "block value." These features are quite straight forward and they do not need particular explanations. The anomalies in values such as the TCP flags may be useful to identify attacks such as SYN floods or port scans that the malware can generate.

The application features (Table II) need more explanation. As the table shows, some of the features are oriented to the *flows* and some to the *block*. The flows are the sets of packets with the same source and destination IP addresses and ports, and the same protocol field that can be found in the IP packet header. A block is formed by all the packets produced by the sample during that session. The encryption features are used because malware often use an encrypted payload in the packets to mask their activity and communication. To understand if the payload is encrypted or not, we derived it from the entropy features as shown in [15] and applied in [12] and [27]. Olivain and Goubault-Larrecq explain in [15] how to calculate entropy using the frequencies of the possible bytes (in our case) that are in the packet. They also explain in detail how to use this information to understand if the packet is encrypted or not using a threshold and some adaptations of the entropy value due to the length of the packets. The "words in application" feature evaluates the presence of at least three letters by decoding the payload in ASCII characters; if the used protocol is SMTP, the words are compared with a blacklist in order to identify possible spam words that are considered in the "mail keywords" feature.

The last family of features are the behavioral features (Table III). Only two of them are flow oriented. One of the features we want to explain in detail is the GeoLocation one: we evaluated the destination IP of all the communication operated by the malware. To represent this information as a feature, we took into account 26 countries all over the world. Additionally, we used three clusters to include also countries that are not part of the 26 taken into account (other EU countries, other Asian countries, and Other), and the local address for a total of 30 possible destinations.

Markov Chains are a model used to evaluate the switch between two possible states of a system; every connection between two states is associated with the probability of transition between the two states. We used the Markov Chain probabilities on the network protocol-ports sequence to evaluate if the malware is repeating certain transitions between types of packets; we evaluated the transition between 18 specific protocols (among network level protocols such as ICMP and application ones such as DNS) and three clusters (other TCP, other UDP, and other network level) for a total of 21 protocols, and 441 probabilities (each one would be a feature). For example, we checked every HTTP packet and their following packet of the network traffic produced by a malware sample. We then filled a vector with the number of transitions between the HTTP protocol and each of the 21 possibilities we considered (also HTTP to HTTP is considered as a transition). To have the probabilities of transition from a state (the HTTP packet) to another (the packet that is following), we just need to normalize the transitions vector we created to make the sum of the components being one.

Features	**Oriented to**
Number of Packets	Flow
Total bytes per flow	Flow
First Length	Flow
Max Length	Flow
Min Length	Flow
Mean Length	Flow
Std Deviation Length	Flow
Delta Time	Flow
Std Deviation Delta	Flow
Flow Duration	Flow
Number of Syn flags	Flow
Number of Res flags	Flow

TABLE I: List of the network features and their orientation.

Features	**Oriented to**
Encryption	Flow
Percentage Encrypted	Flow
Mean Entropy	Flow
Max Entropy	Flow
Min Entropy	Flow
Std Entropy	Flow
Words in Application	Flow
Mail Keywords	Flow
Number of used Protocols	Block
More Used Protocol	Block
Percentage More Used	Block
Less Used Protocol	Block
Percentage Less Used	Block

TABLE II: List of the application features and their orientation.

Features	**Oriented to**
Number of C&C Servers	Block
IP Geolocation	Flow
Markov Chains	Block
Contacted IP IN	Block
Contacted IP OUT	Block
Contacted Domains	Block
DomainsIPs	Block
TCPUDP Packets	Block
TCPUDP Flows	Block
Percentage of Flows started by the sample	Block
Percentage IP Spoofing	Flow

TABLE III: List of the behavioral features and their orientation.

E. Analysis procedure

Before classifying the samples, it is necessary to exhaustively understand the characteristics of the dataset (section III-A) and of the features we extract from the network packets. We start by plotting density functions and histograms of the sample distributions for each feature to understand if any of the feature values were biased and to graphically observe if there are differences between the distributions of *targeted* and *commodity* samples. To validate this observation we run a Cramer Von Mises test [11] (Anderson version [1]) between the distributions of the two samples among all the features. As a result of this test, the p-value tells if the two populations are from different distributions or not. If the two populations are from different distributions, it is more likely to correctly identify samples of the two classes by using statistical classifiers.

The Markov chains features we used were 441; several of them were not relevant values because all the samples were producing 0 as the value for that feature; to maintain a good cost-efficiency ratio of the classifiers we apply a Principal Component Analysis (PCA [21]) to reduce the quantity of features to manage. The last analysis we operate before using the statistical classification is a hierarchical clustering [34]; the hierarchical clustering creates a tree graph that gives an idea if there could be a good separation of the classes we want to identify or if there will be a large number of wrong classifications.

F. Classification phase

Classification is operated by training more than one classifier on the features extracted for each sample. To validate all the attempts, we used two different k-fold cross validations (k=5 and k=10); the classifiers that we apply are random forests and two different kinds of K-nearest neighbors (K=1 and K=3). We used the whole dataset in a first attempt while we continued using a filtered set in which the exchanged network packets are at least 15 per each sample. We repeated the classification as in the first attempt and tried again by using as features only the PCA components derived from the Markov chains.

Random forest [5] combines several tree predictors such that each tree depends on the values given by a random vector, and the vectors must be identically distributed. All the trees give a classification prediction and a majority rule (or in some cases an average) among the predictions is applied to decide the final classification prediction.

K nearest neighbors [13] uses the concept of distance to predict which class the test sample is part of. It calculates which are the K (in our case one and three) nearest training samples in the features' space for each test sample. After it predicts the class of the test sample by looking at which class has the highest number of training samples in the K nearest ones. To avoid problems of two classes with the highest number of training samples in K, an odd number is used; in our case where there are only two classes, the conflict cannot happen with K=1 and K=3.

III. Analysis of the Data

A. Dataset

The dataset is composed of the features extracted from the execution of the targeted and commodity malware samples. For the generic malware samples we used executable files from *virusshare.com* archives, while for the targeted binaries a dataset collected at previous work [4]. Le Blonde et al. in [4] collected and analyzed targeted attacks against a human-rights Non-Governmental Organization representing an ethnic minority living in China. In particular, they analyzed the social engineering techniques, attack vectors, and malware attachments employed in malicious emails.

The malware samples that produced network traffic following the criteria mentioned in section II-C are 109 generic samples and 103 targeted ones. The filtered dataset excluded nine generic samples and 3 targeted resulting in 100 samples per class. We used the complete dataset in the first classification attempt and the second one in the further tries[1]. The filtered dataset does not contain malware samples that produce minimal traffic on the network. This may be due to a sample that is an old version and it is not finding any C&C server active anymore from the list it uses. Since such malware samples receive no orders from their C&C servers, their behavior is the same for generic and targeted samples.

B. Distribution of the samples

This section shows the differences and the similarities between the two populations of samples. These analysis are useful to select only the features that can be relevant and discard those that do not show differences among the samples of the two classes. The first analysis was made by checking the distributions of targeted and commodity samples (Figure 2). We used a logarithmic scale on the X axis to allow a better visualization and we observed how the two populations are distributed over the possible values of the number of packets that are exchanged from the malware. This graph shows how the two populations seem to be from different distributions, while other features such as the "contacted domains" in Figure 3 have populations that seem to derive from the same distribution.

The GeoLocation data needs more explanation: the histograms in Figure 4 and 5 are really different. The first one shows the geolocation of IP destinations applied to commodity malware; there are several contacted countries in a relevant way: the most contacted are US and other European (cluster involving most of Eastern Europe excluding Russia) IP addresses but relevant values come from China and other countries such as Germany and Russia. Another relevant value is given by the local (inside the Virtual Network) IP addresses that are contacted. On the other hand (Figure 5), Targeted malware samples mainly contact Chinese IP addresses. This, however, could be a bias of the targeted dataset: as explained in Section III-A the dataset derives from a previous work [4] that explicitly studied targeted malware created in China which attacked computers of employees of a human-rights Non-Governmental Organization representing a ethnic minority living in China. Therefore, it is normal that the malicious samples are trying to contact Chinese IP addresses. This bias in the dataset could dramatically affect the classification, therefore we excluded the geolocation features from the next steps of the work.

C. Statistical differences in distributions

To make the observations from the previous section more rigorous, we applied the two-sample Cramer Von Mises Test [11] to compare the distribution of each feature on the targeted and commodity malware data sets. For each feature, we computed the test statistic, and took a p-value of less than 0.01 as being evidence that the feature had a different distribution for targeted malware, and would hence be useful for classification purposes. P values greater than 0.01 were taken as evidence that the feature would not be useful, either because its distribution did not vary between the two classes of malware, or because the difference was not large enough to be detectable given the sample size that we had.

Some features which were found not to be useful were the mean length of the packets, the quantity of contacted IP addresses outside the local network and the second component of the PCA (section III-D) applied to the Markov chains probabilities. These features have hence been discarded except for the PCA component in the experiments of Section IV-C. However other features seemed to be important for determining differences between the packets produced from the types of malware; the p values of some tests, such as the one related to the delta mean feature (average of inter-arrival times between packets), have significant values (p value =0.001), but not as far from the threshold (0.01) as some others. The total bytes feature's p value is 5.6×10^{-8}, and the "first length of the packet" p value is 1.2×10^{-9}; the network level features are not the only ones with significant values, for example the "most used protocol" p value is 2.1×10^{-4} and this is an application feature.

An important consideration has to be given to the components derived from the Markov chains features by using the PCA: the second component has a p value that is not significant while the first component (the p value is 3.6×10^{-8}), the third one (p value= 7.4×10^{-4}), and the fourth one (p value= 1.1×10^{-4}) have significant values. We believe it is because PCA is an unsupervised features selection algorithm, therefore the second component is important to highlight the samples distribution but not for identifying the two classes among the components' space.

D. Principal Component Analysis

Principal Component Analysis (PCA [21]) is a dimension reduction technique that allows a large number of features to be summarized by a smaller number of features, projected into a lower dimensional space. We have already mentioned the use of PCA to select a few components, able to represent most of the information given by the 441 features, describing the Markov chains probabilities of transition from a certain packet to another. The quantity of variance (Figure 6) taken into

[1] The network dump files collected as part of these experiments are available at http://dx.doi.org/10.14324/000.ds.1500878.

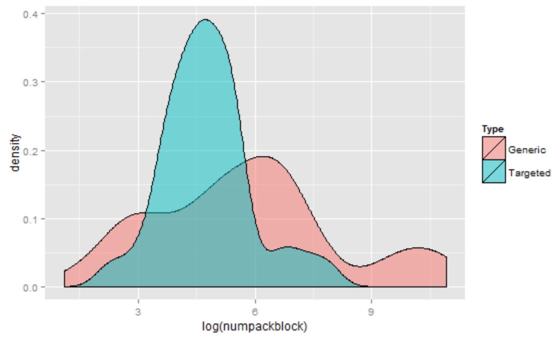

Fig. 2: The distribution of targeted and commodity malware with respect to the values of the “Number of Packets” feature. On the X axis we used a log scale; it is possible to observe that the two distributions seem to be different because while targeted samples are more concentrated around a mean value, the values of the commodity samples seem to be more distributed.

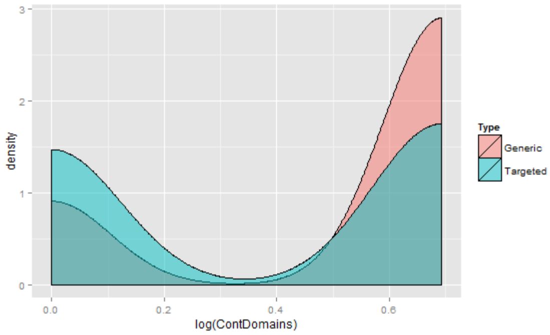

Fig. 3: The distribution of targeted and commodity malware with respect to the values of the “Number of Contacted Domains” feature. On the X axis we used a log scale; it is possible to observe that the two distributions seem to be similar because the slopes of the lines change in the same points of the two functions.

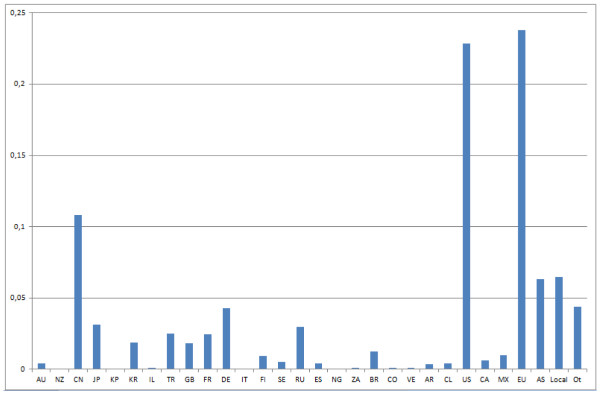

Fig. 4: The distribution over the countries of the IP addresses contacted by commodity malware. The two highest quantities are US and other European (cluster) IP addresses, but there still is a relevant quantity of IP addresses from China, other Asian countries (cluster) or IP addresses from the virtual network. German and Russian IP addresses are often contacted as well.

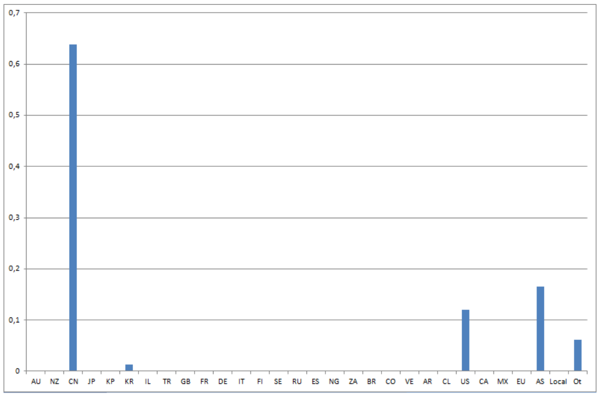

Fig. 5: The distribution over the countries of the IP addresses contacted by targeted malware. The quantity of Chinese IP addresses that are contacted is double the others. Other Asian cluster has a relevant quantity of contacted IP addresses and US has a minor quantity as well.

account by the first four components is already 80% and each feature contains at least 10% of the variance. For this reason, to avoid a dramatic increase of the classifiers’ computational costs we used only this set of components for the next steps. The p values shown at the end of the previous section highlight how the PCA components can be a significant help in the statistical classification. Markov Chains are a stateless tool (they have no memory), therefore the values are based only on the current state (current packet) and are not influenced by the previous ones. This characteristic of the tool will allow us to give a hint about a possible comparison between Markov chains and signature-based intrusion detection systems.

E. Hierarchical clustering

The last analysis step is hierarchical clustering. Hierarchical clustering is used to have a general idea if the classification may lead to a good separation between the two classes or not. Unfortunately it has not been possible to report the hierarchical clustering tree because of its very large dimensions. The height values of the graph indicate a dissimilarity degree, the higher the value where the links of two samples get together into the same root, the more likely the classification will be correct. The samples that are close to others are from a common route with a low height value. We drew the hierarchical clustering tree by using all the features (including the four PCA ones) and without the PCA features. The two trees are identical.

In Section III-C we commented on the p-values saying that three out of four PCA components have significant p values but that they are not as low as a few other features. If the two hierarchical clustering trees are identical, the algorithm evaluates the PCA components as much less important than other significant features. The trees show that targeted samples are mostly in big blocks in the middle of the graphs while most of the commodity samples stay on the sides of the graph; only

	Random Forests	1-nn	3-nn
Accuracy	0.956	0.780	0.784
Precision	0.911	0.762	0.732
Recall	0.905	0.792	0.851
F-Measure	0.906	0.775	0.785

TABLE IV: 5-fold cross validation results of the statistical classifiers. The statistical classification involves the entire dataset and all the features.

	Random Forests	1-nn	3-nn
Accuracy	0.951	0.789	0.795
Precision	0.901	0.773	0.739
Recall	0.901	0.786	0.843
F-Measure	0.897	0.771	0.782

TABLE V: 10-fold cross validation results of the statistical classifiers. The statistical classification involves the entire dataset and all the features.

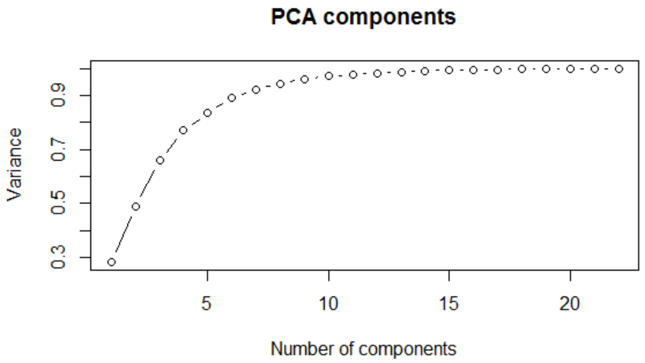

Fig. 6: The cumulative sum of PCA components' variance quantity. The first component already contains more than 30% of the total variance, four components take into account 80% of the information given by the 441 Markov chains features the PCA is applied to.

a few of them are in positions next to samples of the other class. The analysis of this section let us expect good results from the classification phase.

IV. Statistical evaluation

As explained in the methodology Section (II-F), we run the statistical classification by using Random Forests and two different K Nearest Neighbors under two k-fold cross validation algorithms. We present the following tables about the average values of the cross validation results in this section.

The evaluation parameters are accuracy, precision, recall and F-measure. Accuracy and F-measure are two parameters that try to give an overall situation: accuracy is a weighted mean of the correct identifications (true positives and true negatives) while F-measure is the harmonic mean between precision and recall. Precision shows the ratio between the correct positive identifications and the total number of identifications while recall considers how effective is the positive samples identification over the entire positive population. As we consider targeted malware as more dangerous, the positive samples are the targeted ones while the negative samples are the generic ones.

A. Complete dataset classification

The complete dataset is the set of all the samples that produced enough traffic to be considered; they are 109 generic samples and 103 targeted ones. Targeted samples are rare and it is not easy to have access to them; therefore there is no reason in using a large database of generic samples when the targeted one cannot be of the same dimensions. Short communications will be filtered in the next experiments reducing this set to a filtered dataset. Tables IV and V show the results of the statistical classifiers applied to the complete dataset. Using 5-fold cross validation, the random forests classifier reaches really high values: an accuracy of 95.6% means that the system can be trusted even if 90.5% recall indicates that some of the targeted samples are still missed. While it is not possible to clearly state a difference between the two Nearest Neighbors classifiers' results, we can say that they are not effective as the random forests. The values in Tables IV and V show an overall efficiency that is similar between the two K-nn classifiers because accuracy is 78% and 78.4% and F-measure is 77.5% and 78.5%. They reached it through different paths: considering the values of precision (76.2% and 73.2%) and recall (79.2% and 85.1%), it is possible to say that 3-nn tends to recognize the targeted samples better (higher recall) while 1-nn tends to have less false positives. This diversity is probably due to the difference in distributions highlighted in Section III-B: targeted samples are more concentrated in one area than commodity samples. When it comes to classification, if the test sample is in the area where targeted samples are concentrated, it will be more possible to find at least two targeted training samples and label the test sample as targeted more than as generic. The 10-fold cross validation results are similar to the 5-fold ones, the small difference is not relevant.

B. Filtered dataset classification

We repeated the statistical classification by using the filtered dataset. As already mentioned, this dataset does not contain the samples that produce irrelevant quantities of traffic; we applied this filter because a few packets do not describe the malware activity. The new dataset has nine commodity samples and three targeted samples less than the complete dataset. The results are slightly different: random forests slightly increases its efficiency, while the two K-nn fail the correct identification of more samples than in the previous classification. With respect to the previous dataset, the differences are not relevant, but, since the databases did not change a lot, we did not expect really different results. Random forests increases its recall by more than 3%, suggesting that those samples we filtered were part of the wrong classifications. Both K-nn algorithms had a decrease of precision, especially 1-nn. The filter affected more

	Random Forests	1-nn	3-nn
Accuracy	0.964	0.761	0.769
Precision	0.893	0.749	0.728
Recall	0.944	0.781	0.852
F-Measure	0.914	0.762	0.782

TABLE VI: 5-fold cross validation results of the statistical classifiers. The statistical classification involves the filtered dataset and all the features.

	Random Forests	1-nn	3-nn
Accuracy	0.962	0.787	0.789
Precision	0.905	0.777	0.734
Recall	0.933	0.782	0.859
F-Measure	0.912	0.768	0.785

TABLE VII: 10-fold cross validation results of the statistical classifiers. The statistical classification involves the filtered dataset and all the features.

	Random Forests	1-nn	3-nn
Accuracy	0.878	0.831	0.806
Precision	0.798	0.783	0.774
Recall	0.907	0.907	0.8572
F-Measure	0.846	0.838	0.804

TABLE VIII: 5-fold cross validation results of the statistical classifiers.The statistical classification involves the filtered dataset and only the PCA components derived from the Markov chains probabilities.

	Random Forests	1-nn	3-nn
Accuracy	0.887	0.782	0.837
Precision	0.810	0.770	0.794
Recall	0.912	0.790	0.921
F-Measure	0.848	0.772	0.839

TABLE IX: 10-fold cross validation results of the statistical classifiers. The statistical classification involving filtered dataset and only the PCA components derived from the Markov chains probabilities.

the number of generic samples than the targeted one arriving to the same number for the two types of samples. That probably is the reason why there is a disadvantage for negative samples classification with respect with the previous tests.

C. *PCA-only classification*

The last experiment with statistical classifiers (Tables VIII and IX) was conducted using a limited number of features: we used only the four PCA components derived from the Markov chains probabilities. Since we are using features related to the sequences of used packets, it is a kind of defense that could be considered similar to signature-based systems; these systems check certain sequences of packets to recognize malicious activity. The random forests classifier still shows a good accuracy (87.8% and 88.7%) but, as expected, it does not give as high values as in the previous cases: the use of only 4 features, without taking into account the others, is not the optimal case for random forests. It is important to notice that it is the precision that decreases dramatically while recall is still about 90%, meaning that this set of features gives much more false positives than the complete set.

K-nn increases its accuracy by using the restricted set of features on the filtered dataset. The higher value is due to the basic concept K-nn uses: the distance between points; when a high number of features is used, we are working on a big number of dimensions in the feature space and the concept of distance is not straightforward because of scaling issues among variables. K-nn is more efficient in this classification phase because there are only four features (four dimensions); the results of 1-nn are comparable with random forests' ones even if the accuracy is still a bit lower than random forests' accuracy.

It is interesting to notice that this is the only case where 1-nn results change in a relevant way between the two cross validation procedures; this is due to a great difference in the recall value (11.7%). The decrease in targeted samples identification can be explained only with the fact that the PCA transformation may have distributed some commodity samples in the region where more targeted samples were, thus the presence of a few more training samples in the 10 fold cross validation strongly modified the efficiency of 1-nn.

V. Discussion

A. *Results*

The results show that there is the opportunity of distinguishing the network packets generated by targeted malware from those generated by commodity malware using statistical classifiers. An accuracy of over 95% shows the almost total separation between the samples of the two classes. Many insights of the analysis section have been confirmed by the evaluation section and we believe that this work can be a starting point in the automatic evaluation of priorities in malware mitigation. The continuous update of the dataset and the separation into malware families may be important factors to achieve higher accuracy. Network administrators may apply this work to decide which alarm has to be taken into account with higher priority; the different opportunities in tuning training set and statistical classifiers can lead to customized options in order to enhance the defenses basing the system on each single network where it is used.

Analyzing the steps explained in Section III, it is possible to notice that the more interesting and useful features are oriented to the flow, except for the Markov chains probabilities. This means that a more detailed study on the flow analysis can provide even more accurate results in the separation of the types of malware than this work (that already reached important values).

In the evaluation section we show experiments taking into account only the Markov Chains information as features. We explicitly refer to the similarity between the mechanism of this memoryless tool and the signature based approach; the accuracy reached with these features (Tables VIII and IX) is high considered the small amount of features and, as a consequence, of computational cost. This result places important considerations on how effective these tools can be when tuned in an accurate way, and if they can be more robust than similar defensive mechanisms. It is important to observe that, even if the p values of the Cramer Von Mises test were not the lowest ones, Random Forests reaches an accuracy of almost 90% using these features alone. These results are less than the previous ones, but still relevant.

B. Limitations

This work applied strict experimental and analysis procedures to avoid any possible biased or unreliable result. For this reason, the limitations are mostly due to restrictions that did not allow to get higher accuracy but those limitations did not affect the validity of the work.

One of the limitations is due to the restrictions that the sandbox had for the safety of Internet users: the redirection of the SMTP packets did not allow us to completely study malware samples that were using that protocol. Other restrictions might have affected some of the features limiting their efficiency but not biasing the results; for instance the VM life limit may not allow us to observe some particular malware behavior, but it would limit the accuracy of the system and not the reliability.

The other main limitation was the bias in the geolocation features; the targeted dataset is based on malware samples created for only one targeted campaign, therefore there was a strong bias in which IP addresses were contacted. It was necessary to ignore the geolocation features when we used the classifiers, turning down another tool able to increase the accuracy of the system.

VI. Related Work

Cybercriminals created malware samples since the 80's, the first were viruses [10], but the evolution brought more advanced and powerful attacks such as those operated by botnets [22]. While the first attackers were penetrating and infecting systems for fun or to show their skills, malware became a profitable service in the last decade. Botnets have been used to send spam [32] or perform powerful distributed denial of service (DDoS) attacks and through the years countermeasures and mitigation methods have been found on single issues such as, for instance, [6] for DDoS or [17] for bot infection detection or [31] where researchers explained how they took down the Torpig botnet. Botnets activity mitigation efforts are good examples for defenses created to detect and block malware samples. Modern ways to study botnets go through active probing, as explained in [9], infiltrating a machine into the botnet network [8] [30]; another attempt done in previous work is a blacklisting activity explained in [33].

Before the implementation of mitigation techniques like the cited ones, the aforementioned attacks have been successful in several cases, just by trying to do as much damage as possible to the unlucky companies or private users that have been infected.

The difference that distinguished the successive generation of malware has been the target: instead of attacking randomly anybody, the target has become a specific victim, attacked by using vulnerabilities of the victim's system for the profit of the victim's adversaries. These are the targeted malware [4]. If a targeted malware is attacking a certain target (company, organization, or country), it has to intervene rapidly in order to limit the damages; being able to distinguish the type of malware that is attacking represents an asset useful to increase the efficiency of the security systems. The delicate tasks the targeted malware has, request it to hide in the processes limiting its acts, to operate without being noticed almost until the achievement of its goal. This requirement is translated in few light operations using the processors of the infected computers, while only the essential packets on the Internet are exchanged to hide them among the normal network user activity.

Another key aspect to investigate malware is understanding their behavior. This work focuses on the network behavior of malware, when they have to communicate with a computer outside the network like C&C communications, and the reason of these exchanges. Previous work that focused on other topics of malware network behavior have already been done about honeypots [3] or the correlation between malicious activity and previous instructions from a C&C server [16] [35]. A kind of communication is represented by the packets needed for a Denial of Service (DoS) attack; it is important to underline that these packets are different from the ones exchanged between a spyware and its master about the sensitive data on the infected machine. There is a long list of possible malware behaviors and they correspond to different network packets on the Internet; as previously explained, the presence of a targeted malware inside the network is a high threat, and being able to distinguish which malware is operating on the network can be particularly important. Malware identification on a company network can be done by using different security tools with different tasks: a peripheral firewall can already filter those packets coming from a blacklisted IP address, whereas antivirus software is able to detect in several ways suspicious code, and Intrusion Detection Systems (IDSs) identify patterns or anomalies in the traffic of packets that pass from their node.

A detailed overview of the different detection techniques, characteristics, and methods of IDSs has been given by [29], and the focus of this work is in those IDSs that practice misuse detection on the network. As already mentioned, machine learning has been already used in other security studies as [16] and [17] on botnets or [2] [18] [19] [26] on different DNS issues. In two old studies where it was used on IDSs [24] and [25], but the application of statistical classifiers to such a specific and new problem is currently missed. In our work we did not limit the use of statistics to basic analysis and machine learning applications as it is usually done, but we added several steps to understand the characteristics of each feature and their importance during the experiments.

VII. Conclusions

This study analyzed the opportunity of identifying network traffic produced by targeted malware, distinguishing it from the one produced by generic malware. This analysis gives an opportunity to perform prioritization choices to enhance the defenses against the major threats to networks.

We set up a sandbox to collect malware network traffic from the samples; from the recorded packets we extracted several features and, after a meticulous analysis, we applied statistical classifiers to different sets of samples and features. During the analysis we tested the differences between the distributions of the two populations by using graphs and Cramer Von Mises tests. After further analysis by using hierarchical clustering, we applied statistical classifiers to the extracted features reaching 96% accuracy with the Random Forests classifier. The statistical tests indicate the possibility to separate network packets generated by targeted malware samples from generic samples ones.

We executed other tests involving the four PCA features; the tests show a good accuracy from this restricted set of features even if the analysis phase indicated other features as the most effective. The results of this section open interesting possibilities in the identification of malicious activities through modeling the sequences of actions.

Another important aspect of this work is the applied methodology: to study and validate all the aspects of the dataset, the features, and the classifier results we used a careful procedure that can be applied to several problems where complicated distributions of samples have to be evaluated and classified.

This work opens the scenario of prioritization choices between types of malware to evaluate other aspects of this area, for instance in incident response.

VIII. Acknowledgments

We wish to thank the anonymous reviewers for their comments. This work was funded by the EPSRC under grant number N008448. Enrico Mariconti was funded by the EPSRC under grant 1490017, while Jeremiah Onaolapo was supported by the Petroleum Technology Development Fund (PTDF), Nigeria.

References

[1] Anderson, T., and Darling, D. Asymptotic theory of certain 'goodness of fit' criteria based on stochastic processes. *Annals of Mathematical Statistics* (1952).

[2] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., and Feamster, N. Building a Dynamic Reputation System for DNS. In *USENIX Security Symposium* (2010).

[3] Balzarotti, D., Cova, M., Karlberger, C., Kruegel, C., Engin, K., and Vigna, G. Efficient detection of split personalities in malware. In *Symposium on Network and Distributed System Security (NDSS)* (2010).

[4] Blond, S. L., Uritesc, A., Gilbert, C., Chua, Z., Saxena, P., and Kirda, E. A look at targeted attacks through the lense of an NGO. In *USENIX Security Symposium* (2014).

[5] Breiman, L. Random forests. *Machine Learning 45* (2001).

[6] Buscher, A., and Holz, T. Tracking DDoS attacks: Insights into the business of disrupting the web. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2012).

[7] Chien, Eric and O'Gorman, Gavin. The Nitro Attacks. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_nitro_attacks.pdf, 2011.

[8] Cho, C., Caballero, J., Grier, C., Paxson, V., and Song, D. Insights from the inside: A view of botnet management from infiltration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2010).

[9] Cho, C. Y., Babich, D., and Song, D. Inference and Analysis of Formal Models of Botnet Command and Control Protocols. In *ACM Conference on Computer and Communications Security (CCS)* (2010).

[10] Cohen, F. Computer viruses: theory and experiments. *Computers & security 6* (1987), 22–35.

[11] Cramèr, H. On the composition of elementary errors. *Skandinavisk Aktuarie- tidskrift* (1928).

[12] Dorfinger, P., Panholzer, G., Trammell, G., and Pepe, T. Entropy-based traffic filtering to support real-time Skype detection. In *International Wireless Communications and Mobile Computing Conference* (2010).

[13] Fix, E., and Jr, J. H. Discriminatory analysis, nonparametric discrimination. *USAF School of Aviation Medivine, Randolph Field 4* (1951).

[14] GData Labs. Malware Report. https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/GData_PCMWR_H2_2014_EN_v1.pdf, 2014.

[15] Goubault-Larrecq, Jean and Olivain, Julien. Detecting subverted cryptographic protocols by entropy checking. Laboratoire Specification et Verification. http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2006-13.pdf, 2006.

[16] Gu, G., Perdisci, R., Zhang, J., and Lee, W. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In *USENIX Security Symposium* (2008).

[17] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., and Lee, W. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *USENIX Security Symposium* (2007).

[18] Holz, T., Gorecki, C., Rieck, K., and Freiling, F. Measuring and detecting fast-flux service networks. In *Symposium on Network and Distributed System Security (NDSS)* (2008).

[19] Hu, X., Knysz, M., and Shin, K. Rb-seeker: Auto-detection of redirection botnets. In *Symposium on Network and Distributed System Security (NDSS)* (2009).

[20] John, J. P., Moshchuk, A., Gribble, S. D., and Krishnamurthy, A. Studying Spamming Botnets Using Botlab. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2009).

[21] Jolliffe, I. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2002.

[22] Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., and Savage, S. Spamcraft: An inside look at spam campaign orchestration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2009).

[23] Langner, Ralph. To Kill a Centrifuge. http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf, 2013.

[24] Lee, W., and Stolfo, S. J. Data mining approaches for intrusion detection. In *USENIX Security Symposium* (1998).

[25] Lee, W., Stolfo, S. J., and Mok, K. W. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy* (1999).

[26] Passerini, E., Paleari, R., Martignoni, L., and Bruschi, D. Fluxor: detecting and monitoring fast-flux service networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)* (2008).

[27] Rossow, C., and Dietrich, C. Provex: Detecting botnets with encrypted command and control channels. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)* (2013).

[28] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., and van Steen, M. Prudent practices for designing malware experiments: Status quo and outlook. In *IEEE Symposium on Security and Privacy* (2012).

[29] Sabahi, F., and Movaghar, A. Intrusion detection: A survey. In *International Conference on Systems and Networks Communications* (2008).

[30] Stock, B., Gobel, J., Engelberth, M., Freiling, F., and Holz, T. Walowdac - Analysis of a Peer-to-Peer Botnet. In *European Conference on Computer Network Defense* (2009).

[31] Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., and Vigna, G. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *ACM Conference on Computer and Communications Security (CCS)* (2009).

[32] Stone-Gross, B., Holz, T., Stringhini, G., and Vigna, G. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2011).

[33] Stone-Gross, B., Kruegel, C., Almeroth, K., Moser, A., and Kirda, E. FIRE: FInding Rogue nEtworks. In *Annual Computer Security Applications Conference (ACSAC)* (2009).

[34] Ward, J. H. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association 58* (1963), 236–244.

[35] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., and Kirda, E. Automatically Generating Models for Botnet Detection. In *European Symposium on Research in Computer Security (ESORICS)* (2009).