Expert knowledge and data analysis for detecting advanced persistent threats

Abstract: Critical Infrastructures in public administration would be compromised by Advanced Persistent Threats (APT) which today constitute one of the most sophisticated ways of stealing information. This paper presents an effective, learning based tool that uses inductive techniques to analyze the information provided by firewall log files in an IT infrastructure, and detect suspicious activity in order to mark it as a potential APT. The experiments have been accomplished mixing real and synthetic data traffic to represent different proportions of normal and anomalous activity.

Keywords: Advanced persistent threat, Cybersecurity, Data mining, Expert knowledge

MSC: 68T30, 68T05

1 Introduction

Most of the daily activity in modern society involves the use of electronic devices and communication networks through cyberspace. In this context, both hardware and software may suffer attacks and/or threats that compromise activities ranging from provision of public administration services to those tasks that are related to economic aspects, access to information, education, company activities, etc.

As the complexity of attacks is in constant evolution, as well as the technologies and tools they use, today we have increasingly sophisticated threats that exploit vulnerabilities in the code, the protocols, the computer systems, the communication networks, etc.

Advanced Persistent Threats (APT) constitute one of the most serious cybersecurity threats. APTs would be described as malicious or anomalous behaviors that overcome security blocks with the main aims of cyber-spying, stealing and handling sensitive private information belonging to individuals, private corporations and public corporations or even government. These threats are selective (targeted), and their effects quite harmful (see [1] or [2], among others). Moreover, since APTs usually attack sensitive infrastructures – often with law protected data --, access to such log records is in general not granted.

When all the technological tools and security measures that protect an infrastructure fail, the APT reaches its goal of compromising and stealing the valuable information of an active or a service.
Some authors have developed tools that use theoretical models or intelligent systems to detect APTs [3–6]. These systems are based on the analysis of DNS records or hosts, with the common drawback that the models depend on both the datasets and the complexity of the systems.

As the attackers are looking to remain persistent once inside the system, log analysis and identification of behavioral anomalies are usually the key for protecting an infrastructure [5]. This work proposes an intelligent system that generates predictive learning based models of behavior that help us detect anomalous activity that might be classified as APT. The system is based on supervised learning applied to the logs provided by the firewall that filters the infrastructure inbound/outbound traffic. These logs include the registers obtained from one actual APT that reached its goal of remaining persistent for some weeks before it was detected and removed. Since the system is based on real traffic data at a real infrastructure, it can be considered as productive, effective and realistic.

This paper is organized as follows: Section 2 introduces some of the previous related work; in Section 3, we describe the proposed methodology, including collection of information, data processing and statistical analysis; Section 4 shows the experimental results; Section 5 discusses the relevance of the results; then, Section 6 provides the conclusions and future work; and last, two appendices show the algorithm that normalizes the log registers, as well as reports of running our model over real datasets.

2 Background

Several theoretical frameworks provide solutions or predict APT attacks. All of them share a common methodology to design a model, and include an intermediate stage before accomplishing the classification. There is a remarkable exception [3] that does not need any intermediate stage; we first review this system.

System [3] for early detection of APT uses the approximate inference algorithm Belief propagation and data mining techniques. It uses datasets from a corporate private network, and models the communications between devices internal to the network internal (hosts) and between hosts and external domains with the help of a bipartite graph whose edges link those hosts and domains that are connected at least once during the observation period. Applying dimensionality reduction techniques, the system generates a list of suspect domains.

Several models include intermediate stages:

- Attack Pyramid [6] is a model inspired by the attack tree concept proposed in [7] and [8]. Attack Pyramid uses the shape of a pyramid as a model of an APT attack whose apex represents the objective of the APT, while its faces represent the paths and barriers to overcome the threat. The authors define a context of attack in a private corporate network and generate an alert to the system so that it can decide whether there are any hazards inside the network.

- System [5] provides a framework that seeks to generate a particular model depending on the scenario, and using dataset obtained by method proposed in [9]. Two datasets are stored on a stage with no attacks in order to develop the model, and a third set with artificial anomalies to train and evaluate its efficiency. This approach concludes that the model is effective combining datasets generated inside an organization without prior knowledge of their structures. Classification is an intermediate stage, but the ultimate goal is to have a model adapted to each infrastructure and automatically update it based on the inputs.

- System [10] combines automatic systems created from big data and machine learning as well as expert knowledge. Besides, there is continuous feedback so that the model is updated based on new anomalous cases. The authors use datasets generated by web servers to detect attacks on the Internet services and their firewalls datasets for the analysis of possible data exfiltration. The automatic part combines three models that work in parallel (Matrix Decomposition based on the research by [11], Replicator Neural Networks and Density-based outlier analysis). This proposal concludes that the inclusion of the human knowledge achieves a detection improvement of 3.41, while the number of false positives becomes decreased by a factor of 5.

3 Methodology

This section describes the design of one model to identify possible APT attacks within network traffic.
The design of the model relies on a deep analysis of APT behavior. Hence, we have analyzed this type of attacks, as well as the different techniques they use to steal information: IP Address, Domain Lists, Peer to Peer, DomainGenerationAlgorithm or Fast Flux Domain, etc. [12]. We have also studied those security systems that could succumb to an APT, as SEH, SafeSEH, SEHOP, Stack Cookies, ASLR, PIE or NX (see [13–19]).

APTs occur rarely, hence the proportion of their log registers is very small, what means that our datasets involve imbalanced distributions. Several works propose the use of synthetic data to improve datasets which suffer from imbalanced class distributions, including non-heuristic methods such as random undersampling or oversampling [20], and those that use some kind of interpolation for oversampling the training sets [21, 22]. In our case, the imbalanced datasets were improved by random oversampling so that the experiments used actual log files (logs) generated by the firewall of an actual operating infrastructure in combination with synthetic registers generated through expert knowledge. In particular, we have created and analyzed 9 samples ($S_i, i = 1, \ldots, 9$) with different proportions of correct and anomalous behaviors.

Machine learning tools acquire knowledge from experience and are useful for the semiautomatic construction of programs in those cases when experience in a given resolution of tasks is available (see [23]).

In this work, we have measured the accuracy of the proposed model with several samples using bayesian techniques, decision trees and artificial neural networks. Decision trees showed better fitness. Then, we have performed validation tests over all the samples and selected some variables to be assessed: accuracy of the model created with the decision tree, improvement over the trivial model, sensitivity to harmful behavior, resistance accuracy, resistance improvement over trivial model and resistance sensitivity to harmful behavior. In order to choose the best possible proportion of activity logs, we have developed descriptive analysis over each sample with the values of the variables described above (boxplots and arithmetic means). The sample with the highest mean would point to the most adequate model. Figure 1 shows the structure of the whole process.

Once analysis is finished, the final system runs with the best sample and is able to alert of log registers that might be related with APTs.

Regarding the technology and the software, we have used Python 3.5 and KNIME 3.1.2 to develop the process described in Figure 1, that relies on the log files obtained from the real, in operation infrastructure.

### 3.1 Data acquisition

The dataset we have used was composed of log registers provided by the firewall of a real, geographically dispersed, operating infrastructure involving more than thirty buildings interconnected by a fiber optic ring and centralized at a datacenter (Figure 2).

The above mentioned infrastructure consists of more than 500 networked computers, several broadcast domains including DMZ, VPN using IPSec and SSL, more than a hundred tablets and cell phones, more than five hundred VoIP phones, three data centers (one primary and two secondary ones), cluster technology with blade and virtualized servers, more than thirty servers –both virtualized and physical–, two network security appliance high availability firewalls, one proxy-cache server, several NAS and SAN disk arrays, a management core network, intelligent management system cabling, more than twenty uninterruptible power supply units (UPS), fire detection systems, more than thirty switches distributed for voice/data communications, more than thirty communications racks, Oracle 11g Database and sole output channel Internet connection.

The infrastructure is frequently attacked by different external vectors that should be detected by security elements such as antivirus software, IDS, IPS, SIEM, etc. Whenever this defense system detects that the assets are being targeted for an attack, it generates an accurate, fast alert. If the attacker is an internal user, and the propagation of the attack is sneaky, then the threat overcomes the mentioned protection systems. Such suspect behavior can be detected by human experts by deeply analysis of the firewall log registers.
Fig. 1. Structure of the process to generate an APT detector

Dataset

Sample Real Dataset $S$

Raw Logs

Normalized logs

Refined logs

Synthetic Logs

Instances

Sample Datasets $S_i$

Classifying techniques

Naïve-Bayes Classifiers

Decision Trees

Neural Networks

Accuracy

Validation test

Accuracy Resistance

Improvement over trivial model

Boxplot

Improvement over trivial model

Sensitivity to harmful logs

Sensitivity to harmful logs

Valuation of each sample, $S_i$ (quartiles)

Best Sample dataset $S_i$

Run system $\rightarrow$ Alerts of APT
3.2 Dataset description

The analysis of the infrastructure data traffic shows that each log register (log) contains information about one specific event that was produced within the structure. The inbound/outbound traffic generates our log dataset, whose main features are the following:

1. **Volume**: The daily log files average size is 5.46 GB, which means an average of 7.445,736 registers/day. Moreover most of the traffic is external (see Figure 3) and network protection services (Firewall) generate logs on the order of petabytes in size.
2. **Speed (log size/hour)**: Every hour, the system generates logs of 233.2 MB (310,239 registers) on average (see Figure 3).
3. **Variety**: The registers (lines) in every log file include information of different nature – about events, security or traffic related. Anyway, we will only consider those registers concerning the firewall inbound/outbound traffic, as the firewall itself handles the other registers in order to automatically generate alerts in case of attack.

For this work, we have used a sample, $S$, of the dataset, that contains the log files of one whole month (310,239 logs per hour).

3.3 Data pre-processing

Dataset sample $S$ was chosen in such a particular time window that allowed us to classify all the logs in $S$. In fact, all the logs were tagged as **correct behavior** (no risk of APT) and, therefore, in order to complete the model, it was necessary to add synthetic logs representing **anomalous behaviors** (potential APT). Note that this synthetization is an experimental tool usual in absence of data [24].
Hence, we have created nine different samples, $S_i (i = 1, \ldots , 9)$, with different correct/suspect behavior ratios (green/red behavior).

The pre-processing stage involves normalizing the initial real traffic raw logs, refining them by quantization of the information, and obtaining instances suitable for machine learning algorithms. Similarly, synthetic logs would be transformed into synthetic instances. Last, the datasets that would feed the learning algorithms are combinations of real and synthetic instances.

### 3.3.1 Real logs: from raw logs to normalized logs

The real logs – also called natural logs – are those generated as raw logs by our firewall to provide information about 40 items. These logs are normalized by removing those fields that human agents experience says that are not needed. The normalization algorithm was coded using Python structured programming (the interested reader can see the pseudocode in Appendix I).

As a result, normalized logs are state vectors of 12 elements – fields – that contain the non-redundant information, i.e. the discriminant fields in the raw logs that best characterize them under the security approach of identifying APT suspicious behaviors (see Table 1).

**Table 1. Description of the components of the normalized log vector**

| Field                  | Name       | Type     | Length | Description                                                      |
|------------------------|------------|----------|--------|------------------------------------------------------------------|
| Date                   | Date       | Numerical| 8      | Connection date                                                  |
| Time                   | Timestamp  | Numerical| 6      | Connection time                                                  |
| Attack Name            | Attackname | String   | 50     | Name of the attack (if it exists)                                |
| Destination Country    | Dstcountry | String   | 50     | Targeted country                                                 |
| Destination IP         | Dstip      | Numerical| 12     | Targeted IP                                                     |
| Duration               | Duration   | Numerical| 5      | Connection duration (in seconds)                                 |
| Hostname               | Hostname   | String   | 50     | Hostname that performs the connection                            |
| Protocol               | Protocolo  | Numerical| 1      | Protocol used                                                   |
| Received bytes         | Rcvdbyte   | Numerical| 12     | Amount of received bytes                                         |
| Sent bytes             | Sentbyte   | Numerical| 12     | Amount of sent bytes                                            |
| Source IP              | Srcip      | Numerical| 5      | Attacker IP                                                     |
| Status                 | Status     | String   | 30     | Status of the established (or attempted) connection              |

### 3.3.2 Real Logs: From normalized logs to refined logs

Normalized logs include quantitative information whose variability and complexity must be reduced before applying learning algorithms. Hence, refined logs are the result of using expert knowledge based on simple statistical (means, ranges, frequencies, etc.) and trend analysis in order to quantize the information in the fields of the normalized logs.

The quantization of dates and times involves distinguishing between working and non-working days, on the one hand; and mornings, afternoons/evenings and nights, on the other (Equations 1 and 2).

\[
\text{Date} = \begin{cases} 
1 & \text{working day} \\
2 & \text{non-working day} 
\end{cases} \quad (1)
\]

\[
\text{Time} = \begin{cases} 
1 & \text{morning} \\
2 & \text{evening/afternoon} \\
3 & \text{night}
\end{cases} \quad (2)
\]
The other variables were recoded using their arithmetic means (Equation 3).

\[
X = \begin{cases} 
1 & \text{if } x \leq \bar{x} \\
2 & \text{if } \bar{x} < x \leq 1.20\bar{x} \\
3 & \text{if } x > 1.20\bar{x}
\end{cases}
\]

(3)

where \(x\) and \(X\) are the values of the same variable before and after the quantization, respectively.

3.3.3 Real logs: from refined logs to real instances

Once logs are quantized into refined logs, they have to be converted into instances, \textit{i.e.} input vectors that can feed machine learning algorithms.

We have used instances that contain 9 states related with the source IP. Eight of these states are extracted from the information in the refined logs: date, time, duration, received bytes, sent bytes, number of connections (per millisecond), number of denies (per millisecond) and average data traffic. There is an additional variable that tags the behavior associated to one instance as \textit{red} or \textit{green}. On the one hand, \textit{red} behavior would mean that the corresponding log might be considered as anomalous and, therefore, could be related to a potential APT; while \textit{green}, on the other hand, would label those activities that should be considered as harmless. In our case, all the instances coming from real traffic were classified as harmless, \textit{i.e.} \textit{green}.

Hence, the structure of the instances or final vectors is as follows:

\[I = (\text{date}, \text{time}, \text{duration}, \text{received bytes, sent bytes, milliseconds, denies, mean traffic, group behavior})\]

Software that converts normalized logs into real instances was Python coded.

3.3.4 From synthetic logs to synthetic instances

As mentioned above, the frequency of anomalous behaviors was low in our sample \(S\). Thus it was necessary to create synthetic logs to represent APT related activity. These synthetic logs improve the model, allowing fast and efficient simulations of multiple scenarios. They have been massively introduced in the form of instance \(I\), based on expert knowledge focused in 15 types of information that firewall logs provide (see Table 2).

| ID | Information                                                                 |
|----|-----------------------------------------------------------------------------|
| 1  | How long an IP is connected (in-out), (out-in)                              |
| 2  | Sites where it is connected                                                 |
| 3  | What days and at what times it is connected                                 |
| 4  | What days it should be connected and what days it should not                |
| 5  | What times it should be connected and at what times it should not           |
| 6  | What IPs out of schedule have been connected or have had in/out traffic    |
| 7  | What IPs have had access in milliseconds or many times in a second to one or several sites |
| 8  | What external IP's the computer usually connects to                         |
| 9  | The IP requests DNS resolves which do not exit                              |
| 10 | The IP is connected to external addresses from questionable reputation sites|
| 11 | IPs with many firewall DENYs                                                |
| 12 | IPs which connect continuously with DNS servers                             |
| 13 | Very high traffic of an IP to the outside                                  |
| 14 | Average traffic of an IP                                                   |
| 15 | IPs that greatly exceed their average traffic (thresholded)                |

Table 2. Discriminant information to categorize malicious behavior

The synthetic logs were generated with the help of several correlation rules that simulate combinations of values in the instance that are usually related to malicious activity. The number of such correlation rules may be high,
and directly depends on the size of malicious behavior in the initial sample. The actual records in our real data
allow to establish accurate rules or hypothesis, but increasing the number of tests or adaptations could give better
approximations to reality.

Let $x_i$ ($i = 1, \ldots, 9$) be the value of each state in the instance, as described in 3.3.3 (e.g. $x_1 = x_{\text{date}}$, $x_9 = x_{\text{group behavior}}$). Then, Equation 4 shows the following correlation rules, where $|w|_a$ stands for the number of $a$’s in string $w$:

$$x_9 = \begin{cases} r(\text{red}) & \text{if} \quad \frac{|x_3| > 2}{(|x_2| > 2) \land (|x_3| > 0) \land (x_1 = 2)} \lor (x_1 = 3) \lor (x_2 > 1) \land [(x_3 = 3) \lor (x_5 = 3) \lor (x_6 = 3) \lor (x_7 = 3)] \lor (x_3 = 3) \lor (x_5 = 3) \lor (x_6 = 3) \lor (x_7 = 3)] \\ \text{g(\text{green}) in other case} & \end{cases}$$

The result of applying the correlation rules corresponds to the group behavior – last field of the instance.
Hence, some examples of instances after using the correlation rules are the following: $(1, 1, 1, 2, 1, 1, 1, 2, g)$, $(1, 2, 2, 1, 1, 2, 1, 1, 2, g)$, $(2, 1, 1, 1, 1, 1, 2, 3, 1, r)$, $(2, 3, 1, 1, 1, 1, 1, 1, r)$ or $(1, 3, 1, 1, 1, 2, 1, 1, 1, r)$.

All the malicious (red) behavior in our dataset is synthetic, but such red traffic activity incorporates knowledge
from actual exfiltration attempts that had been formerly detected in the real, in operation infrastructure. It is assumed
that the malicious synthetic traffic corresponds to 100% (98% synthetic + 2% copy of malicious samples taken from
real traffic).

The synthetic logs are injected by two applications written in Python. The first one provides an interactive
environment that allows generating logs assigning values to each field using some predefined criteria. The
configuration parameters are the source filename and the number of logs to be inserted into the source file so as
to simulate attacks over the infrastructure.

The second application massively injects logs of attacks without user intervention in order to mix harmless and
malicious activities, getting for each record –lines in the log file– as many lines as fields in the record, and increasing
the value of the fields in some percentage above the average. These logs had information from previously injected
attacks.

### 3.3.5 From instances to sample datasets

The sample datasets $S_i$ ($i = 1, \ldots, 9$) suitable for feeding machine learning algorithms were created from the real
and synthetic instances. The samples are composed of 20% random real data and 80% synthetic data, with different
proportions of green/red behavior so that we might find the best ratio for our model (see Table 3).

| Sample | Green Behavior (%) | Red Behavior (%) |
|--------|--------------------|-----------------|
| $S_1$  | 23.70              | 76.30           |
| $S_2$  | 69.95              | 30.05           |
| $S_3$  | 52.28              | 47.72           |
| $S_4$  | 48.98              | 51.02           |
| $S_5$  | 50.00              | 50.00           |
| $S_6$  | 70.83              | 29.17           |
| $S_7$  | 12.00              | 88.00           |
| $S_8$  | 60.48              | 39.52           |
| $S_9$  | 100                | 0               |

Table 3. Sample Datasets, $S_i$
3.4 Data analysis

This section describes the machine learning techniques used with the dataset, that include Naïve-Bayes, Decision Tree (ID3-C4.5) and Artificial Neural Networks.

Naïve-Bayesian classifier learns the conditional probability of each attribute $A_i$ from the training data given the class label, $C$. After the training stage, the probability of $C$ given one particular instance of $A_1, \ldots, A_n$ is computed by applying Bayes rule in order to predict the class with the highest probability [25]. In this work, the classifier uses the number of rows per attribute value and class for those attributes that are nominal, and a Gaussian distribution for the numeric attributes.

Decision Tree Induction is frequently used in Machine Learning or Data Mining because of its remarkable advantages: they are capable of learning functions from discrete values, even with noisy samples, and obtain sets of expressions that can be easily translated into sets of rules.

In particular, the C4.5 algorithm belongs to the Top Down Induction of Decision Trees family (TDIDT). It generates a decision tree using a “divide and conquer” algorithm, and evaluates information in each case using the following criteria: entropy, gain or proportion gain, as applicable. Besides, the heuristic is based on statistics, making it robust to noise [23].

Artificial neural networks can make decisions from a numerical set of examples, as the function is implicitly determined by that set of examples. Therefore, their objective is simulating the function that characterizes all the elements in the set. Inputs are numerical in the scheme attribute value. The learning method seeks to minimize the error for all the training examples, and has a great capacity to absorb noise [23].

In particular, we used the Probabilistic Neural Network (PNN) based on the DDA (Dynamic Decay Adjustment). This PNN works with labeled data using Constructive Training of PNN as the underlying algorithm, where each rule is defined as a high-dimensional Gaussian function adjusted by two thresholds in order to avoid conflicts with rules of different classes [26]. In particular, the training sets consisted of 65% of each sample, $S_i$, while the remaining 35% was used for test. Table 4 shows the proportions of green and red behavior (GB, RB) in each sample.

**Table 4.** Proportions of green/red behavior (GB/RB) in the training and test sets

| Sample | GB (%) | RB (%) | GB (%) | RB (%) |
|--------|--------|--------|--------|--------|
| $S_1$  | 15.40  | 49.50  | 8.30   | 26.80  |
| $S_2$  | 45.45  | 19.53  | 24.50  | 10.52  |
| $S_3$  | 33.98  | 31.02  | 18.30  | 16.70  |
| $S_4$  | 31.84  | 33.16  | 17.14  | 17.86  |
| $S_5$  | 32.50  | 32.50  | 17.50  | 17.50  |
| $S_6$  | 46.03  | 18.97  | 24.80  | 10.20  |
| $S_7$  | 7.60   | 57.40  | 4.40   | 30.60  |
| $S_8$  | 39.28  | 25.72  | 21.20  | 13.80  |
| $S_9$  | 65.00  | 0      | 35.00  | 0      |

After the training stage, the results with the test sets pointed to Decision Tree as better choice than Naïve-Bayes and PNN. For that very model and for every sample, $S_i$, its resistance is measured using sample, $S_j$ with different green/red behavior proportion as validation tests; for instance $S_6$ validates the model built using $S_1$, while $S_2$ is validated by $S_1$ and so on. In all the analysis, the improvement of each model with respect to the trivial one has been measured. Such trivial model would use the most frequent behavior in the sample to label every unknown activity, i.e. if most of the activity in the sample is green behavior, then the trivial model would label all the elements as green behavior.

Furthermore, sensitivity analysis of red behavior tests has also been accomplished because it is important to avoid false negatives when detecting harmful behaviors in the context of Cybersecurity.

Finally, the values of accuracy and resistance accuracy, of the improvements achieved over the trivial model, and the sensitivities to red behavior would be used to analyze the performance with the decision tree. These values
would, then, be quantized considering the quartile they belong to (4 for the upper quartile, 1 for the lower one), and their average for each sample would estimate its fitness, corresponding the best sample to the highest average.

4 Results

Using the confusion matrices of the analysis tests described in the above section over each $S_i$, we have obtained the results shown in Table 5. We have included the accuracy and the error obtained with the techniques of Naïve Bayes (NB), Decision Tree (DT) and Artificial Neural Networks (ANN).

The confusion matrices results of the analysis described in Section 3.4 are summarized in Table 5, which shows the ID3-C4.5 decision tree provides better accuracies and errors than Naïve Bayes and the probabilistic neural network. Hence, Table 6 shows the values of accuracy, improvement over the trivial model and sensitivity for each of the samples when using such decision tree.

Table 5. Accuracies and errors using Naïve Bayes, a decision tree and a probabilistic neural network

| Sample | Naïve Bayes | ID3-C4.5 Decision Tree | PNN |
|--------|-------------|------------------------|-----|
|        | Accuracy (%) | Error (%)              | Accuracy (%) | Error (%) | Accuracy (%) | Error (%) |
| $S_1$  | 90.26       | 9.74                   | 95.96 | 4.04 | 91.12       | 8.88 |
| $S_2$  | 88.47       | 14.53                  | 99.24 | 0.76 | 88.20       | 11.80 |
| $S_3$  | 89.03       | 10.97                  | 97.49 | 2.51 | 86.39       | 13.61 |
| $S_4$  | 89.15       | 10.85                  | 97.44 | 2.56 | 88.69       | 11.31 |
| $S_5$  | 86.76       | 13.24                  | 97.95 | 2.05 | 89.87       | 10.13 |
| $S_6$  | 85.52       | 14.08                  | 97.99 | 2.01 | 83.18       | 16.82 |
| $S_7$  | 89.20       | 10.80                  | 95.10 | 4.90 | 88.57       | 11.43 |
| $S_8$  | 85.25       | 14.75                  | 98.82 | 1.18 | 91.64       | 8.36  |
| $S_9$  | 83.29       | 16.71                  | 100   | 0   | 100         | 0     |

Table 6. Confusion matrix over $S_i$ after using the decision tree

| Sample | Accuracy (%) | Error (%) | Improvement over trivial model (%) | Sensitivity (g) | Sensitivity (r) |
|--------|--------------|-----------|------------------------------------|-----------------|----------------|
| $S_1$  | 95.96        | 4.05      | 19.73                              | 1               | 0.95           |
| $S_2$  | 99.24        | 0.76      | 29.24                              | 0.98            | 1              |
| $S_3$  | 97.49        | 2.51      | 45.20                              | 0.95            | 1              |
| $S_4$  | 97.44        | 2.56      | 46.42                              | 1               | 0.95           |
| $S_5$  | 97.95        | 2.05      | 47.95                              | 0.96            | 1              |
| $S_6$  | 97.99        | 2.02      | 27.13                              | 0.98            | 0.98           |
| $S_7$  | 95.10        | 4.90      | 7.67                               | 0.94            | 1              |
| $S_8$  | 98.82        | 1.18      | 38.25                              | 0.95            | 1              |
| $S_9$  | 100          | 0         | 0                                  | 1               | -              |

Table 7 shows the results of analyzing resistance accuracy, resistance improvement over the trivial model, and resistance sensitivity for every sample. Improvement over the trivial model is the result of subtracting the higher behavior value for the validation sample (in table 3) from the accuracy/resistance value. The table includes the samples that were used as validation tests, and does not consider $S_9$, as it does not improve the trivial model.

Figures 4 and 5 show the boxplots regarding the accuracy and resistance accuracy for each sample, as well as the improvements over the trivial model, and the sensitivities. Last, Figure 6 shows a bar chart with the mentioned variables after quantization.
Table 7. Results over validation sets of sample datasets $S_i$.

| Sample | Validation sample | Accuracy/Resistance (%) | Improvement over trivial model (%) | Sensitivity (r) |
|--------|-------------------|-------------------------|-----------------------------------|----------------|
| $S_1$  | $S_2$             | 87.37                   | 17.42                             | 0.831          |
| $S_2$  | $S_1$             | 99.98                   | 23.68                             | 1              |
| $S_3$  | $S_4$             | 100                     | 48.98                             | 1              |
| $S_4$  | $S_3$             | 100                     | 47.72                             | 1              |
| $S_5$  | $S_1$             | 99.98                   | 23.68                             | 1              |
| $S_6$  | $S_1$             | 84.14                   | 7.84                              | 0.79           |
| $S_7$  | $S_2$             | 100                     | 30.95                             | 1              |
| $S_8$  | $S_1$             | 96.63                   | 20.33                             | 1              |

Fig. 4. Boxplots of accuracy, improvement over trivial model and sensitivity

Fig. 5. Boxplots of resistance accuracy, resistance improvement over trivial model and resistance sensitivity

5 Discussion

The experimental results led to choosing ID3-C4.5 decision tree to detect anomalous behaviors in the network activity. In order to select those samples that fit best according to accuracy, improvement over the trivial model, sensitivity to red behavior, resistance’s accuracy, resistance’s improvement and resistance’s sensitivity to red behavior, their values are quartile binned first. Then, the mean of such binned variables is used as a measure of the fitness. The results shown in Figure 6 point to $S_3$ and $S_5$ as the best ones, both of them with the higher mean values (equal to 3.33). The resulting model has been used to develop an intelligent system that takes the firewall raw logs as inputs and fires alerts in case of potential APT activity (Figure 7).

The system has proven to be effective, and uses technologies that do not depend on the architecture. However, the model would require continuous updating based on monitoring suspicious activity so as to improve the accuracy of logs categorization. Furthermore, the use of distributed data storage and HPC (High Performance Computing) technologies would allow real-time processing and, hence, improving the performance and, eventually, anticipating the APTs actions.
Fig. 6. Quantized values of accuracy and resistance accuracy, improvements over trivial model, and sensitivities

![Valuation of each Sample](image)

Fig. 7. State diagram of the decision tree based intelligent system

![State diagram](image)
6 Conclusions and future work

The proposed intelligent system predicts suspicious behaviors by analyzing the data traffic in an IT infrastructure, and triggering alerts so that the administrator does not have to read the whole log files. The results conclude that the proposal is suitable for the goal of early detection of APTs, i.e. for proactive security.

Future work is focused on improving the model by monitoring suspicious results and, thus, defining the process of cataloguing such anomalous behaviors. Besides, performance might be improved with the incorporation of real time HPC and Big Data technologies.

Appendices

I. Pseudocode: from raw logs to normalized logs

DO UNTIL end input data:
    Add item to the list.
    Next item
Endo

select file.
Record file list.
Open finput;*** Rawlog file
Open foutput;*** Logs Standardized file.
Read finput:
DO UNTIL FF:
    DO FOR i=0 to end of list;
        Take list field.
        Search and take record file
        Insert data record foutput.
    Endo
    Write record foutput.
    Read finput.
Enddo
Close finput, foutput

II. Running the model against real data

We have tested the S5 model using the decision tree with KNIME over two real datasets. On the one hand, the first dataset represents normal activity, i.e. with no APT logs. The second dataset, on the other hand, contains data concerning one APT that attacked an actual infrastructure and that remained persistent for 25 days, until it was detected by inspection (and removed). During that time the APT generated 3710 firewall log entries. The experiments were carried out sampling the datasets while maintaining the proportion of dangerous/innocuous log registers.

Note that the S5 model gives no false alerts with any of both datasets (Figures 8 and 10), and that does not happen with other models, even for harmless datasets (Figure 9). Although it gives a number of false alerts with the second dataset, the true ones are significant enough to effectively detect and remove the attack, since all of the registers came from the same source.
Fig. 8. Confusion matrix using S5 model and the decision tree over a harmless dataset, $D_h$

Fig. 9. Confusion matrix using S3 model and the decision tree over $D_h$

Fig. 10. Confusion matrix using S5 model and the decision tree over a dataset that contains APT activity
References

[1] Falliere N., Murchu L.O., Chien E., W32.Stuxnet dossier, White paper, Symantec Corp., Security Response 5, 2011
[2] Holguín J.M., Moreno Maite, Merino B., Detección de APTs, CSIRT-CV and INTECO-CERT, Comunidad Valenciana-León, 2013
[3] Oprea A., Li Z., Yen T. F., Chin S. H., Alwais S., Detection of early-stage Enterprise infection by mining large-scale log data, In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, 45–56
[4] Mosso J.M.R., Ciberseguridad Inteligente, arXiv preprint arXiv:1506.03830, 2015
[5] Friedberg I., Skopik F., Settanni G., Fiedler R., Combating advanced persistent threats: From network event correlation to incident detection, Computers & Security, 2015, 48, 35–57
[6] Giura P., Wang W., Using large scale distributed computing to unveil advanced persistent threats, Science Journal, 2012, 1(3), 93–105
[7] Amoroso E.G., Fundamentals of computer security technology, Upper Saddle River, NJ, USA, Prentice-Hall, Inc., 1994
[8] Schneier B., Attack Trees - Modeling Security Threats, Dr. Dobbs Journal, 1999, https://www.schneier.com/academic/archives/1999/12/attack_trees.html
[9] Skopik F., Settanni G., Fiedler R., Friedberg I., Semi-synthetic data set generation for security software evaluation, In: 12th Annual Conference on Privacy, Security and Trust. IEEE, 2014, 156–163
[10] Veeramachaneni K., Ardalo I., Korrapti V., Bassias C., Li K., a²F²: training a big data machine to defend, In: Proceedings - 2nd IEEE International Conference on Big Data Security on Cloud, IEEE BigDataSecurity 2016, 2nd IEEE International Conference on High Performance and Smart Computing, IEEE HPSC 2016 and IEEE International Conference on Intelligent Data and Security, IEEE IDS 2016, 2016, 49–54
[11] Shyu M. L., Chen S. C., Sarinnapakorn K., Chang L., A novel anomaly detection scheme based on principal component classifier, In: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM’03), 2003, 172–179
[12] Zinksecurity Thinking solutions, Advanced Persistent Threats (APTs), Guardia Civil, España, 2015
[13] Yao Y., Zhang J., Zhang Y., Yu Y., Lu J., A method and implementation of control flow obfuscation using SEH, In: Proceedings of the 4th International Conference on Multimedia Information Networking and Security, MINES 2012, 2012, 336–339
[14] Wei Q., Wei T., Wang J., Evolution of exploitation and exploit mitigation, Journal of Tsinghua University, 2011, 51(10), 1274–1280
[15] Support Microsoft, How to enable Structured Exception Handling Overwrite Protection (SEHOP) in Windows operating systems, 2011, https://support.microsoft.com/en-us/help/956607/how-to-enable-structured-exception-handling-overwrite-protection sehop in-windows-operating-systems
[16] Dang T.H., Maniatis P., Wagner, D., The performance cost of shadow stacks and stack canaries, In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2015), 2015, 555–556
[17] Sood A., Enbody R., Targeted Cyber Attacks: Multi-staged Attacks Driven by Exploits and Malware, Syngress, 2014
[18] Sharp B.L., Peterson G.D., Yan L. K., Extending hardware based mandatory access controls for memory to multicore architectures, In: Proceedings of the 4th annual workshop on Cyber Security and Information Intelligence Research: Developing Strategies to meet the Cyber Security and Information Intelligence challenges ahead (CSIIIRW ’08), ACM, 2008, 23:1–23:3
[19] Yotiyana J. P., Mishra A., Secure Authentication: Eliminating Possible Backdoors in Client-Server Endorsement, Procedia Computer Science, 2016, 85, 606–615
[20] López V., Fernández A., García S., Palade V., Herrera F., An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics, Information Sciences, 2013, 250, 113–141
[21] Han H., Wang W.Y., Mao, B.H., Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, In: Proceedings of the 2005 International Conference on Intelligent Computing (ICIC’05), Lecture Notes in Computer Science, 2005, 3644, 878–887
[22] He H., Bai Y., Garcia E.A., Li S., ADASYN: adaptive synthetic sampling approach for imbalanced learning, In: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN’08), 2008, 1322–1328
[23] Borrajo D., González J., Isasi P., Aprendizaje Automático. Ed. Sanz y Torres, S.L, 2013
[24] López-Cabeceira M.M., Diez-Machío H., Trobajo M.T, Carriegos, M.V., Spectra analysis in detection of traces of explosives, Int. J. Modern Phys. B, 2012, 26(25), 1246013
[25] Friedman N., Geiger D., Goldszmidt M., Bayesian Network Classifiers, Machine Learning, 1997, 29(2–3), 131–163
[26] Berthold M.R., Diamond J., Constructive training of probabilistic neural networks, Neurocomputing, 1998, 19(1), 167–183