Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

TC 11 Briefing Papers

Feature analysis for data-driven APT-related malware discrimination

Luis Francisco Martín Liras[a], *Adolfo Rodríguez de Soto*[a], *Miguel A. Prada*[b]

[a] *Dpto de Ingenierías Mecánica, Informática y Aeroespacial Universidad de León, 24071, León, Spain*
[b] *Suppress research group. Dpto de Ingenierías Eléctrica, de Sistemas y Automática Universidad de León, 24071, León, Spain*

ARTICLE INFO

Article history:
Received 12 March 2020
Revised 25 November 2020
Accepted 11 January 2021
Available online 16 January 2021

Keywords:
Malware
Advanced persistent threat
Machine learning
Exploratory data analysis
Feature analysis

ABSTRACT

Advanced Persistent Threats (APTs) have become a major concern for IT security professionals around the world. These attacks are characterized by the use of both highly sophisticated, evasive and cautious human and technical resources. It is very common to notice the combined use of different malware in long APT campaigns. This fact makes it interesting to investigate the malware that has been used in APT campaigns. Different approaches have been proposed to find discriminatory features to detect APT malware. Features from either static, dynamic and network-related analyses have been separately proposed for that aim. The new approach considered in this study aims to identify the most discriminatory features to distinguish APT-campaign-belonging malware from non-APT malware executables. This approach suggests to identify the discriminatory features from not one but all three groups of these analyses by using domain knowledge and with a purpose of interpretability. As a result, a set with the most discriminatory features of each type is provided. To achieve this set, well-known machine learning techniques have been used. One of the most important limitations in the use of these learning techniques is the availability of a relevant amount of data. In this paper, a large dataset of 19,457 malware samples is publicly provided, including both malware known to be related with APTs and generic non-APT-belonging malware samples. In order to analyze the discrimination ability of the features, the proposed approach follows several steps. First, an exploratory analysis is conducted to obtain knowledge about the data structure. Later, feature selection is performed using different discriminatory techniques. The resulting selection of features is assessed by means of four well-known binary classification techniques. The high accuracy of the results shows that the proposed features are discriminative enough for the stated purpose. Finally, these results are interpreted and the findings are discussed from the perspective of prior knowledge and assumptions about APT-related malware.

1. Introduction

An Advanced Persistent Threat (APT) is a computer attack, generally motivated by business or political motives, in which a person or group gains unauthorized access to a system or

E-mail addresses: lmarl@unileon.es (L.F. Martín Liras), arods@unileon.es (A.R. de Soto), ma.prada@unileon.es (M.A. Prada).
https://doi.org/10.1016/j.cose.2021.102202

network and remains undetected for an extended period of time. It is qualified as "advanced" because it requires sophisticated hacking techniques to exploit vulnerabilities in the target system. The "persistent" qualification indicates that it continues being undetected, being monitored and data being extracted for a long period of time (Chen et al., 2014).

Advanced Persistent Threats (APTs) are a major concern for IT security professionals around the world. In recent times Advanced Persistent Threats (APT) have been among the most highlighted threats for security experts. Although APT definitions have been widely discussed (Daly, 2009; Nicho, 2014; Tankard, 2011), it is globally accepted that APTs are developed by determined and well-organized teams (Chia-Mei Chen, 2018). At early stages, APT attacks were addressed to government or financial organizations, but recent studies based on security breaches indicate that such attacks are now carried out on much wider domains. All industry sectors can nowadays be a target of an APT campaign (e.g., the energy sector, healthcare, education and media can be a specific target)(Chen et al., 2014).

The major concern about APTs is that detection of these attacks is extremely difficult, to the point that even major anti-virus firms doubt about the ratio of discovered APT attacks against real attacks (Chia-Mei Chen, 2018). The attackers carefully elude it to ensure persistence and a variety of malware can be used in any stage of the attack to adapt to the circumstances. For that reason, when working with APT attacks, it becomes very difficult to discern whether a malware sample becomes part of an APT attack or not. In many occasions only an expert, after a thorough work of analysis, is capable to decide if a specific sample has the characteristics, time period and opportunity to be another member of a wide set of malware samples that belong to a specific APT attack. And what is more, APT developers are mostly unknown and researchers try to guess those behind the attacks by means of strings and signatures found on the malware samples. Unfortunately, in many cases, they can only partly guess the origin of these developers. It is the case that anti-malware firms find that some old malware samples (sometimes even more than 7–8 years old) might belong to a recently discovered APT campaign. From these analyses, it is often illustrated that source code is reused among different samples of malware and therefore among different samples of APTs (Ruttenberg et al., 2014). As shown by Lai et al. (2011) through the analysis of several APTs behavior, APT samples are naturally grouped into families corresponding to the different developing groups.

Most previous research on APTs is focused on the analysis of their nature (Ussath et al., 2016), life cycle (Vukalovic and Delija, 2015) and network behavior (Lamprakis et al., 2017) instead of on the malware involved in the course of the attack. However, the understanding of the features that characterize the malware used in APTs might help security analysts. This study can be addressed through the collection of interpretable malware features and the use of machine learning techniques to analyze which ones enable a better discrimination of APT-related malware.

The application of machine learning (ML) techniques for the automatic detection or classification of malware samples is a common topic in the literature (Gandotra et al., 2014). Nevertheless, previous works in this area seldom consider specifically the detection of APT-belonging malware. The ones actually covering the detection of APT-related malware have focused so far on a single class of features, generally obtained from the analysis of network behavior (Niu et al., 2017; Zhao et al., 2015) or from static malware analysis (Laurenza et al., 2017; Sexton et al., 2016). It would be interesting to consider different types of features in the study of APT-related malware not only to aim at a better accuracy but also to acquire a better understanding of its nature. For this reason, this work proposes the use of static, dynamic and network-related features, interpretable and selected through domain knowledge, and well-known machine learning techniques to analyze the discriminability of APT-related malware from generic malware without any known association to APTs.

To assess the proposed approach, it is necessary to rely on a dataset that gathers malware used in APT campaigns and generic malware with no previous known association to any APT. To the extent of our knowledge, there was no public dataset with those characteristics. For that reason, in the context of this work, a large dataset with APT-related and non-APT generic malware has been collected and publicly provided (Martin-Liras et al., 2020). This malware dataset contains a total of 19,457 malware analyzed samples (1,497 APT samples and 17,960 non-APT-belonging malware) and 1944 features obtained from the processing of fields extracted from static, dynamic and network traffic analyses. In this paper, the collection, preprocessing and cleaning of this APT malware dataset is described.

The dataset is analyzed from different perspectives by means of ML techniques with the goal of a better understanding of the features of APT malware. An exploratory analysis is first conducted to obtain knowledge about the data structure. Later, a feature selection process is performed using different filter and embedded techniques to find features with high discriminative power to determine whether a malware sample belongs or not to an APT. Feature selection provides a reduced set of relevant features that is assessed by means of well-known classification techniques.

The main contributions of this paper are:

1. A large dataset of Windows PE32 executable malware including 1497 APT-related malware samples and 17,960 non-APT generic malware samples that solves the lack of public data for experimentation in the topic of APT-related malware discrimination.
2. The definition of 1941 binary features with a clear semantic interpretation obtained from static, dynamic and network-related analyses of the malware samples. This hybrid set of features has been rarely used in the literature of malware detection and classification and, to the authors' knowledge, it has not been used in the area of APT analysis.
3. The analysis of those features to determine the most representative subset. As a result, 238 features are selected. A non-exhaustive interpretation of the obtained results leads to findings that seem to empirically support common preconceptions about APTs.
4. The assessment of the discrimination ability of this reduced feature set by means of four well-known binary classification techniques, resulting in all cases in high accuracies.

The structure of this paper is as follows. First, related work is reviewed in Section 2. In Section 3, the proposed dataset is presented, explaining the process of data gathering, selection of potentially interesting features, data preprocessing and cleaning. Section 4 presents the methods proposed for the analysis of the dataset, which includes exploratory analysis, feature selection and binary classification. Experimental results are described and discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Related work

Although there already exist commercial tools for APT detection, they can be bypassed with moderate effort (Ács -Kurucz et al., 2014; Brogi, 2018) so it is necessary to propose better solutions. Existing research efforts have proposed five different approaches for the detection of APTs or a mixture of them: (i) formalization/modelling of the APT detection problem; (ii) analysis of the network traffic generated by APTs (with or without a focus on APT malware); (iii) static analysis of APT malware; (iv) analysis of features built from the malware raw binary; and (v) dynamic analysis of the malware.

Some works have tried a **formalization of the APT detection problem**. In their study, (Bhatt, Yano, Gustavsson, 2014) proposed a 7-phase detection model to identify different steps of an APT. Giura and Wang (2012), proposed an attack pyramid aiming to capture movements of an attacker through different domains (e.g., physical, network, application). Another study proposed possible building blocks for a framework of APT detection (de Vries et al., 2012). However, all these approaches are limited to the proposal of guidelines that should be used to build methods for APT detection, but the definition of detection rules and approaches is left to future work.

In general, the more usual approach used for APT malware detection is through the **analysis of the network traffic** generated by the APT samples. In this regard, Lamprakis et al. (2017) based their research on HTTP traffic generated by command & control (C&C) servers of APTs and generated HTTP traffic request maps to discern the APT involved in the communication. Vance (2014) showed that it could be possible to detect APTs by analyzing Netflow traffic in order to discover C&C connections, whereas Friedberg et al. (2015) used event correlation for APT malware detection. Siddiqui et al. (2016) proposed a fractal-based classification method of anomalous patterns by using several features of a TCP/IP connection. Other works have analyzed both traffic and data obtained from the execution of the APTs. That is the case of Pei et al. (2016) that correlated features extracted from different logs, including HTTP, DNS and system logs to generate multi-dimensional weighted graphs useful for APT detection. In some cases, this analysis of network behavior is focused on the detection APT-related **malware**. In their work, Zhao et al. (2015), used 14 features obtained from the communication with C&C servers, especially from the DNS queries commonly used in this context. Niu et al. (2017) also analyzed DNS traffic to identify an APT from logs of mobile devices.

With regard to the analysis of executable files, a common approach is **static analysis**, i.e., the dissection of resources available in the binary file without the need of executing it. These resources include libraries and functions used and exported, sections of the executable, packers or assembly instructions. Static analysis of PE Headers are well known to possess a strong discriminatory power(Yan et al., 2013). Laurenza et al. (2017) built an APT malware triage framework relying on static malware features obtained from the file headers, obfuscated strings, imports, functions and directories.

The static analysis can follow a different path centered in certain **features of the raw binary file**. This approach uses directly the raw bytes or other low-level elements to build features, e.g., n-grams obtained from byte sequences. Although these features are easy to compute, they generally provide little interpretable information and might show some limitations. For instance, it has been shown that Markov n-gram malware detectors tend to provide a high false positive rate for certain types of files (Shafiq et al., 2008). Sexton et al. (2016) proposed an APT malware detection approach based on the similarity of subroutines, where n-grams on the sequence of opcodes are used as features to summarize the subroutines.

Dynamic analysis (also known as behavioral analysis), which executes malware in a controlled and monitored environment to observe its behavior and its interaction with other resources, might provide information about malware that static analysis cannot find due to obfuscation. Nevertheless, it is difficult to simulate suitable conditions where the malicious behavior of the malware is revealed, since it is usually dependent on the environment and the period of time that needs to be observed is unclear (Sornil and Liangboonprakong, 2013). Furthermore, although obfuscation is more difficult than for static analysis (Rhode et al., 2018), modern malware can exhibit a wide variety of evasive techniques designed to defeat dynamic analysis including testing for virtual environments or active debuggers, delaying execution of malicious payloads, or requiring some form of interactive user input (Vasilescu et al., 2014). This approach has been scarcely used in the context of APTs: Su et al. (2016) proposed a framework to perform this kind of analysis for APTs whereas Li et al. (2011) considered static and dynamic analyses in a case study of APT-related malware.

The application of machine learning (ML) techniques for the automatic detection or classification of malware samples is a common topic in the literature (Gandotra et al., 2014), but there are few works where features from static and dynamic analysis are combined (Damodaran et al., 2017). In the particular case of APT-related malware, the application of ML techniques is a topic still under-explored where previous work includes the use of decision trees (Zhao et al., 2015), anomaly detection techniques (Niu et al., 2017) and classification algorithms (Sexton et al., 2016). A more similar approach to the one proposed in this paper is that of Laurenza et al. (2017), who use a random forest classifier in an APT malware triage system. The main difference with our work is that only features obtained from the static analysis are used. Furthermore, its aim is fast prioritization of malware to be later inspected by human analysts, instead of an analysis of the discrimination ability of interpretable features. The advanced techniques that have been recently proposed in general malware detection and classification to model sequences of events obtained from dynamic analysis, e.g., deep neural networks

(Kolosnjaji et al., 2016) and recurrent neural networks (Rhode et al., 2018), are yet to be applied to APT-related malware.

With regard to the **malware datasets** and/or benchmarks, there are at least two well-known publicly available datasets that have been described in the literature (Anderson and Roth, 2018; Ronen et al., 2018). Nevertheless, they do not discern between APT-related and generic malware because they were not designed for that purpose. The dataset used in the Microsoft classification challenge (Ronen et al., 2018) focuses on static features of malware and contains over 20,000 samples of nine different malware families. The dataset gathered by Anderson and Roth (2018) is a large collection of 1.1 million samples that analyze a group of features obtained again from the static analysis of Windows executable samples. Laurenza et al. (2017) describes an interesting dataset with labeled APT-related and generic malware and only static features.

3. Malware dataset

As mentioned, a large dataset of well-known malicious samples was gathered for this work. This dataset was composed of two sub-datasets that were later joined: a first dataset including a set of well-known APT malware samples, and a second dataset with a collection of "general" malware samples. The executables of these two datasets were analyzed both from a static and dynamic point of view. Existing dynamic analyses from VirusTotal[1] were used, as well as static analyses for the minority of malware samples that could not be gathered. VirusTotal is a renowned and trusted virus scanner that analyzes daily nearly one million distinct files using over 50 different tools (Rhode et al., 2018). As a part of the dynamic analysis, network traffic and several network-related features were also included in the experiment.

3.1. *Data gathering*

It was decided to focus the dataset on Windows executable malware samples (Windows PE32 Executables, Windows DLL libraries and Windows OCX files) belonging to either "general" malware or APT-related malware samples, considering as general the malware samples that, so far, are not known to belong to any APT campaign or APT development group.

General malware samples were obtained from Virusshare[2]. A total of 134,749 general malware samples corresponding to the period 2012–2015 were initially gathered from this resource. This raw malware set was used to detect the most typical packers, imports, exports, resources, among others features found in malware over that period. Nevertheless, the final analysis was performed with a lower number of samples, as not all gathered samples had been previously analyzed by VirusTotal, others were not eventually considered malware by a minimum quorum of antimalware solutions or simply the information included in the VirusTotal report was incomplete or not informative enough.

As a preliminary stage, we performed further preprocessing on the generated dataset. A filtering process was performed to discard instances with fewer than 5 antimalware positive detections in VirusTotal and those that were not present in its database. Secondly, we discarded the instances with any missing value among the selected features.

The final general malware set eventually included a total of 17,960 samples that can be classified as follows: Up to 10,207 instances of malware were mainly considered trojans, 1117 were considered downloaders, 775 rootkits, 1270 worms, 616 spyware and 3975 of other types. This classification used the names given to each sample by Kaspersky Labs.

Regarding APT-related malware samples, 758 of them were collected from ContagioDump[3] and TheZoo[4] websites. To increase the coverage of APT-belonging malware samples available for analysis, we also gathered information from 739 additional APT-belonging malware samples obtained from Kaspersky's SecureList reports. These additional 739 reports were gathered from VirusTotal. Hence, a total of 1497 APT-related malware samples were considered, belonging to the following APT campaigns, based on Kaspersky reference names: APT1, APT28/Sofacy, BlackEnergy, Carbanak, Careto, CosmicDuke, CozyDuke, DarkHotel, Dark Seoul, DragonFly, Duqu, Duqu2, EquationGroup, Flamer, Gauss, Hellsing, Icefog, Kimsuky, Machete, MiniDuke, NetTraveler, Poseidon, RedOctober, Regin, Stuxnet, WildNeutron and Winnti. Again, these 1497 APT malware samples can be classified by type. A total of 827 instances were considered trojan, 370 were considered worms, 158 were considered downloaders and 142 were considered of other types.

For both general and APT-related malware samples, static and dynamic analyses were considered. **Static analyses** were performed for the entire dataset, using a Python script based on the Python PEFile library (Mulder et al., 2011), whereas information from **dynamic** analyses (including network traffic features) was extracted from reports obtained from VirusTotal.

All this information was stored in a MySQL database, totaling more than 105 tables of data. The sources of information considered for the samples and analysis of malware are summarized in Table 1.

3.2. *Proposed features, encoding and preprocessing*

The proposed features that are included in the dataset emerge from the objective of approaching the point of view of malware developers and, consequently, only features with a natural interpretation are considered, excluding other features obtained from the analysis of the raw binary which are commonly found in the literature (see Section 2). The reason to include features obtained from static analyses, dynamic analyses and network traffic is to obtain a dataset as complete as possible, in order to overcome the individual disadvantages of each approach that were briefly presented in the previous section. Furthermore, the proposed dataset focuses on features with a natural interpretation, which are

[1] www.virustotal.com.
[2] www.virusshare.com.
[3] Parkour, M.: ContagioDump, http://contagiodump.blogspot.com.
[4] Nativ, Y.: The Zoo Malware DB, http://thezoo.morirt.com.

Table 1 – Sources of samples and analyses of the malware dataset.

Malware source	# of instances	Type of malware	Source of static analysis	Source of dynamic analysis
Virusshare	**17,960**	General	Own	VirusTotal
ContagioDump & TheZoo	**758**	APT-related	Own	VirusTotal
-(Not gathered)	**739**	APT-related	VirusTotal	VirusTotal

more easily understood by an expert in the context of an exploratory analysis. Under those premises and considering all the information obtained from static and dynamic analyses of both the general malware and APT-related malware samples, we selected a total of 19 data fields to be extracted from the corresponding analysis reports and be included as features in the target dataset. Among these fields, 9 were extracted from the static analyses, 7 from the dynamic analyses and 3 features were related to network traffic analyses.

The selection of these features was focused on the characteristics of APT-related malware during development and compilation. The inclusion of features from different approaches, such as static, dynamic, and network traffic analyses aimed to avoid the inherent limitations and disadvantages of the individual approaches.

Apart from the labels "isAPT" (binary) and "NumAPT" (a value identifying the APT campaign to which the malware sample has been associated), the fields considered for the dataset are presented in the rest of the section with a brief description.

3.2.1. *Static analysis fields*

After performing the static analyses of malware samples, a group of features was selected to continue processing. The following is the list of selected fields introduced in the final dataset:

Packers *(categorical, multiple values)* **.** Malware developers can use packers to avoid antivirus detection or for obfuscation purposes. Different packers can be used simultaneously. The 24 most typical packers in general malware (corresponding with over 80% of the packers detected) were considered individually as features.

Number of Packers by file *(numerical)* **.** APT malware samples are usually considered to be more sophisticated than general malware samples and therefore is to be expected that the number of packers be higher than in general non-APT belonging malware samples.

Type of detected malware *(categorical)* **.** This is a categorical feature with the following possible values: trojan, spyware, botnet, worm, backdoor, adware, downloader, hacktool, virus, rootkit or password-stealer. The malware names given by Kaspersky antimalware firms were used to determine the type of malware. This way, a sample named as *PSW.Win32.Kykymber.leh* was classified as a password stealer (PSW).

Imports *(categorical, multiple values)*. Every Windows executable file needs to use library functions to implement its functionality. By analyzing these library functions imported by a malware instance, we can give a glance to what the running sample is going to do. We considered the 200 most typically imported function names individually as features for the dataset.

Number of imports *(numerical)*. A bigger number of imports could indicate that a malware sample is more complex. Typically, manually-written APT malware samples are more sophisticated than those created from automatized malware development tools.

Overlays. Overlays are the parts of PE file that are not covered by the PE header. Instead, an overlay is data appended to the end of the executable file, which should never exist or, at least, not contain any executable code. However, opening the file for reading will allow access to the entire file including the Overlay portion. For this dataset a binary value was added indicating whether a malware sample includes an overlay or not.

Suspicious APIs *(categorical, multiple values)*. A list of suspicious APIs was obtained from the PEFile Python module and extended with the functions mentioned on Alazab et al. (2010); Vasilescu et al. (2014); Wang et al. (2009). This produced a list of 212 typically suspicious functions.

Languages *(categorical, multiple values)*: The different languages found on Windows PE32 executables' resources directory tables could sometimes explain the origin of an APT and its relationship with other APTs. A total of 123 languages were considered individually as features for the dataset.

Number of resources *(numerical)*: This feature indicates the of different resource entries found in the resource directory tables of the malware executable.

3.2.2. *Dynamic analysis fields*

The fields extracted from dynamic analyses are explained next:

Anti-debug techniques *(categorical, multiple values)*: Many malware samples use different techniques to avoid detection and debugging. Debugging is important among researchers as often is the only way to understand what the malware is doing and why.

Started Services *(categorical, multiple values)*. From the dynamic analysis process, a list of Windows services opened during malware runtime can be obtained.

Suspicious DLLs *(categorical, multiple values)*. The list of most generally imported DLL files was obtained among all the initially downloaded samples of malware. The 100 most recurrent DLL files imported by the whole malware repository were considered as individual features.

File-related features: Files_opened, Files_removed, Files_written, Files_read *(categorical, multiple values)*. These features represent a binary list of directories where files were created, accessed, written and/or removed during the malware execution, without taking into account the file names or extensions.

Table 2 – Features disaggregated by type.

Original database field	An. type	# features in dataset	Original Database field	An. type	# features in dataset
Packers	S	24	Number of resources	S	1
Number of Packers in sample	S	1	Imports	S	161
Type of detected malware	S	1	Number of imports	S	1
Anti-debug techniques	S	20	Suspicious DLL	S	99
Started services	D	219	Directories where files were opened by malware	D	224
TCP IP address country	N	111	Directories where files were read by malware	D	224
UDP IP address country	N	111	Directories where files were written by malware	D	215
DNS IP address country	N	111	Directories where files were deleted by malware	D	224
Overlays	S	1	Malware Type	S	6
Suspicious API discovered	S	187			
Subtotal 1		**786**	**Subtotal 2**		**1,155**
			TOTAL		**1,941**

3.2.3. *Network traffic analysis fields*

Finally, one extra field was extracted from network traffic analysis:

- **Country of IP addresses for DNS, TCP and UDP traffic** (*categorical, multiple values*). From the dynamic analysis report, the IP addresses accessed were recovered and a geolocation script was run to obtain the country codes associated to those IP addresses, which might provide information about the origin of the groups involved in the APT.

Other fields were considered but eventually rejected in order to avoid a negative influence on the performance of the algorithms. In some cases, such as those features obtained from metadata ("*Internal_Name*", "*Product_Name*", "*File_Description*", etc.), the "*AntiVM_techniques*" or the "*contacted_domain*" feature, they were dropped due to the high rate of missing data (from 76% up to 93%). In other cases, the feature was unreliable, e.g., the "*First seen date*", which could help to confirm that a malware instance belongs to an APT, assuming that all samples in an APT campaign should have been created during the same period. However, the readily available timestamp is not the creation or compilation time, which are often forged, but the time of first submission to VirusTotal. Finally, some other features do not seem appropriate for the proposed analyses, because they are conceived to just identify almost identical files. Those are the cases of "*File Size*" "*Ssdeep*" and "*Imphash*". Ssdeep is a well-known program for computing context-triggered piecewise hashes of the file, which is used to find malware that has been slightly modified through insertions, modifications or deletions. Imphash is a hash value based on library/API names and their specific order within the executable, which takes advantage of the way the Windows import table is created and how the hash is calculated to identify similar malware instances.

It is important to understand that some of the 19 fields are multi-valued categorical variables, i.e., their values are a list of $n \geq 0$ categories. An encoding process is necessary for further analysis. For that purpose, the K most typical values of each field have been kept and encoded as binary features, using an analogous scheme to one-hot encoding. As a result of this n-of-K encoding, the preprocessed data set results in a total of 2144 individual features, most of them binary.

After a thorough analysis, further fields were removed. Some AntiVM techniques and languages were discarded because these data were rarely available. Finally, a preliminary correlation analysis (Zhang and Yang, 2015) led us to find some highly or completely correlated features among the imports, suspicious APIs, written directories or suspicious DLLs. These cases were generally due to value repetition during automated labelling. As a result, the number of features was reduced to 1941 features. A summary of the features finally kept in the data set is shown in Table 2.

Finally, the values of the fields were standardized. After these preprocessing steps, the resulting data set contained a total of 19,547 instances (17,960 of general malware and 1497 of APT-related malware) with 1941 columns.

4. Methods

Once the dataset was ready, it was processed in two stages. First, it was necessary to perform an exploratory data analysis, including steps such as correlation analysis and dimensionality reduction, to better understand the characteristics of the dataset. Second, a feature selection process in terms of the relevance of the variables for classification provided new knowledge about the differences between APT-related and general malware.

4.1. *Exploratory data analysis*

The objective of **exploratory data analysis** (EDA) is to understand the distribution of data and the relationships among features. In this case, our aim is to discern the overall structure of the dataset, summarize its main characteristics and obtain a visual representation of the dataset. For that purpose, two different analyses have been performed. First, an analysis of the linear **correlations** among every pair of features from the preprocessed dataset is performed. It is based on the Pearson coefficients (or Pearson's R).

Second, a **dimensionality reduction** process is proposed to obtain a visual representation of the dataset for its better interpretation, therefore reducing the number of dimensions from 1941 to only three dimensions, which can be projected for visualization. Both PCA and t-SNE were considered to accomplish this task. **PCA** uses orthogonal transformations to convert the high-dimensional space into a lower-dimensional space of uncorrelated variables that maintain the highest variances (Shlens, 2014). **T-SNE** (t-Distributed Stochastic Neighbor Embedding) (Van Der Maaten and Hinton, 2008), in turn, is a technique based on the computation of probability distributions over the pair-wise similarity matrices of the original high-dimensional space and the low-dimensional embedding space and the minimization of the divergence between them. It is a dimensionality reduction technique particularly well suited for the preservation of local structure that has proven to provide good visualization characteristics for real high-dimensional datasets. Furthermore, it is a non-linear method that does not impose such strong assumptions as PCA does. T-SNE was already used in the field of malware analysis by Nassar and Safa (2019) and David and Netanyahu (2015). For that reason, it has been chosen as an alternative for the projection of data to a lower-dimensional space.

4.2. *Feature selection and classification*

One of the main objectives of this experiment is to analyze the discriminative ability of different features with regard to APT-related malware classification. Apart from the interest on understanding the features linked with malware used in APTs, it is clear that a detection tool that relies on around 2000 features might be difficult to deploy in a production environment, where it is crucial that the system is practical and lightweight. On the other hand, it is well known the phenomena that arise when analyzing and organizing data in high dimensional spaces, known as the "curse of dimensionality" (Bayer, 2009; Trunk, 1979).

The feature selection process aims to obtain the most representative malware features that could allow to distinguish APT-belonging malware samples from other malware samples that are unrelated to APTs.

For that purpose, three different methods are used: two of them are filter methods (variance and χ^2 statistical test) and one is an embedded technique (tree-based estimator) (Yan et al., 2013). **Variance** is used as a simple baseline approach which assumes that features with lower variance are uninteresting. The other techniques select features that are expected to have a higher capacity of discrimination. The χ^2 **univariate statistical test** selects features that show a dependence relationship with the target category (i.e., the classification of malware as APT-related or generic).

The **tree-based estimator** computes impurity-based feature importances for classification. The application of these three techniques let us discard irrelevant features.

Finally, the relevance of the considered features is evaluated in terms of their accuracy for the classification of APT-related malware. For that purpose, the results obtained from 4 widely-used classification algorithms are used:

1. **Logistic regression**, which models the posterior probability of a class in terms of a linear function of the inputs by means of the logistic or sigmoid function.
2. **K-NN** (k-nearest neighbors), which is a non-parametric method used for classification. An object is classified by a majority vote of its neighbors, with the object being assigned to the most common class among its k nearest neighbors.
3. **Random Forest**. This is a simple and powerful ensemble method for classification. A random forest is an algorithm that uses the aggregation of many decision trees to reduce variance and control the overfitting of classification (Breiman, 2001). It creates several trees, which are grown through random selection of the input variables. The predicted class was obtained through averaging of the probabilistic prediction.
4. **SVM classifier** is a supervised learning model that aims at finding the hyperplane that maximizes the margin between classes while allowing for a minimal number of misclassifications (soft margin). The hyperplane is determined by a small subset of the data points known as support vectors (Cortes and Vapnik, 1995). It is usual to use radial basis function or polynomial kernels to map data to a higher dimensional space where they are more easily separable. However, since the number of features is relatively large in this case, a linear kernel has been chosen because it is expected to offer good performance with an easier optimization procedure (Hsu et al., 2003).

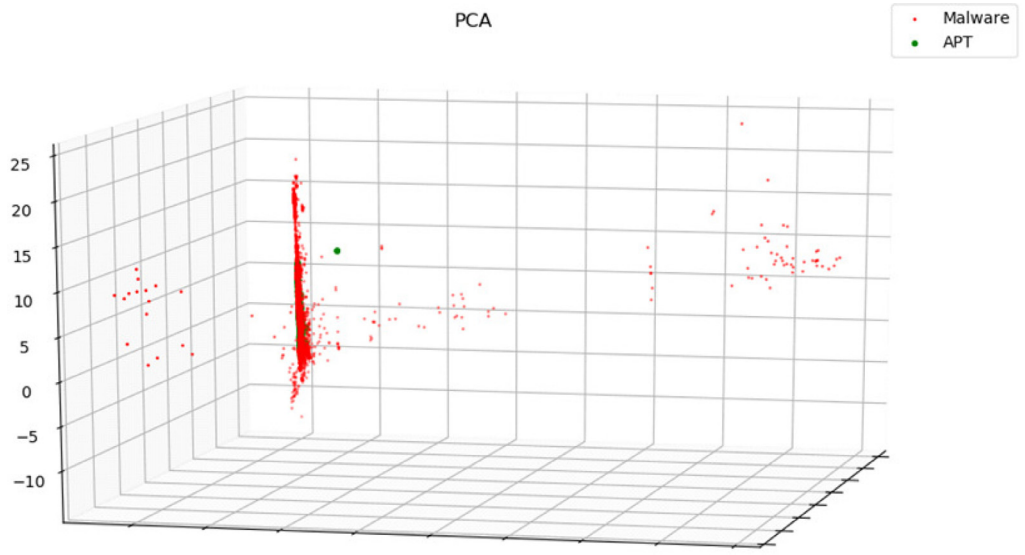
5. Experimental results

The experiments reported in this paper were performed using a Fujitsu Celsius M740 Server with Intel Xeon E5-1650 v3 at 3.70 GHz, 512GB HDD and 16GB of RAM memory. Data processing has been implemented using Python 3.5. For data acquisition, the PECore and Scapy[5] libraries have been used. A relational database over a MySQL 5.1 RDMS was designed to store data obtained from both static and dynamic analyses in a total of 105 tables. The scikit-learn library has been used for the implementation of the data analysis algorithms (Pedregosa et al., 2011).

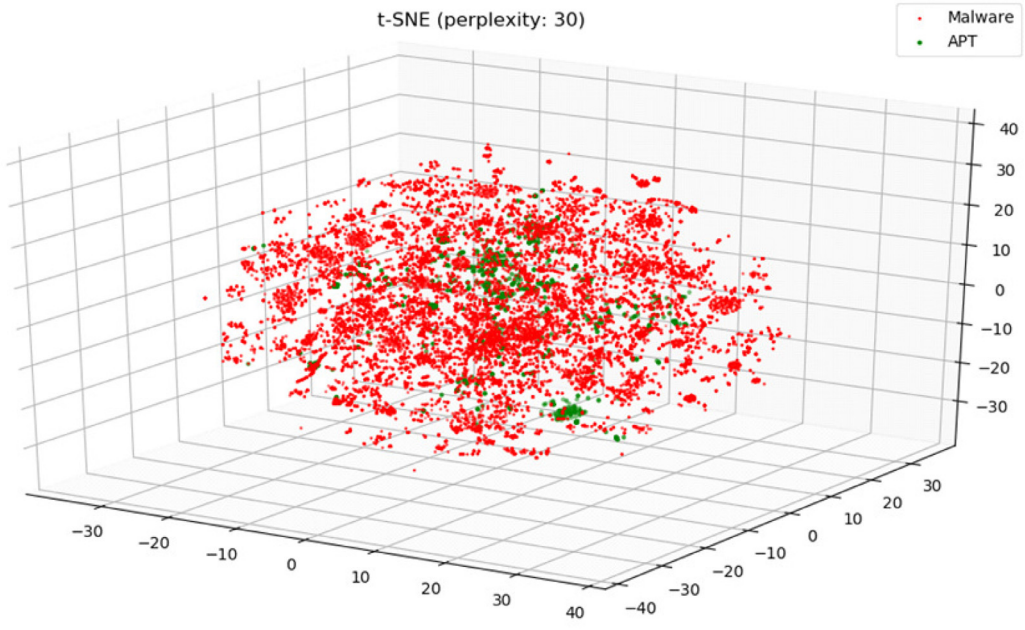
5.1. *Correlation analysis*

A correlation heatmap was obtained among the 1941 studied features. The strongest positive correlations were obtained among the features related to the services started by the malware and, in a smaller degree, among the imported functions and suspicious DLLs. Furthermore, lower correlations were also found between functions and DLLs and between services and files. This means that generally some services are started together by APT malware. These correlations could be interpreted by the fact that groups of services are often started together, by specific imported functions or by services that create or read specific files at the same time. There are specific behaviors inside malware that are common among malware,

[5] Biondi P (2002). Scapy, www.scapy.net.



(a) Projection of the three principal components



(b) t-SNE 3-D projection labelled with respect to being or not an APT sample

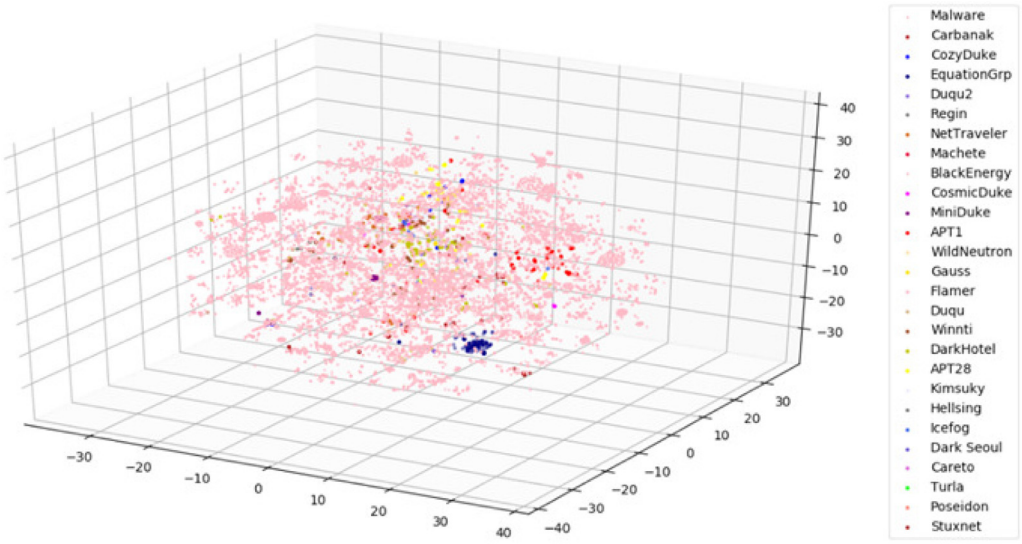
Fig. 1 – Dimensionality reduction of the dataset.

such as creating files for reading/writing, sending data over the network, checking the existence of antimalware software, etc. In order to achieve this, they need services like Cryptsvc used to encrypt communications or NetDDE used to communicate dynamic data over the network.

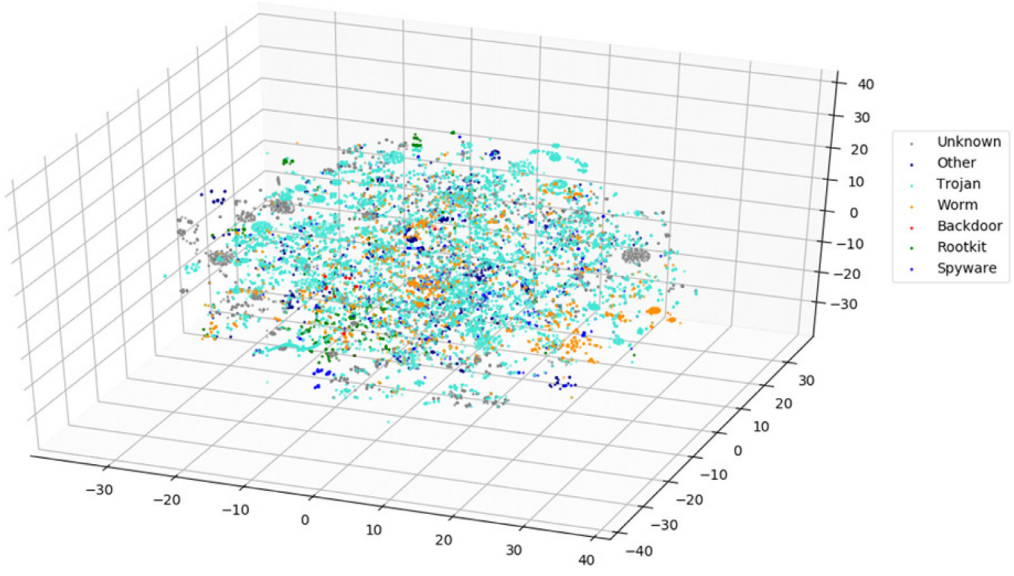
5.2. *Dimensionality reduction*

In order to obtain a visual representation of the dataset, dimensionality reduction of the original dataset onto a 3-dimensional space has been performed. To accomplish this task, both PCA and t-SNE have been applied. The PCA reduction to 3 components is reflected in Fig. 1a. The resulting decomposition retains 11,5% of the variance, consequently failing from a visualization perspective. Most instances, including those of APT-related malware, are projected in a dense group along an axis. The unsatisfactory results could be due to the strong linearity assumptions imposed by the algorithm.

The other alternative approach used for dimensionality reduction is the projection onto a 3-D space of the standardized original data by means of t-SNE. Figs. 1b and 2 show the results with the perplexity parameter set to 30 and the learning rate to 200, although their effect on the quality of the projection was not critical as we observed in a preliminary



(a) Colored according to APT campaigns



(b) Colored according the malware sample types

Fig. 2 – Visualization of the t-SNE 3D projection according to the malware developers and types.

parameter selection stage. The resulting nonlinear projection is more interpretable than that obtained from PCA decomposition. In Fig. 1b, projected data is labeled according to the known previous connection of the malware instance to APTs. Although the resulting projection shows only one cloud of points, APT-related malware instances are projected close to each other, suggesting that the proposed high-dimensional data set might contain knowledge that enables the detection of malware with similar characteristics to those already known to belong to APT campaigns. Indeed, the visualization of the projected data labeled according to the APT campaigns, which is depicted in Fig. 2a shows even more clearly that related malware instances tend to be projected close to each other, displaying some loose clusters of malware instances of the same campaign (APT1, Equation Group, etc.). The most evident example is the small group in the bottom right side of the cloud, composed of malware samples related to Equation Group. Fig. 2b shows the low-dimensional distribution of malware types.

5.3. *Feature selection*

As exposed in Section 4.2, three different methods were used to select the most representative features. The first method consists on removing the features with lower variance. To determine heuristically a variance threshold, the elbow method (Thwe et al., 2019) was used. Following this method, the selected threshold was determined to be 0.03. Those features

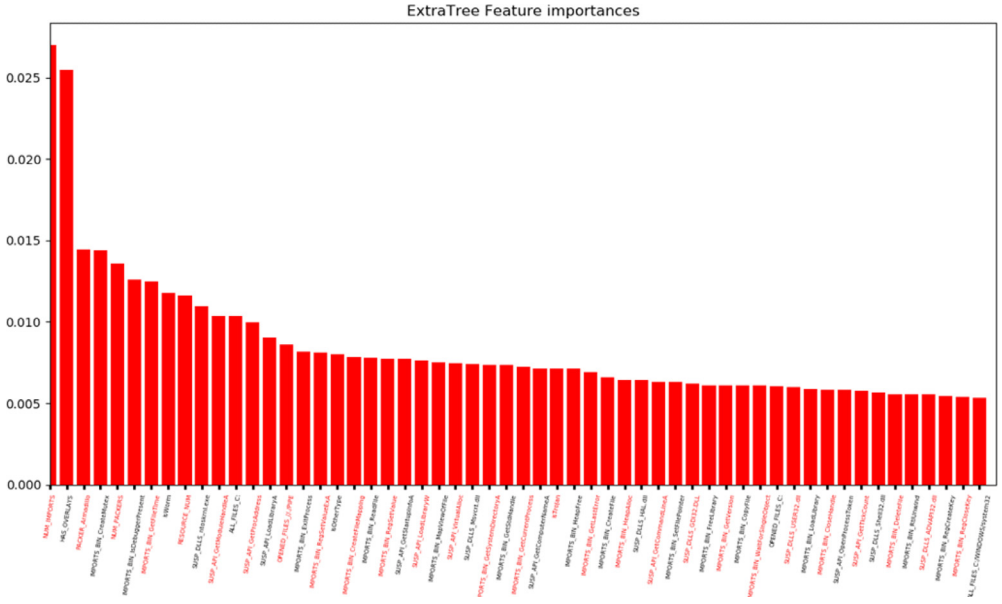


Fig. 3 – Extra Tree Feature Importances distribution.

with a training-set variance lower than this threshold were removed. This resulted in a reduced set of 237 features.

Secondly, a χ^2 test between the feature and the target was performed, selecting features with the highest χ^2 values. Using the same method, the feature threshold was determined to a χ^2 value of 400 and, as a result, only 20 features were selected.

In the third method, feature importances are computed by means of an ensemble of 10 randomized decision trees on various sub-samples of the dataset. The distribution of feature importances has been analyzed to determine the threshold. Again, the threshold was set through the elbow method to a value of 0.005 and, therefore, a total of 56 features were selected.

In short, the three aforementioned methods selected respectively 237, 21 and 56 features. The union of the three sets includes 238 features. The selected feature list is conformed mainly by two big sets of features (IMPORTS -83- and SUSPICIOUS APIs -76-). These two sets (that correspond with 21% of the original feature set) comprise two thirds (66%) of the 238 selected features. This result can be interpreted in the sense that there are some library functions with higher discriminatory power, i.e., that are more usually called in executable APT-related instances than in other generic malware instances. It is also interesting to mention that, from the 219 started services analyzed (11,3% of the total number of features), only four (a mere 1,8%) is selected to be an interesting feature to detect APT malware. This situation could be explained by the following hypotheses:

1. APTs tend to be as stealthy as possible, avoiding opening unnecessary services.
2. APTs are complex developments, so to avoid antimalware detection, they are intended to create their own services instead of using system related ones.
3. Services tend to be initialized in groups, and therefore, other services are ignored during feature selection because of their correlation.

Each of the aforementioned methods seems to highlight a different area of interest, as can be seen in Table 3, which enumerates the 10 most important features according to each of the 3 methods. The variance criterion highlights specific suspicious DLLs and APIs in the malware sample, while the χ^2 selection focuses on imports. The random trees select a more diverse set of features. It is important to note that the variance method does not select features according to their discriminatory power. It was used as a baseline filter method, easy to understand, with the aim to avoid discarding interesting features. The features obtained with these methods are different to the indicated previously in academia (Yonts, 2011, Laurenza et al., 2017, Yan et al., 2013, Sornil and Liangboonprakong, 2013). It must be noted that only the work by Laurenza et al. (2017) aims to discriminate APT-related samples from generic malware samples as in this work. However, their aim (triage) and scope (static features) is different and they do not focus on the analysis of the most discriminatory features. The other three studies are oriented to classify malware from benign executables, using features related to n-grams or static analyses that differ from the ones analyzed in this work.

A detailed analysis of the selected features that provides some insight about their nature might help to understand their usefulness for the APT selection process. Therefore, some of the most important features, listed in Table 3, are explained next.

IMPORTS_GetFileTime. A masquerading technique frequently used by malware is time stomping, i.e. the practice of modifying the creation and or modification time of the executable file (Esposito and Peterson, 2013). In order to perform this technique, both GetFileTime and SetFileTime functions need to be imported. Our hypothesis is that an expert APT

Table 3 – Most relevant features selected by the three selection methods.

	Variance	χ^2	ExtraTreesClassifier
1	SUSP_DLLS_ADVAPI32.dll	Resource_Number	HAS_OVERLAYS
2	SUSP_DLLS_GDI32.DLL	IMPORTS_GetFileTime	NUM_IMPORTS
3	SUSP_DLLS_SHELL32.DLL	IMPORTS_CreateMutex	IMPORTS_GetFileTime
4	OPENED_FILES_PIPE	PACKER_Armadillo	isWorm
5	IMPORTS_GetCurrentProcess	IMPORTS_RegSetValueExA	IMPORTS_CreateMutex
6	SUSP_API_GetModuleHandleA	IMPORTS_MapViewOfFiles	Resource_Number
7	OPENED_FILES_C:\	IMPORTS_RegSetValue	IMPORTS_IsDebuggerPresent
8	SUSP_API_WriteFile	IMPORTS_CreateFileMapping	SUSP_API_GetModuleHandleA
9	SUSP_API_GetModuleFileNameA	IMPORTS_RegCreateKey	NUM_PACKERS
10	SUSP_API_Sleep	IMPORTS_IsDebuggerPresent	PACKER_Armadillo

developer would remove as much information as possible from their samples, therefore changing as many timestamps as possible. Up to 26.4% of APT-related samples import this function whereas only 2.8% of non-APT malware uses it.

IMPORTS_ CreateMutex. Mutexes are used by all kind of executables to understand they already running in a different process. If the mutex, with a specific name, were already created, they would not execute themselves again. Once the executable ends, that mutex disappears. This imported function is used often by malware. Interestingly enough, APT samples seem to be more prone to use this function, although mutexes can be created in other ways. As per our analyses, 33% of APT-related samples use this imported function whereas only 5.2% of non-APT malware uses it.

HAS_OVERLAYS. Generally, malware samples will load suspicious code into memory from the overlay. Current anti-malware solutions usually detect suspicious code in overlays, marking the executable as suspicious. For this reason, APT malware samples are less prone to add overlays with executable code. In our dataset, about 30% of generic malware samples include overlays while only the 20% of APT-belonging malware use them.

NUM_IMPORTS. It seems that malware that belongs to an APT campaign tend to have a higher number of imported functions. Indeed, APT-belonging samples have an average of 26.3 imported functions vs. an average of 22 imports for generic non-APT-belonging malware.

Based on the variance, which does not necessarily select highly discriminative features, the most relevant features would be the use of *ADVAPI32.DLL and GDI32.DLL* Dynamic Link Libraries (DLLs). Generally speaking, DLLs are loaded on demand by executables when needed. From these DLL libraries, different functions are imported.*ADVAPI32.DLL* library allows using advanced functions related to the registry and services management.*GDI32.DLL* exports functions related to Graphical User Interfaces. Only 11% of the APT-belonging samples import any function from this DLL while 50% of generic malware samples load this DLL. A possible explanation is that APT-related malware tries to keep a higher level of stealthiness than generic malware samples avoiding GUIs for their functioning.

5.4. *Classification*

The goal of these experiments is to evaluate the ability of the selected features to classify whether a malware instance is related to an APT or not. This is interesting to analyze the discriminative ability of the proposed features for APT-related malware detection. All the instances in our dataset were used in these experiments, where we compared the well-known classification algorithms listed in Section 4.2 to obtain a global understanding of their capability of detecting APT-related samples among generic malware samples. For this binary classification process (i.e., APT-related malware or not), we consider true positives the malware instances that are correctly identified as APT-related malware.

The dataset was randomly split in a training and a testing set: two thirds of the dataset were used for training and one third for testing. We used 10-fold cross-validation (Zhang and Yang, 2015,Arlot and Celisse, 2010) on the training data set to set the hyperparameters, whose resulting selection is summarized in the following list:

1. *K-NN:* The number of neighbors was varied between 1 and 50 (we tried with 1,2,5,10,25 and 50 neighbors) and the best accuracy during cross-validation was obtained for k=2.
2. *Linear SVM:* The regularization parameter was chosen between 0.01 and 10, obtaining the best results for C=0.01 which also produced the fastest execution time.
3. *Random Forest:* It uses the Gini impurity as split criterion and 100 estimators, to balance the accuracy and the training time.
4. *Logistic Regression:* With tolerance 0.0001, regularization strength $C = 1.0$ and maximum number of iterations equal to 100.

The classifiers were trained with those parameterizations and applied to the test data set. The results obtained by the classifiers are reported in terms of several performance metrics: accuracy, precision, recall, F-measure and false negative rate. Furthermore, the receiver operating characteristic (ROC) curve and the area under this curve (AUC) are computed as a measure of diagnostic accuracy. Table 4 compares the results obtained for the binary classification experiment on the 238-dimensional data set for the different classification algorithms. The ROC Curves for each of the algorithms tested are shown in Fig. 4.

All algorithms provide acceptable accuracies, with areas under the ROC curve (AUC) that are clearly higher than 0.9. The results of random forest are superior to those obtained with other methods regarding both accuracy and AUC.

Table 4 – True & False Positives and negatives, Precision, Recall, F1, Accuracy and Area Under Curve (AUC) Values for each algorithm evaluated.

	TP	FP	TN	FN	FNR	Prec.	Recall	F1	Acc.	AUC	Training Time
Random Forest	440	30	5910	77	**0.1489**	**0.9361**	**0.8510**	**0.8915**	**0.9834**	**0.9955**	0.7390s
Linear SVM	378	43	5897	139	0.2688	0.8978	0.7311	0.8059	0.9718	0.9326	0.3387s
K-NN	376	81	5859	141	0.2727	0.822	0.7272	0.772	0.9656	0.9505	**0.1502s**
Logistic Regression	420	61	5879	97	0.1876	0.8731	0.8123	0.8416	0.9755	0.9855	0.1802s

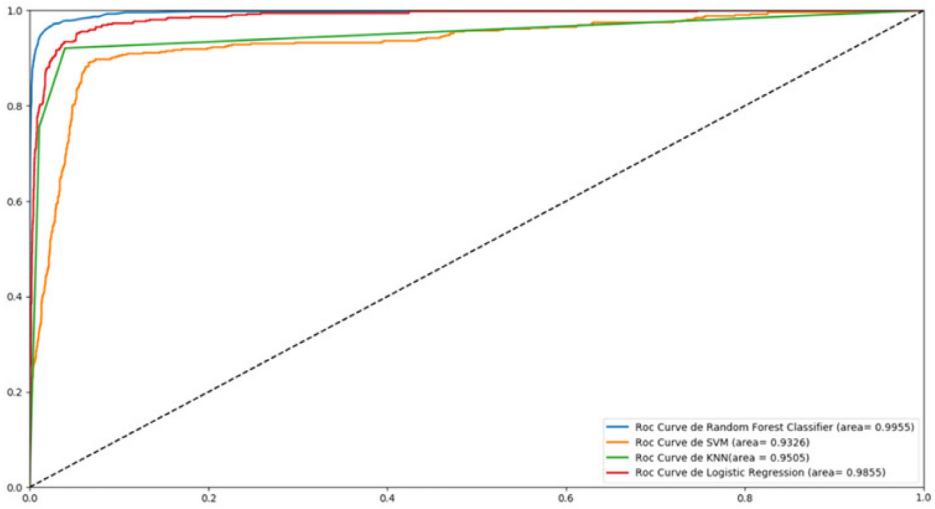


Fig. 4 – Area under the ROC curve for the different classifiers.

Accuracies of Random Forest are followed by Logistic Regression (with AUCs equal to 0.9957 and 0.9906, respectively).

To analyze again the discriminative power of the selected subset of features, their importances in random forest classification were also obtained. It is interesting to note that 8 out of the 10 highest importance scores correspond to features indicated in Table 3 as the most relevant ones selected by the χ^2 and ExtraTreeClassifier methods: *Resource_Number, Has_Overlays, Num_Imports, Imports_GetFileTime, Imports_CreateMutex, Packer_Armadillo, Imports_IsDebuggerPresent* and *Num_Packers*.

As a result, it can be argued that the proposed set of features is discriminative enough to determine whether a malware instance of the proposed dataset is related with an APT. However, it must be noted that the proposed experimentation assumes a fixed perspective of malware, considering that their characteristics do not evolve. On the contrary, it is reasonable to consider that the environment is non-stationary, due to the adversarial nature of malware. The characteristics of new APT-related malware will probably experience some drift, because they need to evolve as a response to advances in detection. As a result, we cannot claim that the obtained classification accuracies can be extrapolated to new unseen malware.

In any case, this is not the ultimate purpose of this work, which aims to acquire a better understanding about the features present in APT-related malware in order to help analysts find common trends in APT development. E.g., a non-exhaustive interpretation of the obtained results leads us to some findings that seem to support the belief that malware used in APT campaigns usually present a greater degree of sophistication than average malware. Specifically:

- APT-related samples tend to have fewer packers than generic malware. Since the presence of strange or multiple packers is one of the clearest evidences of the malicious nature of an executable, APT developers might have dedicated more efforts to avoid their use in order to appear as normal executables.
- Avoidance of reverse engineering is more commonly attempted in APT-related samples. They detect if they are being run in a sandboxed or virtual environment by means of imported functions such as **IsdebuggerPresent**.
- APT-related malware seems to have a low number of resources in the executables. Again, the reason behind this finding could be a particular caution from malware developers to avoid being detected by antimalware software.
- They also seem to request their file name and version with the import **GetFileTime** more regularly that generic malware, which could be explained due to a higher probability of regular updates of malware in the context of APT campaigns.

Another point to consider is the performance of classification in terms of computational resources. In the proposed experiments, model training times were acceptable, because the dataset was small enough to perform quickly. However, an operational system could eventually be trained with millions of samples. Random forest resulted to be the slowest

algorithm, whereas Logistic Regression provided a better balance between accuracy and computational cost. However, it is necessary to consider other problems with regard to the computational performance of the proposed analyses. The efforts to extract features from malware samples and their subsequent preprocessing have not been measured but should not be disregarded, because an on-line malware detector leveraging the aforementioned features for classification would need to extract continuously those features from new samples. Although static analysis can be obtained quickly, dynamic analysis in a short period of time can be a challenge. A possible approach to alleviate this problem could be the one proposed in Rhode et al. (2018), i.e., considering only a short window of execution under the assumption that malicious activity will appear rapidly. Furthermore, algorithms such as k-NN perform a more intensive computation in testing time than in training time.

Nevertheless, computational effort is not the only remaining obstacle for the application of the proposed approach to an operational on-line system for the detection of APT-related malware. It would also be necessary to conduct research in classification algorithms that are suited for incremental learning, to adapt to the expected concept drift.

6. Conclusions

This paper presents a data-driven analysis of malware related to advanced persistent threats (APT). For that purpose, in the first place, a dataset has been built from thousands of APT-related and other generic malwares. The features considered for this dataset, which can be obtained through static, dynamic or network traffic analysis, have been selected according to domain knowledge and a purpose of interpretability. As a result, a large dataset of 19,457 malware instances and 1941 features has been collected.

The contributed dataset has been used to study which features are more useful to distinguish APT-related malware among executable files that have already been labelled as malicious. The ultimate aim is to contribute to a better understanding about the features shared by APT-related malware that might help security analysts. For that purpose, several steps have been taken. First, an exploratory analysis has been conducted to obtain knowledge about the data structure. A nonlinear dimensionality reduction technique, t-SNE, has been used to project the high-dimensional space onto a 3-dimensional representation that can be visualized. The results show that groupings of malware according to their type (e.g., rootkit, worm or trojan) or their known relation with an APT campaign can be found and, therefore, the hypothesis that APT-related malware instances share some common features seems to hold. Furthermore, correlation analysis shows that some features (especially imported services) are strongly correlated, which denotes that some resources are used together in malware. Later, feature selection has been performed using different filter and embedded techniques to find features with high discriminative power. As a result, 238 features were selected to be used for the final classification experiments. A discussion is provided about the meaning and possible interpretation of some of the features that have higher discriminability according to the proposed selection techniques. Finally, the performance of four well-known binary classification techniques has been assessed on the reduced feature set data set.

The results provided by all the evaluated algorithms for the binary classification of APT-related malware in a dataset of malware instances showed high accuracies (>96%) and AUC. This means that the proposed set of features was discriminative enough to determine if a malware instance of the proposed dataset is related with an APT. It must be noted, however, that the proposed experimentation assumes a fixed perspective of malware, which only considers the characteristics found so far in APT campaigns. Since malware development continuously evolves to avoid detection, this environment is non-stationary and so the obtained classification accuracies cannot be extrapolated to future unseen malware. However, the results are still useful to find common trends in APT development, which support the belief that malware used in APT campaigns are usually more sophisticated than the average, exemplified by a particular stress on stealthiness. The small group of selected features can be used to classify newly found malware as susceptible to use in APT campaigns. Since APT attacks are usually long and persistent campaigns, the previous classification of the malware can help in detecting these attacks.

Future work should be aimed at exploiting the contributions of this work for the development of data-driven systems for the automatic detection of APT-related malware. For that purpose, it is necessary to address the technical challenges to provide efficient extraction of the proposed features from static and dynamic analyses. The inclusion of additional features must also be considered for this purpose, balancing both their potential discriminability and the difficulty of acquisition. In this sense, it seems reasonable to investigate the network traffic generated by APT-related malware more thoroughly. On the other hand, it is also essential to evaluate the generalization ability of the proposed approach. The assessment on a completely new dataset could lead to knowledge about trends in malware development. Finally, it would be interesting to conduct research on algorithms that enable incremental learning from new malware instances.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Luis Francisco Martín Liras: Conceptualization, Investigation, Software, Data curation, Writing - original draft. **Adolfo Rodríguez de Soto:** Conceptualization, Methodology, Supervision, Resources. **Miguel A. Prada:** Methodology, Supervision, Writing - review & editing, Resources, Validation, Formal analysis.

Acknowledgements

This work would not be possible without the help, funding and support of various entities. We wish to thank to Scayle (Supercomputación de Castilla y León) for their funding and support. We wish also to thank VirusShare for assisting the malware research community and provide us with a big number of malware samples. VirusTotal was kind to assist us in this research experiment and provided the dynamic analysis reports used during the research process. ContagioDump, led by Mila Parkour, and TheZoo, owned by Yuval Nativ, provided most of the APT samples that have been analyzed. Finally we would like to thank the reviewers for their thoughtful comments and efforts, which have aided to substantially improve our manuscript.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cose.2021.102202.

REFERENCES

Ács -Kurucz G, Balázs Z, Bencsáth B, Buttyán L, Kamarás R, Molnár G, Vaspöri G. In: Technical Report. An independent test of APT attack detection appliances. MRG Effitas and CrySyS Lab; 2014.

Alazab M, Venkataraman S, Watters P. Towards Understanding Malware Behaviour by the Extraction of API Calls. In: 2010 Second Cybercrime and Trustworthy Computing Workshop. IEEE; 2010. p. 52–9. doi:10.1109/CTC.2010.8.

Anderson HS, Roth P. EMBER: An open dataset for training static PE malware machine learning models. CoRR 2018.

Arlot S, Celisse A. A survey of cross-validation procedures for model selection. Stat. Surv. 2010;4:40–79. doi:10.1214/09-SS054.

Bayer U. Large-Scale Dynamic Malware Analysis. TU Wien; 2009.

Bhatt P, Yano ET, Gustavsson P. Towards a Framework to Detect Multi-stage Advanced Persistent Threats Attacks. In: 2014 IEEE 8th International Symposium on Service Oriented System Engineering. IEEE; 2014. p. 390–5. doi:10.1109/SOSE.2014.53.

Breiman L. Random forests. Mach. Learn. 2001;45(1):5–32. doi:10.1023/A:1010933404324.

Brogi G. Real-time detection of Advanced Persistent Threats using Information Flow Tracking and Hidden Markov Models. Conservatoire National des Arts et Métiers; 2018.

Chen P, Desmet L, Huygens C. A study on advanced persistent threats. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag; 2014. p. 63–72. doi:10.1007/978-3-662-44885-4-5.

Chia-Mei Chen D-WW, Lai GH. Evolution of advanced persistent threat (apt) attacks and actors. ICS 2018: New Trends in Computer Technologies and Applications 2018;1013:76–81.

Cortes C, Vapnik V. Support-Vector networks. Mach. Learn. 1995;20(3):273–97. doi:10.1023/A:1022627411411.

Daly MK. The Advanced Persistent Threat (or Informationized Force Operations). 23rd Annual Large Installation System Administration Conference, 2009.

Damodaran A, Di Troia F, Visaggio CA, Austin TH, Stamp M. A comparison of static, dynamic, and hybrid analysis for malware detection. Journal of Computer Virology and Hacking Techniques 2017;13(1):1–12.

David OE, Netanyahu NS. DeepSign: Deep learning for automatic malware signature generation and classification. In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE; 2015. p. 1–8. doi:10.1109/IJCNN.2015.7280815.

Esposito S, Peterson G. Creating Super Timelines in Windows Investigations. In: Peterson G, Shenoi S, editors. In: Advances in Digital Forensics IX. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 135–44. doi:10.1007/978-3-642-41148-9_9.

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. doi:10.1016/J.COSE.2014.09.006.

Gandotra E, Bansal D, Sofat S. Malware analysis and classification: A Survey. Journal of Information Security 2014;05(02):56–64. doi:10.4236/jis.2014.52006.

Giura P, Wang W. A Context-Based Detection Framework for Advanced Persistent Threats. In: 2012 International Conference on Cyber Security. IEEE; 2012. p. 69–74. doi:10.1109/CyberSecurity.2012.16.

Hsu C-W, Chang C-C, Lin C-J. In: Technical Report. A Practical Guide to Support Vector Classification. National Taiwan University; 2003.

Kolosnjaji B, Zarras A, Webster G, Eckert C. Deep learning for classification of malware system call sequences. In: Kang BH, Bai Q, editors. In: AI 2016: Advances in Artificial Intelligence. Cham: Springer International Publishing; 2016. p. 137–49. doi:10.1007/978-3-319-50127-7_11.

Lai A, Wu B, Chiu J. In: Defcon 19. Balancing the PWN trade deficit series: APT secrets in Asia; 2011.

Lamprakis P, Dargenio R, Gugelmann D, Lenders V, Happe M, Vanbever L. Unsupervised Detection of APT C&C Channels using Web Request Graphs. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Springer International Publishing; 2017. p. 366–87. doi:10.1007/978-3-319-60876-1_17.

Laurenza G, Aniello L, Lazzeretti R, Baldoni R. Malware triage based on static features and public APT reports. In: Dolev S, Lodha S, editors. Cyber Security Cryptography and Machine Learning. Cham: Springer International Publishing; 2017. p. 288–305.

Li F, Lai A, Ddl D. Evidence of Advanced Persistent Threat: A case study of malware for political espionage. In: 2011 6th International Conference on Malicious and Unwanted Software. IEEE; 2011. p. 102–9. doi:10.1109/MALWARE.2011.6112333.

Martin-Liras, L., Rodriguez de Soto, A., Prada, M.A., 2020. Static and dynamic analysis of both generic and APT-related malware. 10.17632/w2w8gjsgnt.1

Mulder S, Blunt J, Tauritz D. Adaptive rule-Based malware detection employing learning classifier systems: A Proof of concept.. Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual 2011. doi:10.1109/COMPSACW.2011.28.

Nassar M, Safa H. Throttling malware families in 2D. CoRR 2019.

Nicho M. Identifying vulnerabilities of advanced persistent threats: an organizational perspective. International Journal of Information Security and Privacy (IJISP) 2014;8:18. doi:10.4018/ijisp.2014010101.

Niu W, Zhang X, Yang G, Zhu J, Ren Z. Identifying APT malware domain based on mobile DNS logging. Mathematical Problems in Engineering 2017;2017:1–9. doi:10.1155/2017/4916953.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: machine learning in python. Journal of Machine Learning Research 2011;12:2825–30.

Pei K, Gu Z, Saltaformaggio B, Ma S, Wang F, Zhang Z, Si L, Zhang X, Xu D. Hercule: Attack Story Reconstruction via

Community Discovery on Correlated Log Graph.. In: Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC '16. New York, New York, USA: ACM Press; 2016. p. 583–95. doi:10.1145/2991079.2991122.
Rhode M, Burnap P, Jones K. Early-stage malware prediction using recurrent neural networks. Computers & Security 2018;77:578–94. doi:10.1016/j.cose.2018.05.010.
Ronen R, Radu M, Feuerstein C, Yom-Tov E, Ahmadi M. Microsoft malware classification challenge. CoRR 2018.
Ruttenberg B, Miles C, Kellogg L, Notani V, Howard M, LeDoux C, Lakhotia A, Pfeffer A. Identifying shared software components to support malware forensics. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag; 2014. p. 21–40. doi:10.1007/978-3-319-08509-8-2.
Sexton J, Storlie C, Anderson B. Subroutine based detection of APT malware. Journal of Computer Virology and Hacking Techniques 2016;12(4):225–33. doi:10.1007/s11416-015-0258-7.
Shafiq MZ, Khayam SA, Farooq M. Embedded malware detection using Markov n-Grams, 5137 LNCS; 2008. p. 88–107. doi:10.1007/978-3-540-70542-0-5.
Shlens J. A tutorial on principal component analysis. Google Research 2014.
Siddiqui S, Khan MS, Ferens K, Kinsner W. Detecting Advanced Persistent Threats using Fractal Dimension based Machine Learning Classification. In: Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics - IWSPA '16. New York, New York, USA: ACM Press; 2016. p. 64–9. doi:10.1145/2875475.2875484.
Sornil O, Liangboonprakong C. Malware classification using N-grams sequential pattern features. International Journal of Information Processing and Management 2013;July(4(5)):59–67. doi:10.4156/ijipm.vol4.issue5.7.
Su Y, Li M, Tang C, Shen R. A Framework of APT Detection Based on Dynamic Analysis. Proceedings of the 2015 4th National Conference on Electrical, Electronics and Computer Engineering. Paris, France: Atlantis Press, 2016.
Tankard C. Advanced persistent threats and how to monitor and deter them. Network Security 2011;2011(8):16–19. doi:10.1016/S1353-4858(11)70086-1.
Thwe YM, Ogawa M, Dung PN. Applying Clustering Techniques for Refining Large Data Set: Case Study on Malware. In: 2019 International Conference on Advanced Information Technologies (ICAIT). IEEE; 2019. p. 238–43. doi:10.1109/AITC.2019.8921088.
Trunk GV. A problem of dimensionality: A Simple example. IEEE Trans Pattern Anal Mach Intell 1979;PAMI-1(3):306–7. doi:10.1109/TPAMI.1979.4766926.
Ussath M, Jaeger D, Cheng F, Meinel C. Advanced persistent threats: Behind the scenes. In: 2016 Annual Conference on Information Science and Systems (CISS). IEEE; 2016. p. 181–6. doi:10.1109/CISS.2016.7460498.
Van Der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research 2008;9:2579–605.
Vance A. Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing. In: 2014 First International Scientific-Practical Conference Problems of Infocommunications Science and Technology. IEEE; 2014. p. 173–6. doi:10.1109/INFOCOMMST.2014.6992342.
Vasilescu M, Gheorghe L, Tapus N. Practical malware analysis based on sandboxing. Proceedings - RoEduNet IEEE International Conference. IEEE Computer Society, 2014.
de Vries J, Hoogstraaten H, van den Berg J, Daskapan S. Systems for Detecting Advanced Persistent Threats: A Development Roadmap Using Intelligent Data Analysis. In: 2012 International Conference on Cyber Security. IEEE; 2012. p. 54–61. doi:10.1109/CyberSecurity.2012.14.
Vukalovic J, Delija D. Advanced Persistent Threats - detection and defense. In: 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE; 2015. p. 1324–30. doi:10.1109/MIPRO.2015.7160480.
Wang C, Pang J, Zhao R, Fu W, Liu X. Malware Detection Based on Suspicious Behavior Identification. In: 2009 First International Workshop on Education Technology and Computer Science. IEEE; 2009. p. 198–202. doi:10.1109/ETCS.2009.306.
Yan G, Brown N, Kong D. Exploring Discriminatory Features for Automated Malware Classification. In: Rieck K, Stewin P, Seifert J-P, editors. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013. p. 41–61. doi:10.1007/978-3-642-39235-1_3.
Yonts J. In: Technical Report. Attributes of Malicious Files. SANS; 2011.
Zhang Y, Yang Y. Cross-validation for selecting a model selection procedure. J Econom 2015. doi:10.1016/j.jeconom.2015.02.006.
Zhao G, Xu K, Xu L, Wu B. Detecting APT malware infections based on malicious DNS and traffic analysis. IEEE Access 2015;3:1132–42. doi:10.1109/ACCESS.2015.2458581.

Luis Martin Liras is a Predoctoral student with a wide experience in the fields of Cibersecurity and Computer Science. He graduated on 2002, obtained a master degree of Computer Security in 2003. Since then he have been working for a wide number of cybersecurity related companies. He's is currently working for Hewlett Packard (HP) as a member of the Security team for HP printers firmware development. He has been also working for three years as a lecturer of Computer Science in the Mechanical Engineering, Computer Science and Aerospace Department at University of León, Spain.

Adolfo Rodrıguez de Soto is an associate Professor of Computer Science in the Mechanical Engineering, Computer Science and Aerospace Department at University of León, Spain. He completed his degree in Mathematics with the specialization in Computer Science at the University of Salamanca and the Complutense University of Madrid. He gets his PhD at Polytechnic University of Madrid in 1995. His research work is framed in the area of Artificial Intelligence, being author and co-author of more than 60 peer review national and international publications. Its main lines of research are Fuzzy Set Theory and Machine Learning.

Miguel A. Prada is an associate professor at the Department of Electric and Systems Engineering of the University of León (Spain). His research activity is mainly devoted to machine learning and data-driven modelling of engineering problems, generally focusing on unsupervised learning and anomaly detection techniques. He is also interested in cybersecurity of industrial control systems. He has participated in several research projects, with private and public funding.