Building malware classificators usable by State security agencies

Construcción de clasificadores de malware para agencias de seguridad del Estado

David Esteban Useche-Peláez
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
david.useche@mail.escuelaing.edu.co

Daniela Sepúlveda-Alzate
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
daniela.sepulveda@mail.escuelaing.edu.co

Daniel Orlando Díaz-López
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
daniel.diaz@escuelaing.edu.co

Diego Edison Cabuya-Padilla
Comando Conjunto Cibernético
Bogotá, Colombia
diego.cabuya@ccoc.mil.co

Resumen– El sandboxing ha sido usado de manera regular para analizar muestras de *software* y determinar si estas contienen propiedades o comportamientos sospechosos. A pesar de que el sandboxing es una técnica poderosa para desarrollar análisis de malware, esta requiere que un analista de malware desarrolle un análisis riguroso de los resultados para determinar la naturaleza de la muestra: goodware o malware. Este artículo propone dos modelos de aprendizaje automáticos capaces de clasificar muestras con base a un análisis de firmas o permisos extraídos por medio de Cuckoo sandbox, Androguard y VirusTotal. En este artículo también se presenta una propuesta de arquitectura de centinela IoT que protege dispositivos IoT, usando uno de los modelos de aprendizaje automáticos desarrollados anteriormente. Finalmente, diferentes enfoques y perspectivas acerca del uso de sandboxing y aprendizaje automático por parte de agencias de seguridad del Estado también son aportados.

Palabras claves– Cuckoo sandbox, ciencia de datos, aprendizaje de máquina, análisis de malware, sandboxing.

Abstract– Sandboxing has been used regularly to analyze software samples and determine if these contain suspicious properties or behaviors. Even if sandboxing is a powerful technique to perform malware analysis, it requires that a malware analyst performs a rigorous analysis of the results to determine the nature of the sample: goodware or malware. This paper proposes two machine learning models able to classify samples based on signatures and permissions obtained through Cuckoo sandbox, Androguard and VirusTotal. The developed models are also tested obtaining an acceptable percentage of correctly classified samples, being in this way useful tools for a malware analyst. A proposal of architecture for an IoT sentinel that uses one of the developed machine learning model is also showed. Finally, different approaches, perspectives, and challenges about the use of sandboxing and machine learning by security teams in State security agencies are also shared.

Keywords– Cuckoo sandbox, data science, machine learning, malware analysis, sandboxing.

1. INTRODUCTION

Cyber-attacks currently are not only being aimed against conventional computers, but also against mobile devices and other devices that are part of the Internet of Things like smart TVs or smart watches. According to a study conducted by the security company Kaspersky [1], every second there are 250 new malicious files against users of computers and mobile devices in Latin America and during the last year, there were 1188 million malware attacks approximately repelled by Kaspersky. This situation increases the work of malware analysts who have the task of determining behavior patterns, properties, and indicators that allow to identify and characterize a malware to prevent future infection incidents.

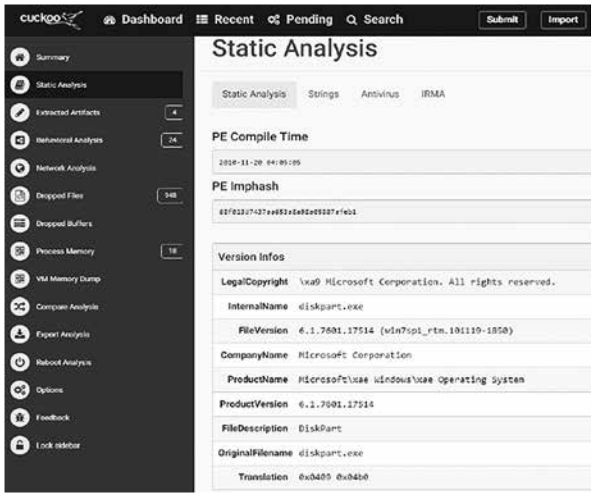
Malware or goodware generally comes as a PE (Portable Executable) file. A PE file contains all information necessary for the installation of a program, which includes imports or exports, compiler that was used (gcc, gcc+, javac, etc.), compilation time zone (e.g. Mon Dec 20 09:03:08 2010 compilation time for WannaCry), digital certificate information, architecture for which the program was designed (32 bits or 64 bits), imphash (Hash generated from exports or imports), resources that use the PE (e.g. The ransomware Cerber used a resource to ASCII Text, another to zip folder.), hashes (MD5 or SHA) associated with the program, developer company signature, installer size, software version, strings, among others. This information is useful to infer suspicious features or behaviors like operating system modules that are being invoked and that can allow access to the kernel, or some not common functions being imported. A PE program can have extensions: exe, dll, efi, acm, amongst others. On the other side, an ELF (Executable and Linkable Format) program, which is the equivalent of PE for Linux systems, can have extensions: bin, so, elf, mod, among others. Malware analysis can be performed through static or dynamic analysis of PE files. In static analysis, information is obtained from the PE or ELF file using different reverse engineering methods such as decompiles, disassemblers, extract of hashes, etc. In dynamic analysis, the PE or ELF file is executed, and its behavior is obtained and analyzed such as changes in the registry keys (deleted, modified or created keys), network behavior (IPs or domains establishing communication) or changes in the file system (deleted, modified or dropped files).

One technique used by malware analyst is sandboxing [2]resulting in tens of billions of dollars in economic damages each year. Among security professionals, the skills required to quickly analyze and assess these attacks are in high demand. Practical Malware Analysis provides a rapid introduction to the tools and methods used to dissect malicious software (malware which allow determining if a software sample contains properties, through a static analysis, or behavior, through a dynamic analysis, that can be considered suspicious. In a sandbox is possible to create controlled testing environments which are isolated from the host operating system, each one of them having its own disk space and memory. A testing environment must not access any resource that has not been specifically assigned to it. Virtual machines are the mechanism to deploy controlled testing environments with a different operative system over which is possible to test suspicious programs and resources without compromising the host.

An example of malware analysis using sandboxing can be seen with WannaCry. WannaCry was a ransomware, which affected many Windows computers and servers last year through the exploitation of an SMB 1 vulnerability, which is a protocol responsible for the communication of Windows computers on a network [3]. WannaCry encrypts information stored in a victim pc and requests a payment that oscillates between 200 to 600 bitcoins to get back access to the files. Through a sandbox and specifically a static analysis of a sample of WannaCry is possible to find that it uses a suspicious function called CryptDestroyKey. The CryptDestroyKey function is used to destroys the key hindering the decryption process.

Another information that could be obtained from WannaCry is the Callback TLS (Thread Local Storage), which are pieces of code that are executed before the Entrypoint, i.e. the sample starting execution point, and that many times are not reviewed by a debug system making them exploitable by malware developers which inject malicious source code into those spaces. Fig. 1 shows the information obtained from a static analysis for a Wannacry sample run in a sandbox, which includes Imphash, sha, compiler, compilation time, among other details.

Fig. 1. STATIC ANALYSIS OF WANNACRY WITH A SANDBOX



Static Analysis

PE Compile Time

2010-11-20 04:05:05

PE Imphash

[illegible]

Version Infos

LegalCopyright	\xa9 Microsoft Corporation. All rights reserved.
InternalName	diskpart.exe
FileVersion	6.1.7601.17514 (win7sp1_rtm.101119-1850)
CompanyName	Microsoft Corporation
ProductName	Microsoft\xae Windows\xae Operating System
ProductVersion	6.1.7601.17514
FileDescription	DiskPart
OriginalFilename	diskpart.exe
Translation	0x0409 0x04b0

Source: The authors.

There are some notable researches that integrate sandboxing with machine learning seeking to potentiate security solutions through interdisciplinarity. One related work is presented in [5] which proposes the use of machine learning classification methods for malware detection using sandbox, behavioral analysis and injection of instrumentation code. Also, [6]however, it\\nconsistently fails to detect new malware. Supervised machine learning\\nhas been adopted to solve this issue. There are two types of features\\nthat supervised malware detectors use: (i proposes the use of supervised machine learning methods to detect malware through the identification of static (extracted without running the sample) and dynamic (require an execution) characteristics. Similarly, [7]custom code bases and in-memory execution. Our hypothesis is that we can produce a high degree of accuracy in distinguishing malicious from trusted samples using Machine Learning with features derived from the inescapable footprint left behind on a computer system during execution. This includes CPU, RAM, Swap use and network traffic at a count level of bytes and packets. These features are continuous and allow us to be more flexible with the classification of samples than discrete features such as API calls (which can also be obfuscated proposes the use of machine activity metrics to automatically identify trustworthy and reliable portable executable (PE) samples, through classification methods that use self-organized feature maps creating unsupervised clusters of similar behavior. On the other hand, Donaldson SE et al. [8] exposes some next-generation cybersecurity axioms and sub-axioms for cybersecurity teams and state agencies which can help them to be more effective against attackers. These axioms states that i) Companies must look at themselves from the perspective of the attacker and design defenses accordingly, ii) Defenses must be designed to detect and delay attacks, so that defenders have time to respond, iii) Layers of defense are required to contain attacks and redundancy in protection and iv) Use of active defense to trap and repel attacks after they start, but before they can succeed.

On the other hand, regarding security proposals for IoT ecosystems, an architecture for an IoT sentinel that protects IoT devices is presented in [4]. This IoT sentinel is capable of identify devices and its types from the network it is connected. The sentinel also reduces potential damage of vulnerable devices by constraining their communications through specific network rules. The identification of the devices is done using their network traffic fingerprint, which is used for anomaly detection using a machine learning model to detect unusual behavior on the network.

The paper at hand is composed as follows. Section 2 presents malware analysis using the Cuckoo sandbox and introduce data science. Then, Section 3 proposes a mechanism to apply data science as support for malware classification and develops two machine learning models that classify malware for desktop and mobile operative systems. Later, Section 3 proposes an architecture for securing IoT ecosystems using a set of rings one of them based on one of the developed machine learning models. Next, Section 4 makes a reflection regarding the use of sandbox and machine learning by state security agencies. Finally, some conclusions and future works are included.

2. CUCKOO SANDBOXING AND DATA SCIENCE

Sandboxing allows performing static and dynamic analysis of samples, enabling a deeper understanding of malware and its variants. This section presents Cuckoo sandbox as a solution to perform malware analysis and introduce data science.

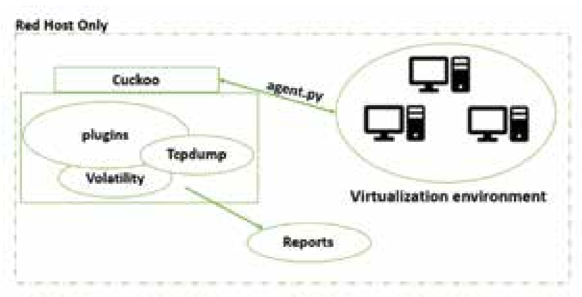
2.1 Cuckoo sandboxing

Cuckoo was born during the Google's Summer of Code of 2010 as part of The Honeynet Project. Claudio Guarnieri is the founder of the project who performs as a researcher in the fields of computer security. Claudio Guarnieri has researched botnets and directed attacks. He has also been a speaker at Hack In The Box, Black Hat, and Chaos Communication Congress. Cuckoo is an open source sandbox, which has an active community in charge of its development and a repository which allow accessing its source code. Cuckoo allows multiple input formats however in the research developed in this paper will be a focus in samples of the type PE and ELF[9].

Fig. 2 shows the topology of Cuckoo sandboxing. The cuckoo server hosts a virtualization environment through tools such as VMWare or VirtualBox and at least one virtual machine. Virtual machines are snapshots of an operative system with an agent.py file installed. The agent.py file is the way by which the Cuckoo server communicates with each of the virtual machines to send samples to be analyzed. Cuckoo server and the virtual machines must compose a Host-Only Network, i.e. a private network that only allows the connection of a cuckoo server with each of the virtual machines.

To develop malware analysis Cuckoo has several tools or plugins, like Volatility[1] [10] and Tcpdump [11]. Volatility is used to perform forensic analysis of the memory ram of virtual machines, on the other side Tcpdump is a sniffer that allows monitoring the traffic of the virtual machine.

Fig. 2. CUCKOO SANDBOX TOPOLOGY



Source: The authors.

Static and dynamic analysis can be carried out by Cuckoo. As a result of static and dynamic analyzes a series of malware analysis reports are generated, which are used to generate signatures that represent characteristics or patterns of a sample. Cuckoo includes a pool of signatures by default and allow to incorporate new ones because is an open source project [12].

Fig. 3 shows a Cuckoo signature which contains a description and an associated severity: low (1), normal (2) and high (3). This signature reports the creation of a window executable on some location of the filesystem, and in the same way, signatures can report another suspicious behavior like the shutdown of an operative system firewall service or the modification of critical operative system files. Cuckoo identifies which of the signatures are present in a sample and include them as matched signatures in the report.

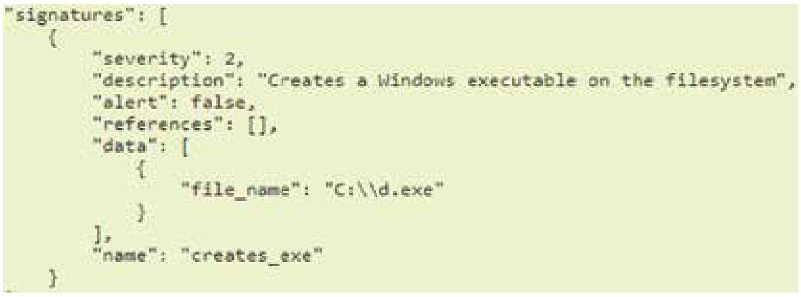
Cuckoo Sandbox offers the following sections accessible after a malware analysis is done (Fig. 4):

- Summary: It includes the most relevant features of the analyzed malware, like hashes, size, imphashes, file type, among others.
- Static Analysis: It contains sections (e.g. .data, .text, etc.), strings, file size, compilation time, amongst others elements, obtained from a decompilation and static analysis of the sample.
- Extracted Artifacts: It indicates what files were extracted successfully from the sample.
- Behavioral Analysis: It contains details of the file system, registry keys, process tree, and process content, amongst others elements, obtained when the sample is executed in a virtual machine.
- Network Analysis: It contains communications managed by the sample through protocols such as DNS, HTTP, UDP, ICMP etc.
- Dropped Files: It shows the files that were created or downloaded by the sample.
- VM Memory Dump: It contains the result of the virtual machine memory dump when the sample is being executed, making possible to obtain some useful information from the sample like the registry keys values.
- Reboot Analysis: Analysis of the status of the virtual machine where the sample was executed after a restart of the virtual machine.
- Dropped Buffers: It shows the buffers that were created when the sample was executed.
- Process Memory: It contains information about the processes created in the volatile memory by the sample.
- Compare Analysis: Information provided by Cuckoo when two sample analysis are compared.
- Export Analysis: It offers the option of exporting the sample report to JSON (JavaScript Object Notation) format.

Fig. 5 describe the general steps of a malware analysis using Cuckoo Sandbox. First, a sample is submitted to Cuckoo sandbox selecting, between all available Cuckoo virtual machines, the one over which the sample must be executed according to the sample extension type. Second, Cuckoo receives the sample and performs different analyzes (static and dynamic) from the application of reversing techniques and the installation and the execution of the sample in the selected virtual machine. Finally, Cuckoo generates reports which help the analyst to determine if the sample is goodware or malware.

[1] https://www.volatilityfoundation.org/

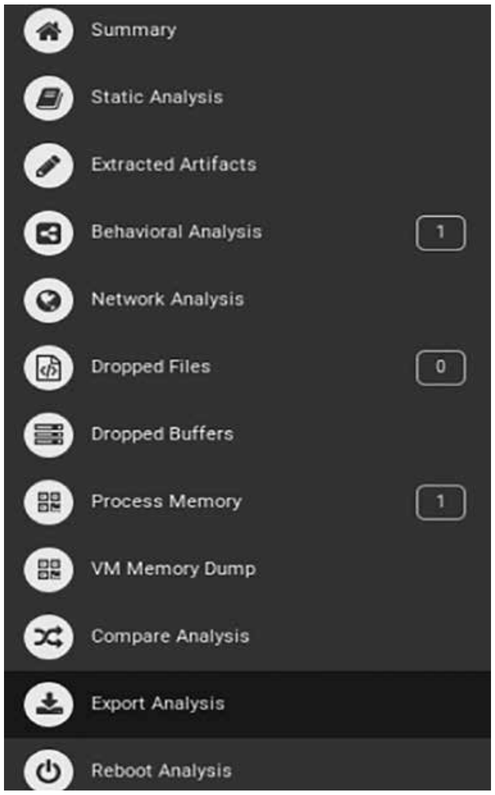
Fig. 3. CUCKOO SIGNATURE



```
"signatures": [
    {
        "severity": 2,
        "description": "Creates a Windows executable on the filesystem",
        "alert": false,
        "references": [],
        "data": [
            {
                "file_name": "C:\\d.exe"
            }
        ],
        "name": "creates_exe"
    }
```

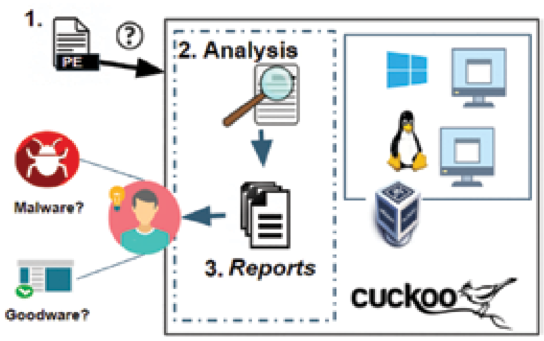
Source: http://jamu.info/pwnypot/docs/customization/signatures.html

Fig. 4. SECTIONS DISPLAYED BY CUCKOO



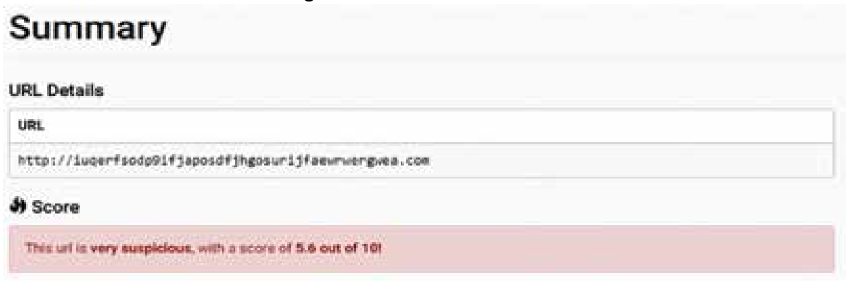
Source: The authors.

Fig. 5. MALWARE ANALYSIS IN CUCKOO SANDBOX



Source: The authors.

Fig. 6. WANNACRY URL ANALYSIS



Summary

URL Details

URL
http://iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

Score

This url is **very suspicious**, with a score of **5.6 out of 10!**

Source: The authors.

Cuckoo also allows analyzing URLs to identify suspicious network behavior i.e. if after accessing an URL some malicious file is downloaded or some strange activity is evidenced. Fig. 6 shows a fragment of a report generated by Cuckoo which includes the score of how suspicious is a WannaCry URL. Fig. 7 shows all communications toward different domains identified by Cuckoo for the URL mentioned in Fig 6. URL analysis allows to verify communications with domains considered as malicious, or consumption of resources from domains that are considered clean now of access but that resolves toward an IP that in the past was involved with the malicious activity. A domain name that changes IPs frequently or an IP that modify domain names recurrently are suspicious findings that can be obtained from an URL analysis. This information must be reviewed by a malware analyst to classify a sample as goodware or malware.

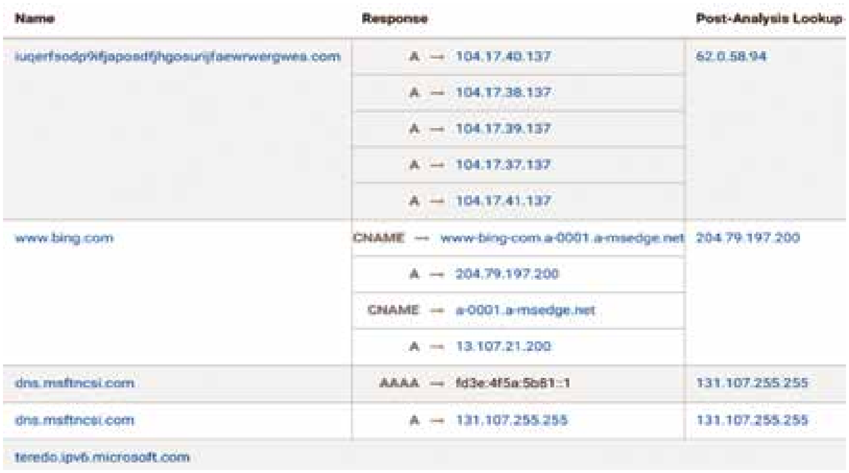
2.2 Data science

Last years the amount of available data has increased markedly, so traditional analytic, based mainly in statistics, modelers and manual analysis, has been overpassed by the volume and diversity of the data to analyze [13] Also, it is fundamental to not only extract information from that data, it has to describe more than the training set, it has to generalize those attributes to the possible data the model will receive [14].

In general, data science is the discipline of using quantitative methods from traditional analysis fields, like statistics and mathematics and combine them with modern concepts to create algorithms that analyze the data to discover patterns that allow to generalize over all the possible data, and to make predictions that solve different kind of problems [15].

An important key component in data science projects is the development of a data science model which can works with an enough efficiency. The selection and tuning of a data science model is a task that requires some knowledge about the application of machine learning models and its mathematical fundamentals, so they can be adjust properly.

Fig. 7. URL NETWORK ANALYSIS BY CUCKOO



Name	Response	Post-Analysis Lookup
iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	A → 104.17.40.137	62.0.58.94
	A → 104.17.38.137	
	A → 104.17.39.137	
	A → 104.17.37.137	
	A → 104.17.41.137	
www.bing.com	CNAME → www-bing-com.a-0001.a-msedge.net	204.79.197.200
	A → 204.79.197.200	
	CNAME → a-0001.a-msedge.net	
	A → 13.107.21.200	
dns.msftncsi.com	AAAA → fd3e:4f5a:5b81::1	131.107.255.255
dns.msftncsi.com	A → 131.107.255.255	131.107.255.255
teredo.ipv6.microsoft.com		

Source: The authors.

A wide understanding of the problem to be solved is also an important aspect that the members of a data science project must consider, as it helps in different phases of a data science life cycle like the business understanding, the data filtering and interpretation and the model evaluation and deployment [16]. The problems, predictions or questions that data science can answer are:

- Is it A or B? (Classification or categorization)
- Is this strange? (Detection of anomalies) [17]
- How much or how many? (Prediction of values)
- How is it organized? (Description of items)
- What should I do? (Prescription) [18]

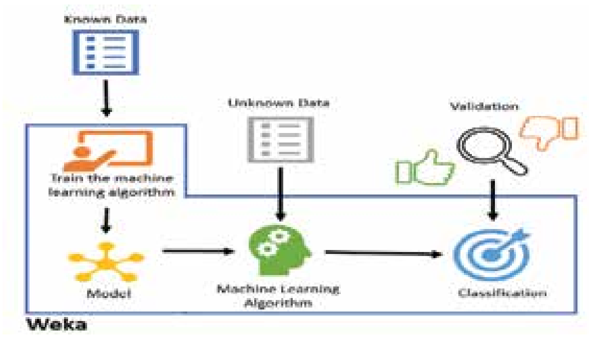
To develop data science projects, a set of available data regarding the problem to solve must be available. These data must fit at least the following criteria, which allow to validate the applicability of a data set to solve a problem:

- Volume: The available data must have a considerable size, due generally as much as data is included better result can be obtained.
- Relevant: Data should be relevant to the context of the question to be solved.
- Precise: The available data must be correct, so they can represent closely the reality of the problem.
- Not connected: Available data should also be not-linked but related, so the values of the features have not dependable relations that generate mistakes in the data science model [19].

Machine learning is the way knowledge is extracted, ordered and classified from apparently unconnected data, this knowledge is useful to do predictions and take decisions, with this definition and the definition of data science it is possible to say that machine learning is a discipline of data science [20].

The machine learning models can be generated from data using Weka [21], which contains a collection of machine learning algorithms for data mining. As shown in Fig. 8, a classificatory algorithm can be trained using an available data set that has been classified previously. Once a model is trained, unclassified data can be entered to the model, so it can make a classification for each one of them.

Fig. 8. DEVELOPMENT OF A CLASSIFICATORY MACHINE LEARNING MODEL


Known Data
Unknown Data
Validation
Train the machine learning algorithm
Model
Machine Learning Algorithm
Classification
Weka

Source: The authors

3. MACHINE LEARNING MODELS APPLIED TO MALWARE ANALYSIS

Even if sandboxing is a powerful technique to perform malware analysis, it requires that a malware analyst performs a rigorous analysis of the results to determine the nature of the sample: goodware or malware.

In a sandbox, we can observe the behavior of a malware that is trying to access, destroy, copy, or alter information with a motivation that can be economical or political. Those characteristics are obtained by analyzing PE or EFL for desktop applications o APKs for mobile applications. The results of the analyzes provided by a sandbox must be interpreted by an expert to determine if it is a malware or not. This analysis generally consumes time and the quality of the result depends on the expertise of the malware analyst. So, this paper proposes a mechanism that aims to resolve the data science question of classifying unknown samples between two possible options (Goodware or Malware) helping in this way to the malware analyst and reducing the time that is required for an analysis.

The mechanisms proposed in this paper are based on the development of machine learning models that can classify windows desktop and mobile android applications. An integration of Cuckoo with machine learning is proposed, so the results of Cuckoo are taken, and a machine learning model is trained with them, so it can be used to make classification of unknown samples. Fig. 9 shows the process of training of machine learning models using samples with that APK (Android Application Package) and EXE extensions (PE files). The process of training of machine learning

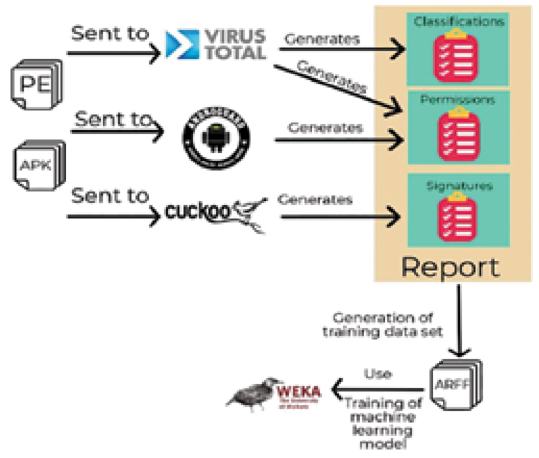
TABLE I
STRUCTURE OF THE TRAINING DATA SET FOR DESKTOP APPLICATIONS

	Signature 1	Signature 2	...	Signature 58	Class
Sample 1	1	0	...	1	GOO
Sample 2	0	0	...	1	MAL
...	...	...	...	...	...
Sample 108	1	1	...	0	MAL

Source: The authors.

models follow the steps: Collection of samples, classification of samples using VirusTotal, analysis of samples using Cuckoo sandbox or Androguard, extraction of sample features from Cuckoo or Androguard, generation of a training data set and training of a machine learning model. These steps will be applied in sections 3.1 and 3.2.

Fig. 9. TRAINING OF MACHINE LEARNING MODELS FROM SAMPLES



Source: The authors.

3.1 Analyzing PE files

The development of a machine learning model that allows classifying a sample between malware and goodware, required to download 108 samples from The Zoo[2] and from various official download pages, then such samples were classified using the API (Application Programming Interface) of Virus Total (as malware and goodware). In the next step, each of the files was uploaded to Cuckoo sandbox to perform a malware analysis. The results of these analyzes contain the signatures with which the sample matches. Cuckoo has a collection of signatures, which are updated or created daily by contributors around the world since Cuckoo sandbox is an open source project. The signatures allow us to identify patterns associated with previously known malware behaviors, e.g. a try of access to kernel commands or the holding of fake certificates. This analysis allow us to obtain a context of the sample and facilitates the interpretation of the results. Cuckoo sandbox has signatures that allow you to identify a family or categories of malware (spyware, worms, etc.) and identify modifications or installations that specific malware perform to gain control of a target. Currently, signatures are also used by some antivirus to detect malware [36].

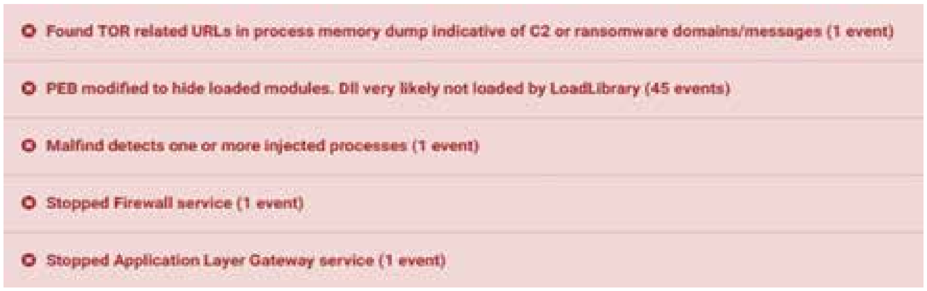
Each report generated by Cuckoo was exported and signatures were extracted from them, which became the basis for the construction of the machine learning model.

Fig. 10 shows some of the signatures found in a report indicating that the virtual machine firewall service was stopped, some URLS related to command and control servers were requested and processes were injected, amongst others suspicious findings that indicate that the sample presents a malicious behavior. To automatize the generation of reports from Cuckoo sandboxing, a python script (Parser.py) was created which extracts the Cuckoo signatures from reports and parse them to train the machine learning model.

A training data set was built from the obtained Cuckoo signatures of each sample for a total of 58 signatures or features. Each row in the training data set is a sample that has a set of signatures (marked as ones), that indicate that the sample matched that signature. On the other hand, the last column refers to the sample classification (Goodware or Malware) was obtained from the API of Virus Total. A representation of the training data set can be seen in table I. All the malware samples evaluated were taken from The Zoo.

[2] https://github.com/ytisf/theZoo.

Fig. 10. CUCKOO SIGNATURES



- Found TOR related URLs in process memory dump indicative of C2 or ransomware domains/messages (1 event)
- PEB modified to hide loaded modules. Dll very likely not loaded by LoadLibrary (45 events)
- Malfind detects one or more injected processes (1 event)
- Stopped Firewall service (1 event)
- Stopped Application Layer Gateway service (1 event)

Source: The authors.

The developed machine learning model has an accurateness of 88.8889% using the Random SubSpace method [22] using a 20-fold cross-validation. This model obtained the results shown in table II.

TABLE II
EVALUATION VALUES FOR RANDOM SUBSPACE MODEL

Kappa statistic	0.7295
Mean absolute error	0.2697
Root mean squared error	0.3338

Kappa statistic is a statistic measure that help to evaluate how much two classifiers agree in the classification of samples. When classifiers agree all the times, Kappa value is 1.

On the other side, when the classifiers only agree by chance, Kappa is 0 [23]. The Kappa statistic of the developed model is 0.7295, so it is not as good as expect, but it is high enough to be considered a substantial agreement without chance. The Mean absolute error (MAE) and the Root mean squared error are measures to describe the error in the average operation of the machine learning model [24].

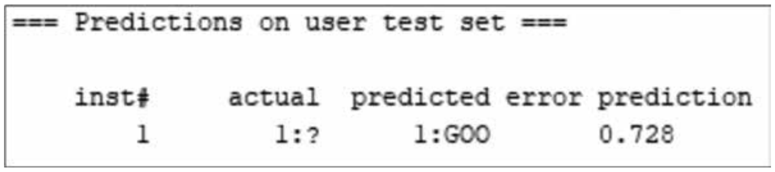
The MAE indicate the deviation between the classification got from the model versus the actual sample classification. In Fig. 11. an example of prediction using this machine learning model is shown, where a sample is classified as goodware with an error of 0.728.

3.2 Analyzing Android mobile applications

Despite all advantages and functionalities that Cuckoo provides in the analysis of malware for desktop operating systems, it has many restrictions and problems in the analysis of malware for mobile applications. Some of the problems are the reduced compatibility with Android SDK (Software Development Kit) and the restriction of just to allow one hypervisor running at the same time on the same machine, i.e. an android emulator could not run over a virtualbox instance.

On the other hand, Android emulation require high computational requirements, not to mention the incompatibility of processor architectures due to most of mobile devices are made with ARM architecture, while the sandbox and Cuckoo were implemented on Intel architecture. With these previously mentioned constraints, an analysis over an Android emulator could last up to three hours. To solve these problems, Androguard was used, which is a tool written in Python that have multiple functions to analyze Android files. Androguard client is simple to use and allows the automation of processes using classes included by default. Androguard allows to extract permissions requested by Android applications and use them as features to compose a training data set that can be used later in the machine learning generation process.

Fig. 11. PREDICTION FOR A DESKTOP MALWARE USING WEKA



```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1        1:?      1:GOO          0.728
```

Source: The authors.

A machine learning model was developed with 123 APK (Android Application Package) samples, which were classified as Goodware or Malware according to the results obtained from VirusTotal API. VirusTotal reports were used also to extract the Android permits of each APK sample. A training data set was built from the permits of each sample, having one feature for each permit, for a total of 256 signatures/features. Each sample is an element of the training set having a set of signatures/features that indicate if the sample matched with that permit.

Each permit has at least one application using it. The structure of the training data set for the model can be seen in table III. T generate the model, a python script was developed that takes VirusTotal reports, extract the identified permits and generates a training data set in a format (.arff) that can be accepted by Weka.

The trained machine learning model has an accurateness of 93.4959% using a neuronal network "Multilayer Perceptron" and a cross-validation of 10 blocks. This model obtained the results shown in table IV.

The Kappa coefficient of the model is 0.8105, it means that two classifiers are in a substantial agreement regarding if a sample is considered as goodware or malware. Following the analysis of these measures, the mean absolute error is quite low, and the mean square error is higher, meaning that the model has an acceptable accurateness.

Once the model has been trained, this can be used to make predictions about unknown Android samples. Testing of the model can be done using samples that are different from the ones used to compose the training data set. These samples can are analyzed by Androguard to obtain the permits and then parsed to compose a testing data set in a format that is accept by Weka. Once the testing set is parsed, it can be entered to the trained model in Weka and a prediction for each sample is generated which can support the analyst in the process of identifying unknown sample as malware. An example of prediction using Weka can be seen in Fig. 12, where an unknown sample is classified as malware with a 100% of probability.

3.3 Use case: IoT t3

This section presents a proposal of architecture for malware detection in IoT ecosystems using the machine learning model developed in 3.2. The model can be incorporated to a security component called sentinel which automate the classification of samples that are downloaded by IoT devices, so it can be possible to protect a network of IoT (Internet of Things) devices[37]. The sentinel architecture has several security rings that protect IoT devices with a defense in-depth strategy [25]. If one sample pass as goodware by the first ring, it will be evaluated by the second ring and then by a third ring. Sentinel architectu-

TABLE III
STRUCTURE OF THE TRAINING DATA SET FOR ANDROID APPLICATIONS

	Permit 1	Permit 2	...	Permit 256	Class
Sample 1	1	0	...	1	MAL
Sample 2	0	0	...	1	MAL
...	...	...	...	...	...
Sample n	1	1	...	0	GOO

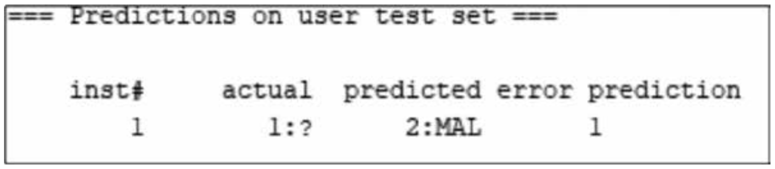
Source: The authors.

TABLE IV
EVALUATION VALUES FOR NEURAL NETWORK MODEL

Kappa statistic	0.8105
Mean absolute error	0.0679
Root mean squared error	0.2274

Source: The authors.

Fig. 12. PREDICTION FOR AN ANDROID MALWARE USING WEKA



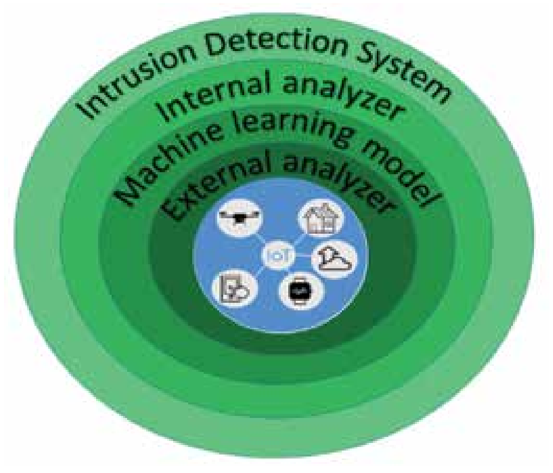
```
=== Predictions on user test set ===

    inst#     actual  predicted error prediction
        1        1:?      2:MAL       1
```

Source: The authors.

re can be seen in Fig. 13 with each ring having the following functions:

1. IDS (Intrusion Detection System) [26]: This ring detects intrusions in the network, whether by ethernet or wifi. It also analyzes the network searching for vulnerable IoT devices. All information collected by this ring is sent to an event correlation server.
2. Internal Analyzer Yara Rules [27]: These rules allow to define features to look in samples, so through static reversing engineering can be possible to detect malware samples.
3. Machine Learning model: In this ring, the sample is analyzed by the machine learning model developed in section 3.2. The machine learning model offered by this ring can be consulted through an API REST. Through this API the sample permits are gotten and passed to the trained model, so it can classify the unknown sample.
4. External analyzer: If none of the previous rings have detected the sample as malware, then it is forwarded to an external analyzer (e.g. VirusTotal) so it can be analyzed, and a classification report can be obtained.

Fig. 13. IOT SENTINEL ARCHITECTURE



Source: The authors.

4. USING SANDBOXING AND MACHINE LEARNING IN STATE SECURITY AGENCIES

The defense of malicious attacks is one of the biggest challenges for organizations and cybersecurity teams nowadays. In this regard, Symantec in its Internet Security Threat Report 2018 [28] mentions that threats to digital security can come from unexpected sources and that with each passing year will increase the volume and diversity of threats, with attackers working hard to discover new forms of attack and concealment. Likewise, this report highlights the following key points around existence of malware [29]:

- Coin mining was the largest area of growth in cybercrime in 2017, with antivirus detections of up to 8,500% compared to 2016.
- For 2017, ransomware infections increased 40% with respect to 2016, driven mainly by WannaCry (Ransom.Wannacry).
- The number of ransomware variants for 2017 increased 46% with respect to 2016, even though fewer new families emerge, indicating an activity intensified by established groups, i.e. existing families increments its number of ransomware variants.
- Emotet (Trojan.Emotet) reappeared at the end of 2017 as a major threat for banking sector and Emotet detections increased by 2,000% in the last quarter.
- E-mail filtering, intrusion prevention system (IPS) and machine learning have supported the early detection in the ransomware chain of infection. In this regard, 2017 presented an increase of 92% compared to 2016, in blocking of scripts and macro downloaders, which are one of the largest sources of ransomware and banking threats.
- The general variants of malware have increased by 88%; mainly composed by a single type

of threat, the Trojan.Kotver! gm[2], that represents 95% of the total data.

As mentioned by Yokoyama A. et al [30], these growing trends in the amount and variety of malware are consequence of polymorphism, i.e. malware that uses a polymorphic engine to mutate itself while keeping its original algorithm intact [31]. This technique is commonly used by computer viruses and worms to hide its presence. Malware growing trends are also impacted by new threats discovered almost daily. In this sense, cybersecurity teams use technologies and techniques that facilitate malware analysis and help to prevent attacks on critical assets, such as sandboxing [30].

Sandboxing is a technique that allows to enforce security policies for untrustworthy applications in a secure environment to reduce risks for the host system. It is appropriate to mitigate threats that traditional security systems are unable to prevent [32]. Due to new threats imply a more detailed and holistic analysis of its multiple features, sandboxing can be enhanced by other techniques such as machine learning, which seeks [33] to resolve problems in an automated way using experience and empirical data. Two sandbox approaches are traditionally managed: rule and isolation based, each one has advantages and limitations. In an isolation based approach an application A executes an application B in a sandboxed environment, usually virtualized, where application B has access to some specific and controlled sandbox resources [34]with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to protect against contemporary threats, such as previously unknown ('zero-day'. On the other hand, in a ruled based approach each sandbox enforces a specific policy regarding the resources that applications within the sandbox can access. Application A can launch application B implicitly in sandbox B which enforce a policy for B [34]with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to protect against contemporary threats, such as previously unknown ('zero-day'.

Among the most prominent variations of sandbox approaches is the one presented by Potter and Nieh, mentioned by Schreuders ZC et al. [34] with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to protect against contemporary threats, such as previously unknown ('zero-day', who propose a container based sandbox for application-oriented access control. This proposal is known as Apiary and works in a similar way to the security operating system Qubes, which was developed by Rutkowska and Wojtczuk, and aims to provides isolation through a virtualization interface for security purposes [34] with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to protect against contemporary threats, such as previously unknown ('zero-day'. Another solution that applies sandboxing principles is Cuckoo SandBox, which is an open source software for automated analysis of suspicious files that monitor the behavior of malicious processes under an isolated environment.

Similarly, Schreuders ZC et al. [34]with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to protect against contemporary threats, such as previously unknown ('zero-day' mention that some sandbox schemes allow applications to read data from the host computer with some restrictions like copy-on-write functions, which are resource management techniques used in programming to implement efficiently a "duplicate" or "copy" operation [35]. Copy-on-write functions allow to write any modification made by the application in a virtual disk instead of the real hard disk. Solutions like Sandboxie[3], and Shade Sandbox[4] follow the approach mentioned previously. Other sandbox solutions run stand-alone applications through a Virtual Machine Manager (VMM) or an interpreter, thus, avoiding applications of making changes to the host computer without user intervention. Java, Silverlight and Flash Applets use the approach here mentioned to insert mobile code content into websites with a very limited access to storage and additional accesses granted through user interaction [34]with each process typically capable of utilising all of the user's privileges. Consequently such security mechanisms often fail to

3 https://www.sandboxie.com/
4 https://www.shadesandbox.com/

protect against contemporary threats, such as previously unknown (‘zero-day’.

Sandboxing integrated with machine learning constitute a defense that fits with more of these axioms because it is designed thinking in the attacker capabilities to build sneaky software that contains malicious functions. Additionally, it allows to examine malicious malware before it reaches critical assets, offering a detection and delay of the attack. It also constitutes an additional security layer that allows to avoid that an attack succeeds. This is due to sandboxing is designed with the purpose that a malware does not affect the area outside an isolated environment, even if the malware defeats all the defenses arranged in the sandbox. Through sandboxing it can be possible to detect and delay attacks by running suspicious samples in a test environment which can be empowered with machine learning to offer an additional layer of defense against unknown threats or mutations of some already extended malware. This would allow security teams to have time to respond to attacks, supplying the shortcomings of other security layers like antivirus. Now, in relation to national initiatives related to the use of sandboxing and machine learning by cybersecurity teams and state security agencies, the Government of Colombia has been developing some projects applied to its cybernetic defense processes, but these are isolated and used alternately and not combined. That is why, the developments presented in this paper are a novel proposal in this regard which integrates sandboxing and machine learning, supporting in the way cybersecurity teams such as the Cybernetic Joint Command (CCOC). Specifically, machine learning models proposed in Section 3.1 and 3.2, and architecture described in Section 3.3, are evidence of how machine learning and sandboxing can work together to automates the classification process of unknown samples.

Regarding the CCOC [36], who is responsible for the nation cyberdefense, some research and development (R&D) initiatives have been implemented, with support of the Colombian School of Engineering Julio Garavito, to include machine learning techniques in cybersecurity process. One example is the project “Implementation of a Security Information and Event Management System for information asset protection” which integrates machine learning models in event correlation engines, so it can be possible to predict cybersecurity incidents related with critical cybernetic infrastructure. Another example is the project “Open Source Intelligence applied to the Colombian context” which develops some machine learning models to make sentiment analysis of information collected from social networks. These two initiatives optimize labors of CCOC agents working on SOC (Security Operation Center) and intelligence groups. A governmental initiative related with sandboxing is the “Malware Analysis Service” offered by the Colombian National Police [37] through the CSIRT division. This service allows to perform file or URL analysis, identifying suspicious features that are common in malware. It is implemented over Cuckoo Sandbox and outputs a summarized malware analysis that help Colombian citizens to determine if a file or URL is malicious.

As described previously, cybersecurity teams and governments must face the challenges of new threats through techniques that allow the software to be examined holistically and provide an advanced level of analysis, beyond the techniques that traditionally have led the battle against malware. That is why initiatives such as those presented in this paper, which combine malware analysis techniques such as sandbox with advanced analysis techniques such as machine learning, are highly relevant and allow a proactive reaction to cyberattacks. At last, it must be remembered that sandbox solutions are susceptible to anti sandbox techniques, e.g. use of virtualization detection techniques, detection of presence of real users (Turing test), detection of hooking, exploitation of sandbox limitations or use of advanced Anti-VM and Anti-Sandbox tools. This poses a new challenge related to how protection against sandbox evasion techniques must be conducted, and in the same way it is a warning for sandbox operators regarding threats that in fact may already be executed silently with such techniques [30]. Finally, there are many advantages of using sandbox and machine learning, even if it also presents some challenges in for its application by cybersecurity teams of state security agencies. Specifically, for the Colombian State, the identified challenges are:

- Update of machine learning models to be re-trained with recent malware samples.
- Counter techniques and tools for Anti-VM and Anti-Sandbox.
- Implement the sharing of malware information with national and international security agencies that allow to strength the cyber-defense processes.
- Deal with the existence of malware samples that are found only in Colombia.
- Build more complete machine learning models that include more malware features, in addition to signatures and permits.

5. CONCLUSIONS

Sandbox techniques are useful to make malware analysis however they can be automated and supported by machine learning models, so a malware analyst can perform malware identification tasks with more precision and in less time. Additionally, regarding analysis of mobile malware threats with sandbox, it is needed that the sandbox solution has integration with mobile devices operative systems. To develop machine learning models, it is mandatory to validate the data quality in terms of the criterions mentioned in section 2.2 (Volume, Relevant, Precise, Not connected). The two models developed in this paper use signatures and permits as features of samples which were considered as relevant in a malware detection problem. The precision of the data depends on the ability to extract these features from the samples using Cuckoo, Androguard or Virus Total. As future work we will increase the number of samples to increase the volume of data.

The machine learning models developed in this paper can classify a sample as Goodware and Malware, however with a bigger dataset it could be possible to classify samples in more than two classes. A more fine-grained classification of samples could allow to identify different malware categories and would be useful to determine specific countermeasures for each class. Finally, the approaches and perspectives presented in section 3 regarding the use of machine learning models can support the development of new IoT security devices that offer a strong security schema against new malware that has not been previously classified by regular security mechanism.

ACKNOWLEDGMENT

This work has been supported partially by the Colombian School of Engineering Julio Garavito (Colombia) through the project “Cyber Security Architecture for Incident Management”, funded by the Internal Research Opening 2017.

REFERENCES

[1] Kaspersky, “Kaspersky Lab detects 360,000 new malicious files daily – up 11.5% from 2016,” 2014. [Online]. Available: https://kaspersky.com/about/press-releases/2017_kaspersky-lab-detects-360000-new-malicious-files-daily. [Accessed: 13-Aug-2018].

[2] M. Sikorski and A. Honig, *Practical Malware Analysis : a Hands-On Guide to Dissecting Malicious Software.* No Starch Press, 2012.

[3] J. M. Ehrenfeld, “WannaCry, Cybersecurity and Health Information Technology: A Time to Act,” *J. Med. Syst.*, vol. 41, no. 7, p. 104, Jul. 2017.

[4] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, “IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2177-2184.

[5] C. Wang, J. Ding, T. Guo, and B. Cui, “A Malware Detection Method Based on Sandbox, Binary Instrumentation and Multidimensional Feature Extraction,” in *Advances on Broad-Band Wireless Computing, Communication and Applications*, 2018, pp. 427-438.

[6] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, “OPEM: A static-dynamic approach for machine-learning-based malware detection,” in *Advances in Intelligent Systems and Computing*, 2013, vol. 189 AISC, pp. 271-280.

[7] P. Burnap, R. French, F. Turner, and K. Jones, “Malware classification using self organising feature maps and machine activity data,” *Comput. Secur.*, vol. 73, pp. 399-410, Mar. 2018.

[8] S. E. Donaldson, S. G. Siegel, C. K. Williams, and A. Aslam, “Defining the Cybersecurity Challenge,” in *Enterprise Cybersecurity Study Guide: How to Build a Successful Cyberdefense Program Against Advanced Threats*, Berkeley, CA: Apress, 2018, pp. 3-51.

[9] O. Ferrand, “How to detect the Cuckoo Sandbox and hardening it ? Keywords.”

[10] T. Teller and A. Hayon, "Enhancing Automated Malware Analysis Machines with Memory Analysis."

[11] R. Messier, *Network Forensics*. Wiley, 2017.

[12] D. Oktavianto and I. Muhardianto, *Cuckoo malware analysis: analyze malware using Cuckoo Sandbox*.

[13] M. A. Waller and S. E. Fawcett, "Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management."

[14] F. Provost and T. Fawcett, *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O'Reilly Media, 2013.

[15] G. S. Nelson, *The analytics lifecycle toolkit: a practical guide for an effective analytics capability*.

[16] D. (Computer scientist) Dietrich, R. Heller, B. Yang, and EMC Education Services, *Data science and big data analytics: discovering, analyzing, visualizing and presenting data.*

[17] T. Dunning and B. E. Friedman, *Practical machine learning: a new look at anomaly detection*. O'Reilly Media, 2014.

[18] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business Intelligence and Analytics: From Big Data to Big Impact," *MIS Quarterly,* vol. 36. Management Information Systems Research Center, University of Minnesota, pp. 1165-1188, 2012.

[19] L. Sebastian-Coleman, *Navigating the Labyrinth: An Executive Guide to Data Management.* Technics Publications, 2018.

[20] A. L'heureux, K. Grolinger, H. F. El Yamany, M. A. M. Capretz, A. L'heureux, and K. Grolinger, "Machine Learning with Big Data: Challenges and Approaches 4 PUBLICATIONS 100 CITATIONS SEE PROFILE," 2017.

[21] B. Kaluža, *Instant Weka how-to: implement cutting-edge data mining aspects in Weka to your applications*. Packt Pub, 2013.

[22] D. Tao, S. Member, X. Tang, S. Member, X. Li, and X. Wu, "Asymmetric Bagging and Random Subspace for Support Vector Machines-Based Relevance Feedback in Image Retrieval."

[23] J. M. G. Anthony J. Viera, "Understanding interobserver agreement: the kappa statistic," 2005.

[24] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Clim. Res*., vol. 30, no. 1, pp. 79-82, Dec. 2005.

[25] R. Lippmann *et al*., "Validating and Restoring Defense in Depth Using Attack Graphs," in *MILCOM 2006*, 2006, pp. 1-10.

[26] S. Snapp *et al*., "DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype," *http://www.academia.edu/download/4378230/10.1.1.46.4991.pdf*, 2017.

[27] M. Mansoori, I. Welch, and Q. Fu, "YALIH, yet another low interaction honeyclient," *Proc. Twelfth Australas. Inf. Secur. Conf. - Vol. 149*, pp. 7-15, 2014.

[28] Symantec Corporation, "ISTR Internet Security Threat Report.," Mountain View, CA 94043, 2018.

[29] S. Corporation, "ISTR Internet Security Threat Report Volume 23," Mountain View, CA 94043, 2018.

[30] A. Yokoyama *et al*., "Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9854 LNCS, pp. 165-187.

[31] D. Harley, R. Slade, and U. E. Gattiker, "Polymorphism," in *Viruses Revealed: Understand and counter maliciosus software*, United States: McGraw-Hill/Osborne, 2001, p. 10.

[32] M. Stephens, "Sandbox," in *Encyclopedia of Cryptography and Security,* H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA: Springer US, 2011, pp. 1075-1078.

[33] Gass S.I., Ed., "Machine Learning," in *Encyclopedia of Operations Research and Management Science*, Boston, MA: Springer US, 2013, pp. 909-909.

[34] Z. C. Schreuders, T. McGill, and C. Payne, "The state of the art of application restrictions and sandboxes: A survey of application-oriented access controls and their shortfalls," *Comput. Secur*, vol. 32, pp. 219-241, Feb. 2013.

[35] D. P. (Daniel P. Bovet and M. Cesati, *Understanding the Linux kernel.* United States of America: O'Reilly, 2002.

[36] CGFM, "Comando Conjunto Cibernético," 2018. [Online]. Available: http://www.ccoc.mil.co/.[Accessed: 13-Aug-2018].

[37] PONAL, "CSIRT - Equipo de Respuesta a Incidentes Informáticos." [Online]. Available: https://cc-csirt.policia.gov.co/Sandbox. [Accessed: 13-Aug-2018].