

A hybrid machine learning approach for analysis of stegomalware

Prudence Kadebu

*Department of Information Systems, Women's University in Africa,
Harare, Zimbabwe*

Robert T.R. Shoniwa

Zimbabwe Information and Communication Technology, Harare, Zimbabwe

Kudakwashe Zvarevashe

Victoria Institute of Technology, Adelaide, Australia

Addlight Mukwazvure

Department of Computer Engineering, University of Zimbabwe, Harare, Zimbabwe

Innocent Mapanga

*A Division of Zimbabwe Institution of Engineers,
Zimbabwe Information and Communication Technology, Harare, Zimbabwe*

Nyasha Fadzai Thusabantu

*A Division of Zimbabwe Institution of Engineers,
Zimbabwe Information and Communication Technology, Harare, Zimbabwe and
Department of Academics, Harare Institute of Technology, Belvedere, Zimbabwe, and*

Tatenda Trust Gotora

*A Division of Zimbabwe Institution of Engineers,
Zimbabwe Information and Communication Technology, Harare, Zimbabwe and
Department of Computer Science, Midlands State University, Gweru, Zimbabwe*

Abstract

Purpose – Given how smart today's malware authors have become through employing highly sophisticated techniques, it is only logical that methods be developed to combat the most potent threats, particularly where the malware is stealthy and makes indicators of compromise (IOC) difficult to detect. After the analysis is completed, the output can be employed to detect and then counteract the attack. The goal of this work is to propose a machine learning approach to improve malware detection by combining the strengths of both supervised and unsupervised machine learning techniques. This study is essential as malware has certainly become ubiquitous as cyber-criminals use it to attack systems in cyberspace. Malware analysis is required to reveal hidden IOC, to comprehend the attacker's goal and the severity of the damage and to find vulnerabilities within the system.

Design/methodology/approach – This research proposes a hybrid approach for dynamic and static malware analysis that combines unsupervised and supervised machine learning algorithms and goes on to show how Malware exploiting steganography can be exposed.



Findings – The tactics used by malware developers to circumvent detection are becoming more advanced with steganography becoming a popular technique applied in obfuscation to evade mechanisms for detection. Malware analysis continues to call for continuous improvement of existing techniques. State-of-the-art approaches applying machine learning have become increasingly popular with highly promising results.

Originality/value – Cyber security researchers globally are grappling with devising innovative strategies to identify and defend against the threat of extremely sophisticated malware attacks on key infrastructure containing sensitive data. The process of detecting the presence of malware requires expertise in malware analysis. Applying intelligent methods to this process can aid practitioners in identifying malware's behaviour and features. This is especially expedient where the malware is stealthy, hiding IOC.

Keywords Malware analysis, Steganography, Stegomalware, Malware detection, Machine learning

Paper type Research paper

1. Introduction

Cyber attackers are constantly conceiving new sophisticated ways to perpetrate breaches on systems while making their exploits difficult to detect. Injection of malicious software is one typical method commonly used in these attacks. Malicious software, also known as malware, refers to any type of software such as viruses, trojans, rootkits, trapdoors, adware and ransomware (Distler, 2007). It has become one of the many threats wreaking havoc across the Internet, threatening individuals and businesses around the world. These have a negative impact both on business operations and service delivery as well as the privacy of personal information of individuals. Moreover, significant financial losses as a result of malware are constantly reported. Those who use the Microsoft Windows operating system have been particularly vulnerable. In the past, ransomware cyberattacks were commonly observed in major corporations, but nowadays, even individuals have suffered security breaches. Various kinds of point-of-sale (POS) malware have infected merchants' POS systems, with the goal of harvesting the credit card details of customers utilizing remote access tools and credential dumpers to gain access to the system and inject malware (Rautmare, 2020). Malware can be introduced into the system via phishing attempts, in which an insider opens a link sent to their email that contains malware. This is a typical method for threat actors to obtain remote access to systems.

Of late, a type of malware has emerged applying methods for steganography, which is a technique for covering data so that it cannot be easily identifiable. This has become popular as malware is embedded into images through subtle alterations of pixels, especially in colour images with many thousands of pixels. This is also possible in smartphone App assets like databases, documents and multimedia content (Suarez-Tangil *et al.*, 2014). This type of malware using steganography is very stealthy and difficult to detect by conventional methods. Ucci *et al.* (2019) states that as security technology evolves, evasion occurs very instantly, demonstrating that attackers are constantly devising ways to circumvent security mechanisms designed to safeguard systems against breaches. Malware analysis involving the detection, analysis and investigation of malware in a system is the way cyber security practitioners gain traction ahead of threat actors while taking into cognisance the fact that these criminals are also doing the same, hence the need for continuous improvement of methods for dealing with malware. There are two main types of malware analysis: dynamic analysis and static analysis (Oktavianto and Muhandianto, 2013). The former refers to methods aimed at profiling actions of the malware binary at runtime, and the latter is aimed at decompiling and analysing the internal structure of the binary itself (Sharif *et al.*, 2008). In static analysis, the malware programme code is inspected to identify hidden patterns and all behavioural scenarios of the malware there without execution of the programme code (Chumachenko, 2017). Techniques such as file fingerprinting are used with static analysis to ensure that the file does not change during the analysis. MD5sums and MD5deep are some examples used for this purpose (Kendall and Mcmillan, 2007). The source code may not be available for static analysis, and thus the analysis has to start with some reverse engineering tasks of disassembling and decompiling for static analysis

to be conducted (Oktavianto and Muhardianto, 2013). Dynamic analysis entails that the behaviour of the malware is monitored during execution of the file. Malware authors make use of techniques such as obfuscation (Bhojani, 2014), viral polymorphism (Bashari Rad *et al.*, 2012) packers, anti-debugging, anti-disassembling, sandbox evasion and antivirus evasion. These mask the traces of their presence and behaviour patterns, allowing them to dodge detection and impede mechanisms put in place. Static analysis, which would ordinarily aid in the reverse engineering of malware code, is hampered by them. As a result, dynamic analysis outclasses static analysis. Sandboxing is a technique for dynamic analysis that involves running files in a virtual environment for a more detailed analysis (Sethi *et al.*, 2017). Isolating suspicious files ensures analysis is done without the risk of the malware affecting the system during execution. Cuckoo sandbox is one such type (Oktavianto and Muhardianto, 2013). If a system is compromised, the first action in incident response is to isolate the affected system. If it is a server, it should be disconnected from the network to prevent the damage from spreading to other areas of the system and subsequently followed by either static or dynamic analysis.

Because most tools and approaches, such as antivirus software, rely on malware signatures, any techniques that cannot detect stealth attacks are useless in mitigating them. Thanks to advances in the field of machine learning, there has been an uptick in strategies that are more effective in dealing with malware and can defend against the threats posed by malware exploits. To that end, the goal of this work is to propose a machine learning approach to improve malware detection by combining the strengths of both supervised and unsupervised machine learning techniques. This study is crucial as malware has certainly become ubiquitous in its use by criminals to attack systems on the cyberspace (Lengyel *et al.*, 2014). Malware analysis is required to reveal hidden indicators of compromise (IOC), as well as to comprehend the attacker's goal, the severity of the damage and to find vulnerabilities within the system (Kendall and Mcmillan, 2007).

Given how smart today's malware authors have become, employing highly sophisticated techniques, it is only logical that methods be developed to combat the most potent threats, particularly where the malware is stealth, making IOC difficult to discover. After the analysis is completed, the output can be employed to detect and then neutralize the attack (Baker, 2020). The research paper is organized as follows: Section 2 focuses on the review of literature on malware analysis with more focus on malware detection. The methodology will come in Section 3 to describe the proposed work. Section 4 looks at the methods for malware analysis, followed in Section 4 by a proposed framework for hybrid malware analysis using supervised and unsupervised machine learning, then in Section 5 by results and discussion. Finally, in Section 6, we conclude the work.

1.1 Objectives

The objective of the research is to uncover the methods used in malware analysis by surveying related literature. The research also aims to expose the various means by which the malware authors try to ensure the malware remains undetected. The authors also aim to assess the extent to which machine learning techniques are being applied in malware analysis and propose a hybrid framework for unsupervised and supervised machine learning for malware analysis.

2. Literature review

The increase in connected devices and access to resources deployed online has led to the widening of the attack surface of most systems. The Internet is a jungle awash with malware of all types deployed in diverse ways. However, the task of detecting and identifying the

malware is a daunting one due to obfuscation which is applied by cyber attackers to mask its presence (Komatwar and Kokare, 2021; Okane *et al.*, 2011). Steganography and obfuscation is one way malware gets deployed onto a victim's system through embedding it within an image to produce a type of malware known as stegomalware. This leverages the prevalence of free image hosting services on the Internet and the absence of techniques that can efficiently detect malware within images. Scanning every image for malware also comes with high overheads which may slow down the system. Suarez-Tangil *et al.* (2014). present a method supported by a tool for identifying stegomalware in smartphones. Their method focuses on identifying the location and components to be extracted in malware detection. They used the tool to analyse 55,000 apps, and their findings show the presence of stegomalware in many apps.

Early discovery of stegomalware includes ZeusVM banking Trojan which was discovered at Trend Micro embedded in JPEG images meant to launch a man-in-the-browser (MITB) attack in 2014. ZeusVM, using steganography from seemingly harmless images, retrieved configuration information (O'Meara *et al.*, 2016). This was used to intercept banking transactions, allowing the attackers to transfer money from the victim's bank account. Steganography permits the malware to evade intrusion detection systems and antivirus software.

Sharif *et al.* (2008) propose Eureka, a framework for analyzing static Internet malware binaries using a de-obfuscation technique. In a Eureka-managed VM environment, Eureka examines the subject-packed malware binary for evidence of tracing or debugging. After that, the malware is unpacked. Eureka uses a coarse-grained execution tracker that employs a heuristic-based as well as a binary n-gram statistical trigger to determine whether the unpacked process picture has reached a stable state. The IDA-Pro disassembler disassembles the reconstructed process image and performs API resolution, which is a procedure that uses automated de-obfuscation to recover concealed API invocations enabling static analysis to proceed. The image of the code is then sent to the analysability metrics module, which evaluates the static analysis results. Finally, Eureka uses the code graph and ontology creation module to generate the call graph and ontology labels, as well as extracting, annotating and simplifying the structure. This effort is critical in addressing the issue of obfuscation, which would ordinarily obstruct static analysis. It will not, however, be able to assist in the classification of the malware that has been found.

A dynamic malware analysis system they termed DRAKVUF is designed to improve stealth by executing malware samples while leaving no trace in the analysis VMs is presented in (Lengyel *et al.*, 2014). We also present novel techniques to eliminate blind-spots created by kernel-mode rootkits by extending the scope of monitoring to include kernel internal functions, and to monitor file-system accesses through the kernel's heap allocations. With extensive tests performed on recent malware samples, we show that DRAKVUF achieves significant improvements in conserving hardware resources while providing a stealthy, in-depth view into the behaviour of modern malware. DRAKVUF is implemented based upon four requirements of scalability, which allows for reduction in performance overhead in sample analysis while improving on the capacity to analyse large samples concurrently. The second requirement, fidelity, ensures that enough data are collected at runtime while also allowing for resistance to against tampering for accurately analysis. The third requirement, stealth, ensures that DRAKVUF remains undetectable even within a monitored environment. Lastly, isolation means DRAKVUF should be protected against tampering by isolating it from the analysis VMs. The advantage of this malware analysis system is its ability to execute stealthily, thus making it easy to detect malware without the intruder being aware. However, like in the study by Lengyel *et al.* (2014), it cannot assist in classification of malware.

Min and Varadharajan (2014) proposed a technique of "feature-distributed malware" which applies a method for bypassing security defence mechanisms such as application

whitelisting, and also evades behavioural detection by antivirus' software. The malware is sophisticated in the way it distributes its features to several software components dynamically, thus making its detection very difficult. The paper exposes the risks associated with such advanced malware and also suggests ways to circumvent them. The defence utilizes digital certificates of software components to thwart any attempts to load malicious components.

A survey of malware analysis using machine learning is presented in the study by [Ucci et al. \(2019\)](#) where they expose the need to make trade-offs between maintaining high accuracy and performance of malware analysis and supplying the required equipment, a study termed malware analysis economics. They also make a connection between feature extraction and execution time, and the work goes on to assess whether desired features come from static or dynamic analysis, which has a bearing on execution time since, unlike dynamic analysis, static analysis does not require running the samples. Although the research gives a survey of techniques, it also provides a qualitative and simplified illustration of analysis leveraging on the introduced trade-off.

A two-level framework for detecting (macro) and classifying (micro) malware is proposed in ([Sethi et al., 2017](#)). The Cuckoo sandbox is used in the work to generate reports for static and dynamic analysis from executing sample files in a virtual environment. Their framework includes a novel feature extraction module. Weka framework is used to build and test different machine learning models on a data set of 220 samples of both malicious and benign files. J48, SMO and random forest yielded 100%, 99% and 97% detection rate, respectively, and 100%, 91% and 66.67%, respectively, for classification. The data set used in the experiments is too sparse which may create bias.

A data mining framework that uses automatically discovered patterns from data to detect new malicious binaries is proposed in ([Schultz et al., 2001](#)). They try to solve the problem of hand-generated heuristics used by antivirus software which are usually costly and ineffective as virus signatures evolve. In contrast to the normal virus scanner technologies that have a virus signature detector and a classifier to detect new viruses based upon the detected signatures, the proposed framework trained data mining algorithms which improve on the reduction in false positive rates. Most of the literature reviewed focused on dynamic malware analysis which involves execution of the malicious code to understand the malware. [Tables 1 and 2](#) give an overview of malware analysis and a brief summary of some of the researches, respectively. The success of malware analysis techniques depends upon how fast the malware can be detected. Machine learning techniques have been popularly applied to detect malware with very high performance.

In contrast to the normal virus scanner technologies that have a virus signature detector and a classifier to detect new viruses based upon the detected signatures, the proposed framework trained data mining algorithms which improve on the reduction in false positive rates. Most of the literature reviewed focused on dynamic malware analysis which involves execution of the malicious code to understand the malware. [Tables 1 and 2](#) give an overview of malware analysis and a brief summary of some of the researches, respectively. The success of malware analysis techniques depends upon how fast the malware can be detected. Machine learning techniques have been popularly applied to detect malware with very high performance.

3. Methods

This section discusses the most common techniques for malware analysis that is sandboxing and also machine learning. The Cuckoo sandbox as a case study will be explored to understand what information can be realized regarding the malware behaviour and attributes.







Types of Malware Virus Worm Trojan Horse Backdoor Spyware Adware Rootkits Stegomalware		Types of Malware Analysis Static Analysis Dynamic Analysis Hybrid	
	Tools for Analysis Sandboxes Sniffers Reverse code Engineers Disassemblers Debuggers Decompilers Network analyzers		Malware Defense Firewall Antivirus Software Secure Web Gateways Sandboxing
Malware Attacks/Acquisition Phishing emails Social Engineering Infected attachments Infected websites malicious browser plugin USB infection		Incident Response Isolation of compromised unit Preparation, Identification, Containment, Eradication Recovery, Lessons learned.	

Table 1.
Malware analysis
in brief

3.1 Cuckoo sandbox

The Cuckoo sandbox, developed by Claudio “nex” Guarnieri, is an open source sandboxing tool that is used for dynamic malware analysis (Oktavianto and Muhardianto, 2013). It implements a virtual malware analysis lab which offers a platform to isolate and analyse suspicious files without risking having malware replicating and affecting other parts of the system. Cuckoo automatically executes files, analyses them and collects comprehensive results about the behaviour of the malware as it runs in an isolated virtual environment. A file with malware can be submitted into the Cuckoo virtual environment for analysis. After submission, Cuckoo creates several subfolders for the analysis including *memory.dmp*, file containing the full memory of the analysis machine, and is useful in memory forensic analysis. In the subfolder, there is also the *dump.pcap*, which is the network dump file that can be further analysed with a network packet tracer like Wireshark in the case of Cuckoo. Additional tools come in handy for various processes, and these may also be combined. For instance, an advanced persistent attack (APT) is analysed using Cuckoo sandbox, volatility and another tool YARA for a more enhanced analysis.

Cuckoo sandbox was used to analyse an infected executable file. In Table 3, the executable file was identified to have a virus known as Sality. Static analysis performed using Cuckoo shows that some libraries are imported from KERNEL32.DLL, which can then add an entry to the registry for instance:

```
HKEY_LOCAL_MACHINE\Software\ Microsoft\Windows\CurrentVersion\Run
```

This defines programmes that can run at start-up; thus, the virus will use this to try and maintain access to the victim’s machine this way. The malware can imitate legitimate software. Its activities are shown in the processes section. Malware executable files can be viewed by means of a disassembler application such as Radare, a reverse engineering tool.

Table 2.
Summary from
literature review

Ref	Main idea	Contribution	Techniques	Data	Performance	Remarks
Sharif <i>et al.</i> (2008)	Eureka framework for enabling static Internet malware binaries analysis	Course-grained execution tracker applying a heuristic based and a binary n-gram statistical trigger to estimate when to stop the malicious process image	API resolution techniques IDA-Pro Disassembler De-obfuscation	Corpus of 1,291 malware instances, 479 malicious executables from spam traps, 435 malicious executables - HoneyNet	97.7% Spam malware corpus unpacking 93.3% honey net malware corpus unpacking Unpacking of 90 binaries/hr	Automated classification of malware
Lengyel <i>et al.</i> (2014)	DRAKVUF- dynamic MA system	Improves stealth by enforcing scalability, fidelity, stealth and isolation conserving resources	Hardware virtualization extensions and the Xen hypervisor	1,000 samples from shadow server	Memory saving of 62.4%	Automated classification of malware
Ucci <i>et al.</i> (2019)	A survey on MA through machine learning techniques	Novel concept of MA economics, malware anti-analysis techniques, etc.	A qualitative analysis	Processing one million malware per day	86% accuracy using 3 as minimum n-grams size	Tuning strategies to balance metrics such as accuracy and cost in designing MA environment
Schultz <i>et al.</i> (2001)	A data mining framework for automatically detecting new malicious binaries	Method for detecting previously undetectable malicious executables	Naive Bayes, multimodal-naive Bayes, RIPPER standard statistical cross-validation	Data set of 4,266 programmes 3,265 malicious binaries and 1,001 clean programmes	Multi-naive Bayes yielded highest detection rate 97.76%	Extension of learning algorithms to make use of byte-sequences
Sethi <i>et al.</i> (2017)	A framework for detecting and classifying malware	Intelligent MA framework for dynamic and static analysis of malware samples based on similarity	J48, SMO and random forest Cuckoo sandbox for malware analysis	220 Samples of malicious and benign files	100%, 99% and 97% detection rate, and 100%, 91% and 66.67% classification, respectively	Data set of 220 samples needs to be expanded

VirusTotal	38/42 (collapse)	
	Antivirus	Result
	nProtect	Win32.Sality.E
	CAT-QuickHeal	None
	K7AntiVirus	Virus
	TheHacker	W32/Sality(rp).I
	VirusBuster	Win32.Sality.L
	NOD32	Win32/Sality.NAE
	F-Prot	W32/Sality.K
	Symantec	W32.HLLP.Sality.O
	Norman	W32/Sality.N
	ByteHero	None
	TrendMicro-HouseCall	PE_SALITY.AE
	Avast	Win32:Sality-U
	eSafe	Win32.Sality.gen
	ClamAV	W32.Sality.N
	Kaspersky	Virus.Win32.Sality.I

Table 3.
Sample results of
analysis of
executable file

Besides disassembling, it is also widely used in debugging, analysing and manipulating binary files. Bokken, a front-end application, can also be used to support the process, and it shows the flowgraph of the binaries after disassembling. Hexdump tab and file info tab can be viewed in Bokken. At this point, dynamic analysis can be done with Cuckoo sandbox to give details of the behaviour of the malware in terms of what it does in the system and what changes it makes. Cuckoo uses various reporting formats, such as human-readable format, MAEC (malware attribute enumeration and characterization) format and can also export data reports to other formats. Once reports have been produced from a sandboxing tool such as Cuckoo in a human readable format in natural language, it becomes easier to augment the results by applying machine learning techniques to the process to improve on the quality of malware analysis for further instances.

3.2 Steganalysis

A scanning method is proposed that can be used to detect the presence of stegomalware embedded in JPEG images, the analysis and reporting of the findings from the data as shown in [Figures 1 and 2](#). The method is mainly focused on steganalysis of images with the intent of exposing the presence of hidden malware in an image, extraction thereof and analysis of the data if malware is indeed present. This is proposed to aid in the de-obfuscation process focusing on stegomalware. The process involves loading of the image to be scanned into memory, followed by assignment of a unique identifier so that if the same or a similar image with a different file name is also scanned during the same session, there will be no redundancy. The whole procedure can be applied in the same manner for all images with the same identifier.

This improves the efficiency of the whole procedure. The steganalysis algorithms (chi-square attack, visual detection, histogram analysis) are then used to check if the image has any steganography artefacts inherent. These artefacts are any identifiable signs or properties of the image that could be signs of the presence of steganography implemented on it. If this is not so, an image threat Level is then assigned and set to 0 if no steganography is identified. However, in the case of suspicious files, which could either be executable or non-executable, the scanner would attempt to extract the data from the stego-image and analyse the headers of the retrieved data to check for “magic numbers” which are unique identifiers describing the

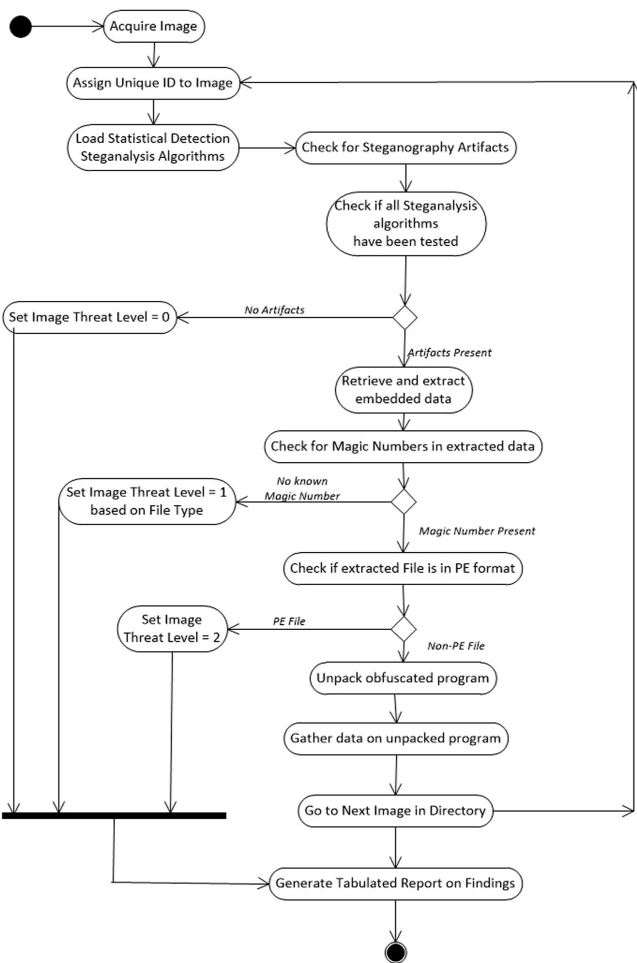


Figure 1.
Proposed scanning
method activity
diagram

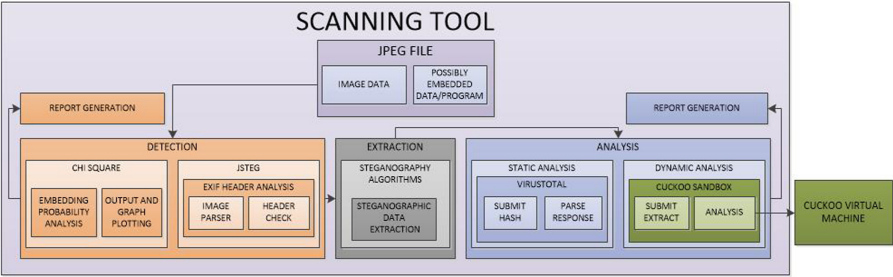


Figure 2.
Scanning process

file type and are located in a file's header. Usually, executable files such as .exe and .dll files are the most suspicious. Where the files are non-executable, a threat level of 1 is assigned signifying an intermediary threat potential. In the case of executable data, commonly known as portable executable (PE) format, the image threat level is set to the highest level of 2. A command is then run on the extracted file, and it will return the kind of packer used on the file as well as its basic properties such as size, and then a report is subsequently generated showing the findings. The scanning process is shown in [Figure 2](#).

The concept of machine learning was introduced into the architecture of the research ([Shoniwa and George, 2015](#)). After the steganalysis has been performed to determine if the embedded files contain malicious software, machine learning was then applied to determine what exact form of malware was embedded in the image. The primary flaw with the previous approach was that it only applied to static analysis of malware in that a signature was generated for each extracted file and then compared to the virus total database of known malware signatures. However, the implementation of machine learning helps overcome that flaw. This is because all the indicators of compromise (IoC) generated by an executing malware are then tested against a known data set of various malware. These IoCs include IP addresses connected to, URLs accessed, files created, registry keys created as well as new paths and folders added. These IoCs are collected from the Cuckoo sandbox which executes the suspected malicious files and produces output which includes these IoCs. A trained data set can then be used to effectively determine which files are indeed malicious even though they may not have known signatures in any antiviral database.

3.3 Machine learning

Machine learning, which endows computers with the “ability to learn without being explicitly programmed” ([Lee et al., 2017](#); [Muñoz, 2012](#)), comes in handy to aid in the automatic detection of malware. ML achieves this by applying techniques that improve performance by reducing false positives ([Gavriliuț et al., 2009](#)). A typical malware detection algorithm is shown in [Figure 3](#). This is supervised learning applied to detection of malware, a classification problem. Examples of benign and malicious executables are passed to train a machine learning algorithm to produce a predictive model to use in the second leg to detect new instances of unknown executables. This can now be categorized as either benign or malicious with the aim of achieving lowest false positive rate. Malware detection techniques are either signature based or anomaly based. This applies to signature-based techniques for malware detection, which depend upon the known signatures of the malware. If a known signature is detected, then the executable is labelled as malicious.

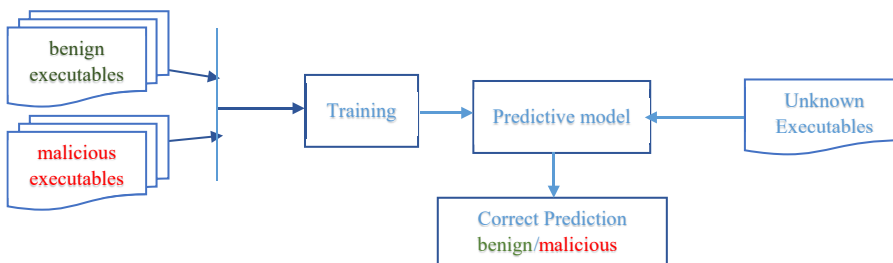


Figure 3.
Supervised machine
learning algorithm for
malware detection

4. A hybrid malware analysis using supervised and unsupervised machine learning

Static malware analysis works well with known malware signatures. However, the challenge comes in the case of zero-day malware detection where there are no known signatures. Anomaly-based methods become more potent as they depend upon analysing and profiling the behaviour observed as anomalous which is malicious or normal with statistical or unsupervised learning methods. A hybrid malware analysis framework shown in Figure 4 which incorporates both dynamic and static analysis applying a stacked model for unsupervised and supervised machine learning, is proposed. Malware authors use techniques such as wrappers, encryption and compression and steganography for obfuscation to hide malware code within executable files so that it evades detection by antivirus software and suppresses any reverse engineering attempt by the malware analysis. In the first part of the framework, executable binaries are presented for analysis in any or a combination of the forms highlighted above. Both dynamic and static analysis is difficult to perform under the circumstances.

In the case of images, the steganalysis process described in Section 3.1 ensues within the virtual environment. For other binaries, the packed executables have to go through the unpacking or de-obfuscation process. This decompresses the executables and also decrypts in the case of encrypted executables. Depending on how the malware has been embedded in the executable, other processes that will happen within the virtual environment include disassembling and decoupling to expose the code and enable dynamic analysis. An unsupervised learning algorithm is then applied to profile the behaviour of the malware. This process applies clustering methods which are able to determine behaviour that is anomalous

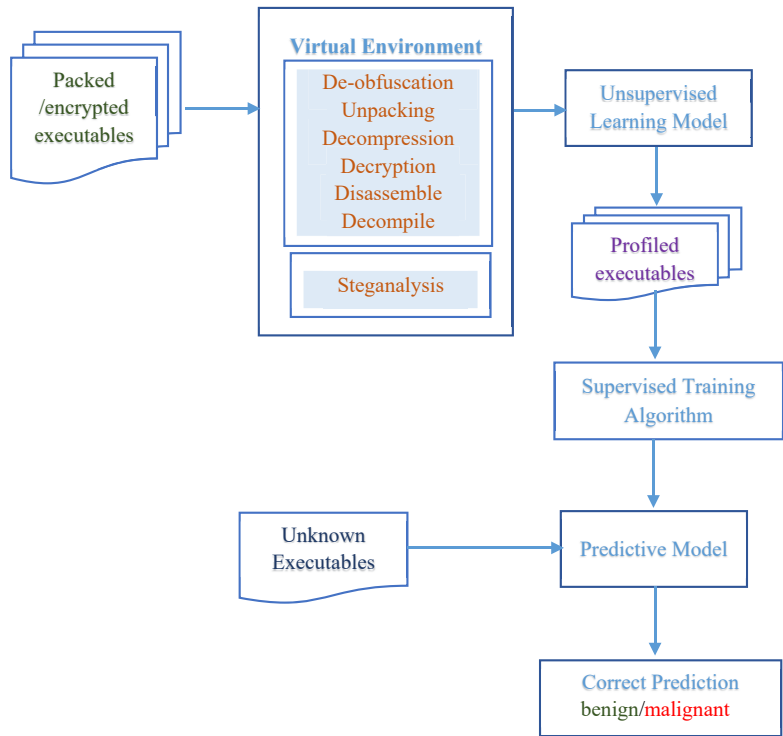


Figure 4.
Hybrid framework for
malware analysis

and that which is normal. It then categorizes the binaries according to the identified patterns of behaviour. The output are the profiled executables. Supervised training is applied with the profiled executables as the training data. This is part of the static analysis process which involves analysing the profiled executable files without executing them. A supervised machine learning model is applied to learn the profiles which are anomalous or normal. Thus, it extracts useful information from the binary to classify it as either benign or malignant. Now new instances of unpacked executables can then be classified as either benign or malicious by feeding it into the predictive model. The model is expected to discover the patterns exposed by the executable binaries from what it has learnt in the training with the profiled executables.

5. Discussion

A method for analysis of images to detect presence of stegomalware is presented. This method is applied to further support a hybrid framework which implements a stacked unsupervised and supervised machine learning model for malware analysis. This framework seeks to profile files based on behaviour that is deemed as normal and that which is anomalous in the presence of malware, catering for both dynamic and static malware analysis of executable binaries using machine learning. unsupervised learning algorithm supports dynamic analysis as executable binaries are unpacked, making it possible to create profiles for the files. This is important as malware behaviour can be identified instead of dependency upon signatures alone which is not adequate to detect zero-day malware instances. The output then passes on to the supervised learning algorithm which supports static analysis whereby the profiled files become input to a predictive model malware detection. This combination of various methods for malware analysis is expected to yield better results towards protection of systems. This may help to detect stealth malware attacks and reduce the risk of extensive damage to the systems.

The proposed hybrid framework is the earliest of efforts on the roadmap by the researchers to contribute towards this important area of cyber security. The current work will culminate in the creation of the scanning tool integrated with supervised and unsupervised machine learning algorithms. In addition to simple machine learning methods, ensemble machine learning and deep learning methods shall also be tested in future work.

6. Conclusion

The research work revealed that, as techniques for introducing malware in systems and evading detection continue to evolve at a fast pace, there is need to exert more effort towards improving defence mechanisms and safeguards as well as detection techniques to identify presence of malware. More stealth detection techniques are being developed to improve on the efficiency thereof. Dynamic and static malware analysis techniques are commonly applied with hybrid techniques, yielding better results by harnessing the strengths of the two types of analysis. Malware authors aim to bypass the defences by applying techniques such as obfuscation, anti-debugging, anti-disassembling, sandbox evasion, antivirus evasion and so forth. A proposed method for de-obfuscation and steganalysis was presented to identify malware embedded in images. A hybrid framework for unsupervised and supervised machine learning was proposed to enable dynamic analysis to be performed in a virtual environment to prevent execution of malware directly in the system which would harm the system. Implementation and validation shall be done in the future works.

References

Baker, K. (2020), "Malware analysis", available at: <https://www.crowdstrike.com/epp-101/malware-analysis/>

- Bashari Rad, B., Masrom, M. and Ibrahim, S. (2012), "Camouflage in malware: from encryption to metamorphism", *International Journal Of Computer Science And Network Security (IJCSNS)*, Vol. 12 No. 8, pp. 74-83, available at: http://paper.ijcsns.org/07_book/201208/20120813.pdf
- Bhojani, N. (2014), "Malware analysis". doi: [10.13140/2.1.4750.6889](https://doi.org/10.13140/2.1.4750.6889).
- Chumachenko, K. (2017), "Machine learning methods for malware detection and classification", available at: <https://core.ac.uk/download/pdf/80994982.pdf>
- Distler, D. (2007), *Malware Analysis: An Introduction*, SANS Institute, available at: <https://sansorg.egnyte.com/dl/7WQkeUbSPW>
- Gavriliuț, D., Cimpoeșu, M., Anton, D. and Ciortuz, L. (2009), "Malware detection using machine learning", *Proceedings of the International Multiconference on Computer Science and Information Technology, IMCSIT '09*, Vol. 4, pp. 735-741, doi: [10.1109/IMCSIT.2009.5352759](https://doi.org/10.1109/IMCSIT.2009.5352759).
- Kendall, K. and Mcmillan, C. (2007), "Practical malware analysis", *Network Security*. doi: [10.1016/S1353-4858\(12\)70109-5](https://doi.org/10.1016/S1353-4858(12)70109-5).
- Komatwar, R. and Kokare, M. (2021), "A survey on malware detection and classification", *Journal of Applied Security Research*, Vol. 16 No. 3, pp. 390-420, doi: [10.1080/19361610.2020.1796162](https://doi.org/10.1080/19361610.2020.1796162).
- Lee, A., Taylor, P. and Kalpathy-Cramer, J. (2017), "Machine learning has arrived!", *Ophthalmology*, Vol. 124 No. 12, pp. 1726-1728, doi: [10.1016/j.ophtha.2017.08.046](https://doi.org/10.1016/j.ophtha.2017.08.046).
- Lengyel, T.K., Maresca, S., Payne, B.D., Webster, G.D., Vogl, S. and Kiayias, A. (2014), "Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system", *ACM International Conference Proceeding Series*, 2014-Decem(December), pp. 386-395, doi: [10.1145/2664243.2664252](https://doi.org/10.1145/2664243.2664252).
- Min, B. and Varadharajan, V. (2014), "Feature-distributed malware attack: risk and defence", in Kutylowski, M. and Vaidya, J. (Eds), *Computer Security - ESORICS 2014*, ESORICS 2014, Lecture Notes in Computer Science, Springer, Cham, Vol. 8713, doi: [10.1007/978-3-319-11212-1_26](https://doi.org/10.1007/978-3-319-11212-1_26).
- Muñoz, A. (2012), "Machine learning and optimization", *Computer Science*, available at: <https://www.semanticscholar.org/paper/Machine-Learning-and-Optimization-Muñoz/7fbba79630b5a09dd66ab13f00c3aefaa56cf268>
- Okane, P., Sezer, S. and McLaughlin, K. (2011), "Obfuscation: the hidden malware", *IEEE Security and Privacy*, Vol. 9 No. 5, pp. 41-47, doi: [10.1109/MSP.2011.98](https://doi.org/10.1109/MSP.2011.98).
- Oktavianto, D. and Muhandianto, I. (2013), *Cuckoo Malware Analysis Analyze Malware Using Cuckoo Sandbox*, Packt Publishing.
- O'Meara, K., Shick, D., Spring, J. and Stoner, E. (2016), "Malware capability development patterns respond to defenses: two case studies executive summary", Software Engineering Institute, Carnegie Mellon University, available at: https://resources.sei.cmu.edu/asset_files/whitepaper/2016_019_001_453290.pdf
- Rautmare, C. (2020), "Visa alert: POS malware attacks persist - BankInfoSecurity", available at: <https://www.bankinfosecurity.com/visa-alert-pos-malware-attacks-persist-a-15126>
- Schultz, M.G., Eskin, E., Zadok, E. and Stolfo, S.J. (2001), "Data mining methods for detection of new malicious executables", *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 38-49, doi: [10.1109/secpri.2001.924286](https://doi.org/10.1109/secpri.2001.924286).
- Sethi, K., Chaudhary, S., Tripathy, B. and Bera, P. (2017), "A novel malware analysis for malware detection and classification using machine learning algorithms", pp. 107-113, doi: [10.1145/3136825.3136883](https://doi.org/10.1145/3136825.3136883).
- Sharif, M., Yegneswaran, V., Saidi, H., Porras, P. and Lee, W. (2008), "Eureka: a framework for enabling static malware analysis", *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5283, pp. 481-500, doi: [10.1007/978-3-540-88313-5_31](https://doi.org/10.1007/978-3-540-88313-5_31).
- Shoniwa, R.T.R. and George, G. (2015), "Design of application to detect images embedded with malicious programs", *International Journal of Science and Research (IJSR)*, Vol. 4 No. 3, pp. 1899-1903, available at: https://www.ijsr.net/get_abstract.php?paper_id=SUB152465

Suarez-Tangil, G., Tapiador, J.E. and Peris-Lopez, P. (2014), "Stegomalware: playing hide and seek with malicious components in smartphone Apps", *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8957, pp. 496-515, doi: [10.1007/978-3-319-16745-9_27](https://doi.org/10.1007/978-3-319-16745-9_27).

Ucci, D., Aniello, L. and Baldoni, R. (2019), "Survey of machine learning techniques for malware analysis", *Computers and Security*, Vol. 81, pp. 123-147, doi: [10.1016/j.cose.2018.11.001](https://doi.org/10.1016/j.cose.2018.11.001).

Corresponding author

Prudence Kadebu can be contacted at: pkadebu@gmail.com