TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Weerarathna Patabendige Samoda Abeydeera 184685IVCM

FILELESS MALWARE DETECTION IN CLOUD USING MACHINE LEARNING TECHNIQUES

Master's Thesis

Supervisor: Alejandro Guerra Manzanares
Cyber Security

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Weerarathna Patabendige Samoda Abeydeera

19.05.2020

Abstract

Fileless threats are on the rise and most recently adopted by a broader range of malware such as ransomware, crypto-mining malware. In this modern era, cloud computing is widely used due to the financial benefits and high availability. Virtualization is the base concept of cloud technology. Due to the prevalence of virtual resources, organizational cloud is one of the most targeted points by cyber attackers. Sophisticated attacks such as fileless threats are most popular among cyber attackers as they are hardly detected by traditional detection solutions. Besides, advanced attacks are capable of altering the exiting detection solutions to evade detection. Therefore, typical antivirus solutions considered as untrusted upon the presence of advance attacks. Machine learning technology widely used in numerous researches as an alternative mechanism to detect malware attacks. In this paper, novel and trusted methodology have been used to detect fileless threats in cloud. Volatile memory dumps collected from virtual machines upon the execution of each malware and legitimate applications. In addition, necessary features extracted for analysis by using the opensource Volatility framework. Sandbox analysis report has been used to obtain additional network-level features related to each program. Thereafter, extracted volatile memory features and network-level features analyzed using random forest machine learning model to predict malicious behaviour of fileless threats. Altogether, 33 features analyzed using machine learning model. As per the analysis, 96% accuracy has been obtained using entire feature set and 83% accuracy achieved using the most important feature. Despite the fact that fileless threats do not leave any traces in the hard disk, still, it has traditional malware features such as create network connections and processes. This analysis showed that fileless threats executes in the memory, leaves traces that allow detecting using feature analysis. Finally, the proposed methodology shows that analysis of different feature, including volatile memory, can be used to distinguish the different state of computer systems and use them to optimize the new trend of fileless malware detection accuracy.

This thesis is written in English and is 61 pages long, including 6 chapters, 8 figures and 11 tables.

Annotatsioon

Failivaba pahavara tuvastamine pilveteenuses kasutades masinõpe tehnikaid

Failideta ohud on tõusuteel ja viimasel ajal on neid laiemalt kasutusele võetud pahavarades, näiteks lunavara ja krüptokaevandamise pahavarad. Sel kaasaegsel ajastul kasutatakse pilvandmetöötlust laialdaselt rahaliste eeliste ja kerge kättesaadavuse tõttu. Pilvetehnoloogia põhikontseptsioon on virtualiseerimine. Virtuaalsete ressursside levimuse tõttu on organisatsiooni pilv küberründajate üks enim sihtpunkte. Keerukad rünnakud, nagu failideta ründed on küberründajate seas kõige populaarsemad, kuna traditsiooniliste tuvastamislahendustega neid peaaegu ei saa tuvastada. Lisaks on täiustatud rünnakud võimelised muutma olemasolevaid avastamislahendusi, et vältida tuvastamist. Seetõttu peetakse tüüpilisi viirusetõrjelahendusi eelnevalt mainitud rünnakute esinemise korral ebausaldusväärseteks. Uuritud on kõigi pahavara ja õigustatud rakenduste käivitamisel virtuaalsetest masinatest kogutud hävimälu mälumahud. Lisaks on analüüsimiseks vajalikud omadused eraldatud, kasutades avatud lähtekoodiga Volatility raamistikku. Iga programmiga seotud täiendavate võrgutaseme tunnuste saamiseks on kasutatud liivakasti analüüsi aruannet. Seejärel analüüsiti eraldatud hävimälu mäluomadusi ja võrgutasandi omadusi, kasutades „random forrest“ õppimismudelit, et ennustada failideta ohtude pahatahtlikku käitumist. Kokku analüüsiti masinõppe mudeli abil 33 omadust. Analüüsi kohaselt on kogu tunnuste komplekti abil saavutatud 96% täpsus ja kõige olulisema omaduse analüüsimisel saadud 83% täpsus. Vaatamata asjaolule, et failivabad ohud ei jäta kõvakettale jälgi, on sellel siiski traditsioonilised pahavarafunktsioonid, näiteks võrguühenduste loomine ja protsesside käivitamine. See analüüs näitas, et failivabad ohud käivituvad mälus ja jätavad jälgi, mis võimaldavad erinevate omaduste analüüsi abil neid tuvastada. Kokkuvõtteks näitab pakutud metoodika, et erinevate omaduste, sealhulgas hävimälu analüüsi abil saab arvutisüsteemide erinevat olekut tuvastada ja kasutada neid failivaba pahavara tuvastamise tehnikate täpsemaks optimeerimiseks.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 61 leheküljel, 6 peatükki, 8 joonist, 11 tabelit.

List of abbreviations and terms

LoL	*Living Off the Land*
WinRM	*Windows Remote Management*
RAM	*Random Access Memory*
WMI	*Windows Management Instrumentation*
SMB	*Server Message Block*
NAT	*Network Address Translation*
VM	*Virtual Machine*
GB	*GigaByte*
C&C	*Command and Control*
HIDS	*Host-Based Intrusion Detection System*
DLL	*Dynamic Link Library*
PCA	*Principal Component Analysis*

Table of contents

Author's declaration of originality 2
Abstract 3
List of abbreviations and terms 5
Table of contents 6
List of figures 8
List of tables 9
1 Introduction 10
2 Background Information 13
2.1 Fileless malware 13
2.1.1 Detection of fileless threats, ... 16
2.2 Virtualization and cloud computing 16
2.2.1 Virtual machine 17
2.3 Volatile memory forensics 17
2.4 Machine Learning 18
2.4.1 Machine learning classification algorithms 19
2.4.2 K-Fold cross-validation 20
2.4.3 Confusion matrix 20
2.4.4 Feature selection 22
2.4.5 Correlation matrix 23
2.4.6 Feature importance 23
2.4.7 Principal component analysis 23
3 Related work 25
4 Methodology 30
4.1 Acquisition of volatile memory 30
4.2 Data collection 32
4.2.1 Fileless malware samples 34
4.3 Feature Extraction 35
4.3.1 Volatility Framework 35

4.3.2 Feature extraction for dataset creation 35
4.3.3 Set of features 37
4.4 Data processing using machine learning algorithm 38
4.4.1 Classification machine learning algorithms 38
5 Evaluation 39
5.1 K-Fold cross-validation 39
5.2 Random Forest 40
5.3 Fileless threat detection: performance 40
5.4 Correlation matrix 41
5.5 Analysis of feature importance 42
5.6 Principal Component Analysis 44
5.6.1 Performance evaluation using PCA 46
6 Conclusion 48
References 50
Appendix 1 – Data set used for the analysis 55

List of figures

Figure 1: Hyper-V virtualization infrastructure..... 30
Figure 2: Process flow of collecting snapshots 31
Figure 3: Snapshot creating process 33
Figure 4: Feature Extraction Process 36
Figure 5: Correlation between features of the dataset 41
Figure 6: Top ten features according to feature importance 42
Figure 7: 2D principal component graph 45
Figure 8: Cumulative explained variance with number of components 47

List of tables

Table 1: Confusion matrix 21
Table 2: Virtual machine specifications 32
Table 3: Programs executed in virtual machines 34
Table 4: List of Volatility Plugins 36
Table 5: List of features from Volatility 38
Table 6: List of features from Sandbox report 38
Table 7: K-Fold cross-validation performance 39
Table 8: Confusion matrix derived from the model 40
Table 9: Performance-based on selected features 44
Table 10: Accuracy based on PCA model 46
Table 11: Accuracy based on the selected number of principal components 47

1 Introduction

Cloud computing is one of the widely used concepts in the modern era. Cloud computing enables end-user to access computer system resources and data via the internet. This is one of the key features that emphasize cloud computing from traditional computer system infrastructure. Most organizations deploy their computer infrastructure in the cloud to ensure high availability. Virtualization is one of the key concepts in cloud computing. Virtual resources such as virtual servers, virtual switches and virtual network devices widely used to provide more accurate and feasible service while providing high availability for the organization computer resources. Based on the Flexera survey on cloud computing trends [1], 84% of organizations use multi-cloud (including private, public and hybrid cloud environment). Furthermore, as per the survey, this use of cloud trend has drastically increased compared to the previous year.

As cloud computing is one of the most popular and widely used services among the organizations, cyber-attacks are more frequent for this platform. Most cyber attackers are more focus on servers and data storages deployed in the cloud platform as critical data is one of their primary concerns. Upon the advance and sophisticated attacks, organization may not even discover the compromised assets despite the presence of endpoint detection platforms such as antivirus. In addition, cloud services introduce new challenges to security organizations as one compromise virtual machines may lead to a huge security breach [2].

Fileless or non-malware attacks are on the rise and most recently adopted by a broader range of malware such as ransomware, crypto-mining malware. Fileless threats are one of the advance attack types among the malware trends history. Fileless malware differs from traditional malware as it doesn’t require to install any malicious software to infect victim machine [3]. Additionally, unlike typical malware, fileless threats don’t write anything to disk, but it writes to the RAM and evade detection of traditional antivirus security [4]. Customer service and managerial staff at retailers targeted by fileless

malware campaign called August from TA530 during November 2016 [5]. August steal credentials and sensitive data from the infected computer.

As per the existing security solutions, prevention of fileless attacks are relies upon endpoint protection solutions such as antivirus. However, these solutions have limited capabilities to detect unknown attacks and sophisticated attacks, as most of them are relying on signature matching. As fileless threats do not leave any malicious files in the system and use only legitimate Windows components to carry out the attack, endpoint detection software cannot generate a signature definition to detect the specific threat. Signatures refer to data explained the characteristics of malware. This data can be used to determine whether the victim machine or software application contains malware [6]. This makes a challenge, as the antivirus doesn't know what to look for. Since fileless attacks use living off the land (LoL) tactics, it is more difficult to detect by traditional detection engines. Cyber attackers tend to use fileless malware to carry out the attack as it has the capability to evade most of the traditional detection methods. Therefore, it is not surprising that most organizations face difficulty to detect and handle these types of incidents. As per the Symantec threat intel report, attackers use LoL tactics and preinstalled system tools to evade detection and hide malicious activities during the attack [7].

As the report by ThreatVector, even though fileless attacks are relatively rare, existing attacks are advance and sophisticated compare to the traditional attacks [8]. Fileless attacks get its name by not leaving files on disk, but it is residing in the memory and execute commands and run tools which are legitimate and already installed in the victim machine. For instance, often fileless threat use PowerShell to carry out the attacks. PowerShell is a powerful scripting language and can use for various reasons such as make remote connections, invoke other applications and process etc. "Kovter" is one of sophisticated fileless attack which began leveraging in 2016. "Kovter" is a pervasive click-fraud trojan that utilizes a fileless persistence mechanism to build up the attack and evade traditional detection mechanisms [9].

Apart from that, most of the sophisticated advance malware capable of altering or interfere with the detection functionality exiting antivirus solutions to avoid detection or trigger alarms/alerts. This considers as another significant weakness of existing detection solution that is available on the market.

As per the insufficient accuracy of detecting fileless threats by using traditional signature matching detection solutions, security researchers work on different detection logics such as machine learning and behavioural analysis. Behaviour pattern analysis and machine learning detection techniques have proven that those are accurate and efficient enough to detect advance malware threats such as ransomware [10]. However, features in volatile memory have not been tested with machine learning algorithms to detect most emerging fileless threats.

In this study, we focus on presenting a novel methodology to detect most emerging fileless threats in the organization's private cloud by analyzing volatile memory using machine learning algorithms. Volatile memory dumps have been extracted from the virtual machines. Volatility framework has been used to extract the meta-features from memory dumps and leverage them using machine learning algorithms to detect fileless malware threats. In the proposed methodology, memory extraction, analysis and detections are mainly operating on hypervisor level. Therefore, malware running inside the virtual machine cannot interfere or evade the analyzing and detection functionality. For instance, we analyze the volatility memory dumps taken from victim machines and malware that is running on the machine cannot detect any underground operation such as taking the snapshot as it executes in the hypervisor level.

Most of the traditional threat detection solutions detect malware upon the results of static or dynamic analysis of suspicious file/s running in the victim machine [11]. Thus, the existing solution is not capable of detecting fileless threats as it does not create any malicious files to carry out the attack. To overcome this limitation, the proposed methodology evaluates the volatile memory using machine learning algorithms to distinguish the usual and unusual behaviour of the system. In this study, infrastructure has been mainly deployed on Microsoft Hyper-V virtualization environment and VMware virtual environment. Additionally, the proposed methodology has been tested against Windows 7, Windows 10 using a list of known fileless attacks.

2 Background Information

Important background information regards to the fileless threats, virtualization, cloud computing, virtual environment, and volatile memory forensics have been expressed in this section.

2.1 Fileless malware

Fileless malware is one of the most emerging threats in history. According to the Cybereason [12], unlike traditional malware, fileless threats do not require the threat actor to install any software on the victim machine. Instead, fileless threats leverage tools and application that are in-built in Windows, such as PowerShell to take out the malicious actions. Since these tools and related operations are usually trusted by all detection software, it is particularly a challenge to differentiate and distinguish these types of malicious activities from legitimate activities [12]. Apart from that, fileless infection does not store anything in the hard drive; instead, it goes straight into the Random-Access Memory (RAM) which leaves no traces behind. However, as this type of malware operates in volatile memory, the operation can be terminated upon the system reboot. Same as most advance attacks today, fileless attacks also use social engineering tactics for the entry points such as malicious link in the email which leads the user to download the word document with a malicious macro. According to Microsoft fileless malware article, there are three primary categories of fileless attack types [13].

- **No file activity performed:**
  Completely fileless malware that is not writing any file on the disk. For instance, during SMB EternalBlue vulnerability also refer as SMB MS17-010 vulnerability exploitation, victim machine receives malicious network packets that exploit SMB vulnerability, leads to install DoublePulsar backdoor and end up reside only in the kernel memory. EternalBlue is a powerful cyberattack exploit released by well-known hacker group known as Shadow Brokers. This exploit kit uses SMB MS17-010 vulnerability to exploit unpatched Microsoft systems. Importantly during this attack, no “write to file” operation involved.

- **Indirect file activity:**

  This type does not directly write files on the system but ends up using files indirectly. For example, attackers use PowerShell to execute malicious commands and configure Windows management instrumentation (WMI) to run commands periodically. WMI is a Microsoft implementation which allows to obtain data from a remote computer. Also, this compatible with WMI scripting facility to automate administrative tasks. WMI mainly developed by Microsoft for administrative purpose. However, attackers misuse WMI to interact with remote machines and perform malicious operations such as information gathering and remote execution [14].

- **Files required to operate:**

  This malware contains partial fileless characteristics as sometimes files involve delivering the payload. An example for this scenario is "Kovter" fileless malware which leaves open a backdoor in the victim machine by executing malicious script through the legitimate tool mshta.exe. Payload for the "Kovter" can receive via word document with a malicious macro.

Apart from that, there are some other methods that fileless threat can arrive into the machine such as via exploit, through compromised hardware, or via execution of script/application. Some advance fileless threats infect into the master boot record (MBR), which bootstrap the execution of malware even before load the operation system [13]. MBR is a special section of the disk that loads operating system information and hard disk partition information. Adversaries overwrite this special section to invoke malicious code upon the start-up of normal bootloader [15]. According to the McAfee threat report, fileless threats are typically used for lateral movement and gaining fist level access to the system [16]. Lateral movement refers to a set of techniques and tools that attackers use to progressively spread throughout the victim network and obtain higher privilege access [17]. Creative cybercriminals aim at four main aspects by developing fileless malware [18],

- **Stealth:** The ability to hide malicious activities to evade the detection
- **Privilege escalation:** The ability to take over the administrative access by exploiting system vulnerabilities

- **Gather information:** Harvest information and data about victim network for later usage
- **Persistence:** Malware remain undetected

Fileless threat mainly differs from typical malware is where the advance techniques of persistence and stealth mechanisms take place. Attacker also develops fileless threats to first store the payload on the RAM to gain persistence. The main reason to hide the payload in the RAM instead of hard disk, is to evade the detection. There are mainly three techniques that fileless threats follow to remain undetected [18],

- **Memory-resident malware:** Load malicious code into the memory and use windows authentic programs to carry out the attack. For instance, use Windows legitimate process mshta.exe to execute HTML applications and scripts. This gains less attention from the endpoint sensors since all the legitimate tools involved in the attacking process. The primary focus of this is to remain undetected throughout the attack.

- **Rootkits:** Rootkits often hide behind the kernel; thus, it maintains the persistence upon system restart and antivirus scans. As the name implies, rootkit contains a set of tools that ability to gain administrative access without informing the user. Rootkit is able to track everything on the victim machine. Once after delivered the payload to the victim machine, dropper installs rootkit. In most fileless threats, dropper delivered to the victim machine as a word document with an embedded malicious macro. These rootkits are difficult to detect as they are mostly executing as child processes under legitimate parent processes.

- **Windows registry malware:** Modern fileless threats are capable of resides in the Windows registry files. Windows registry stores the low-level settings related to the operating system and certain applications. During this type of attack, malware file executes in the registry file and then self-destruct the malware file upon completion of the malicious task.

2.1.1 Detection of fileless threats,

Currently, software-based solutions are available to detect malware based on machine behaviour. Unlike traditional signature matching solutions, behavioural based analysis based threat detection tools identify malicious, suspicious behaviour by analyzing differences in usual everyday activities in computer system to proactively mitigate cyberattacks before the attackers fully execute their destructive plans. For instance, *net.exe* executed in human resource manager computer is a suspicious behavioural activity. Genuine *net.exe* is a legitimate software component in Windows system that used to control network connections, services, users and groups. Use of net.exe in human resource manager PC may indicate compromised scenario where attacker performing local reconnaissance, enumerating accounts to identify high-privilege targets [19]. However, most of the detection's software can be manipulated by advance malware and turn off the alerts. For example, Regsvr32.exe is a legitimate command-line utility used to register and unregister DLLs files on Windows machines. *Regsvr32.exe* can also use to execute arbitrary binaries. Malicious actors take advantage of this functionality to register a malicious program to avoid triggering security alerts [20]. Apart from that, since fileless threats use LoL binaries, leave no suspicious behaviour while executing the attack. Same as typical malware, fileless threats also built to use an operating system or application vulnerability to exploit the system. For instance, EternalBlue exploits server message block (SMB) vulnerability in Microsoft. As referenced before, EternalBlue is a critical exploit released by well-known hacker group. WannaCry ransomware use this exploit to attack Windows computers which contain a vulnerable version of SMB [21]. McAfee recommends regular patching and system hardening to reduce the risk [22].

2.2 Virtualization and cloud computing

Virtualization allows users to create a computing environment or IT resources by utilizing traditional bounded hardware [23]. This enables organizations to build up multiple virtual resources by partitioning a single physical computer. Each virtual resource can be used for an independent task with different features such as operating system while sharing the same hardware resources [24]. Virtualization is the fundamental technology that powerup cloud computing [25]. Cloud computing is one of the famous IT trends in the modern era. Cloud computing enables to deliver different services via the internet. Due to the cost

benefits, high availability and efficiency cloud computing are more popular among organizations [26]. Virtualization is the underground base concept of cloud computing.

2.2.1 Virtual machine

Virtual machine is an emulation of computer that provides the same functionality as a physical computer. Virtual machine implements on top of specialized hardware and software combination. Same physical resources can be shared among multiple virtual machines. This feature is one of the key advantages of virtual machines. Virtual machines deploy on top of the hypervisor, which is mainly responsible for allocating physical resources to virtual machines. Besides, the same hypervisor can manage multiple virtual resources and ensure that machines are completely isolated from each other [27].

2.3 Volatile memory forensics

"Computer forensics is a method of extracting and preserving data from a computer so that, it can be used in a criminal proceeding as evidence" [28]. The exact objective of computer forensic it to uncover and determine the truth behind the digital crime. Digital forensic plays a significant role in cyber-attacks and digital crime investigations.

Computer mainly stores its data in hard drive (non-volatile memory) and volatile memory. Hence, security investigators usually analyze both hard drive and volatile memory during the forensic investigation to reveal the scenario. Volatile memory keeps instructions and data related to the running programmes [29]. Volatile data will be lost upon the reboot or machine power off. Some advance malware leaves no data in hard drive but resides in volatile memory. Therefore, volatile memory analysis plays a significant role during the investigation of such sophisticated attacks. For instance, essential data such as keys used for encrypting volumes during ransomware attacks (TrueCrypt, VeraCrypt, BitLocker) can be retrieved by analyzing volatile memory [30]. Mainly there are two approaches in volatile memory forensics which are live response and memory imaging. During the live response method, investigator access to the device using remote shell connection and do in-depth analysis and take immediate incident response actions to promptly mitigate identified threats [31]. However, there are certain disadvantages in this method, such as if malware running in the target machine, it will manipulate the data collection process. During the memory imaging approach, volatile memory image will be extracted from the computer and store for analysis. The main advantage of this method is not changing the

state of the system. The image remains as same before and after the analysis. Additionally, memory imaging analysis can be more trusted than live response as malware cannot manipulate the analysis process.

In this study, memory imaging approach has been used to analyze the volatile memory as it is more trusted and accurate. Also, infrastructure has been deployed in the Hyper-V and VMware environment. We used their inbuilt snapshot feature to take fresh memory image from virtual machines.

2.4 Machine Learning

Machine learning is a method of teaching computers to automate certain operations based on data analysis [32]. Machine learning is mainly used to predict and forecast certain features and parameters, also called supervised learning. For example, machine learning algorithms can be trained to forecast traffic pattern for busy intersection by analyzing certain data about past traffic pattern. The accuracy of the prediction depends on multiple parameters such as dataset, machine learning algorithm etc. Machine learning algorithms use data set to find a pattern of distribution to make better decisions and predictions. There are mainly two types of machine learning methods, which are supervised learning and unsupervised learning. Supervised learning trained the machine learning model using data and correct labels. This allows the machine learning model to map the data with correspondent labels or classes and make the prediction based on data provided as input. However, inappropriate data feed to the model could give inaccurate results in supervised learning method. Supervised learning can further group into two main approaches which are, classification and regression. Classification use when the desired output is class or category. For instance, machine learning model for detect spam email. In this case, two categories are spam and not-spam. Supervised classification model is appropriate for such scenarios. Regression approach uses when the expected output value is variable such as weight or dollars. For instance, machine learning model uses to predict the stock market price for next month. In unsupervised learning, only the data feed to the model without any labels. The expectation of unsupervised learning is to build its own structure based on the input data [33]. Main disadvantages of unsupervised learning are computationally complex and less accurate as it is not trained with labels. In this study, we used meta-

features of volatile memory as a dataset to train the machine learning algorithm and distinguish legitimate and malicious fileless threats.

2.4.1 Machine learning classification algorithms

Classification is a supervised learning technique that allows to categorize data into appropriate classes. This process starts with training the model with test data and finally predict the unseen classes according to input data. In this study, decision tree, random forest, k-nearest neighbour and support vector machine classification models accuracies have been evaluated with our dataset.

- Decision Tree

  Decision tree breaks down the data set into subsets. This typically consists of three main elements which are, root node, branches and leaf node. Each node represents a feature while each branch represents a decision, and each leaf represents an output [34]. Root node represents the final decision based on the data. Besides, using the decision tree, it is easy and accurate to understand the most significant variables and the relationship between features. This considered as one of the major advantages of the decision tree. Besides that, overfitting considered as one of the major limitations in the decision tree. Overfitting explained briefly in section 5.1.

- Random Forest

  Random forest contains large number of individual decision trees. Each individual tree predicts the class. The highest number of prediction class become the final prediction result. More about random forest explained in section 5.2.

- K-Nearest Neighbour

  k-Nearest Neighbour also known as kNN is a supervised learning algorithm which relies on labels in the dataset. k-Nearest Neighbour put all the training data into n-dimensional space. Unknown data class predicts based on the closest k-number of neighbours, and the most common class take as the final value. In kNN, classification mainly based upon the assumption of similar data points exist close to each other [35].

- Support Vector Machines

SVM is a supervised learning algorithm used for classification and regression problems in machine learning. SVM model basically denotes a different set of classes in a hyperplane in multidimensional space [36]. Hyperplane refers to decision boundaries that classify data points in the graph. So, data points belong to each side label with the corresponding class.

2.4.2 K-Fold cross-validation

Machine learning data set can be divided into two main sets, training set and testing test. Training set used to train the model while testing test used to evaluate the result of the training set. Usually, 70% of the dataset used to train the model and 30% of data set to use as testing dataset to evaluate the trained model. K-Fold cross-validation allows to compare the different machine learning classification algorithms to pick up the most suitable method. In this k-fold cross-validation, the data set divide into k number of sets and then train the module using (K-1) number of subsets and the remaining set will be used to evaluate the trained model. This process iterates K number of times with different subset as testing module [37]. This method allows to use each subset for validation exactly once while (K-1) number of subsets use as the training set. The average of the result after K number of iterations take as K-fold cross-validation.

2.4.3 Confusion matrix

This section describes the effectiveness of apply random forest to the training dataset. Confusion matrix is a method of evaluating the performance of the machine learning classification algorithm [38]. As the name implies, confusion matrix allows to visualize the confusion between classes predicted by the selected classification algorithm. Moreover, confusion matrix summarizes the number of correct and incorrect predictions made by classification algorithm. The key advantage confusion matrix is, it just not only provides insight on correct and incorrect predictions, but it gives the type of errors also. Example for confusion matrix as follows. Note that, table rows represent actual classes while columns represent predicted classes.

		Predicted class	
		Malware	**Legitimate**
Actual class	**Malware**	True Positive (TP)	False Negative (FN)
	Legitimate	False Positive (FP)	True Negative (TN)

Table 1: Confusion matrix

As per the confusion matrix table, true positive and true negative states indicate the correct prediction of the machine learning model. True positive (TP) illustrates the state where the actual malware program predicted as malware by the machine learning model. True negative (TN) is the state where legitimate application classified as legitimate by the machine learning model. In other words, these TP and TN states illustrate the correct prediction of the machine learning model. False-negative (FN) denotes actual malware program that predicted as legitimate and false positive (FP) indicate legitimate application predicted as malware by machine learning model. In data science, false-positive known as *Type I Error* and false-negative called *Type II Error* [39].

Different optimization methods used to reduce these two types of error. In this study, an optimized machine learning algorithm has been used to reduce these two types of errors to increase the accuracy of detecting threats. Besides, *Precision* and *Recall* used to evaluate the accuracy of the detection model. *Precision* is ideal to use when the weight and cost of false-positive are high.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

This is equal to true positives count divided by all positive prediction count. As name implies, *precision* equation calculates how precise the model is by getting the ratio of correct positive predictions to count out of total positive predictions [40].

In this research, we have calculated the *precision* with the below characteristics. Python *scikit-learn* inbuilt method has been used for the calculation.

Recall method is ideal when the high weight and cost associated with false negative. *Recall* can be calculated using the below equation:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

As equation describes, *Recall* calculates the ratio between the number of correctly predicted positive predictions and the number of overall all positives. In this study, *Recall* can be used to calculate how many malicious files detected from total malicious application dataset by using our trained machine learning classification model.

Ideally, accuracy is a fraction of, number of correct predictions out of the total number of predictions. This is not a good approach to follow if the data set is not balanced. Accuracy can be calculated using the below equation,

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

These *precisions* and *recall* methods are individually suitable to evaluate the accuracy in different aspects under different characteristics. However, a good machine learning model should perform well in both situations. F1-Score is the combination of both precision and recall, which balance the two approaches [40]. Furthermore, "*F1-Score* is the harmonic mean of this precision and recall" [41]. This can be calculated using the below formula,

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.4.4 Feature selection

Feature selection is the selection of features or attributes in the dataset that are more relevant to the prediction of classes [42]. Feature selection method is useful to detect and avoid redundant and irrelevant unnecessary features that do not contribute to the final accuracy of the model. Also, this helps the model to run faster as it significantly reduces the data set. There are three main models in the feature selection [42],

- Filter method: Each feature in the data set ranked by using statistical measuring method. Feature removal decision can be taken based on the ranked score.
- Wrapper method: Evaluate different combination of features with other combinations. Machine learning model used to evaluate the combinations of features. Assign score for combinations based on the accuracy of the model.
- Embedded method: Determine which feature and how much it contributed to the model accuracy.

2.4.5 Correlation matrix

In data science, correlation matrix is a table that shows the measure of the correlation coefficient between features/variables. Correlation coefficient calculates the strength of the relationship between two variables [43]. This strength value, or in other words, correlation coefficient, varies within the range of (-1.0) to (1.0). In this range, -1.0 denotes the perfect negative correlation, and 1.0 indicates the perfect positive correlation. Any value in the correlation matrix that is greater than zero represents a positive relationship, while less than zero indicate a negative relationship. In the negative correlation, the involvement of two variables/features moves in opposite direction. For instance, when variable A increases, variable B will decrease if both values have a negative correlation and vice versa in positive correlation [44]. Also, if the correlation coefficient value is zero, it indicates that there is no linear relationship between two variables, but it can be some other type of relationship such as quadratic.

2.4.6 Feature importance

Generally, in feature selection, the main functionality is removing unnecessary features then re-train the model and compare the accuracy with baseline. However, in feature importance approach is little bit different from feature selection method as feature importance choose the best set of features that contribute to the accuracy. However, both models achieve the same goal from different directions which is to remove unnecessary features and increase the efficiency of the model. Python *feature_importance_* library allows to evaluate the relative importance of each element.

2.4.7 Principal component analysis

Principal component analysis (PCA) is an unsupervised machine learning method that is used for dimensionality reduction. Dimensionality is the number of variables in the data set. Model overfitting is a primary problem associated with high dimensionality. Overfitting refers to when a model gets trained with too much of data including noisy data, the model learned from those inaccurate data entries. In this case, the model does not classify data accurately because of inaccurate learning data. This caused overfitting in machine learning [45]. This mainly reduces generalization in the machine learning model. Generalization refers to the ability of model to adapt for new, unseen data from the same distribution [46]. High dimension makes overfitting in the model, which increases the generalization error [47]. Furthermore, high dimension data extremely

complex to analyze and process due to some feature's incompatibility. By applying PCA to the data set, it is able to reduce dimensionality by only selecting the important features that capture maximum information about the dataset. Also, by lowering dimensionality PCA allows to speedup the data processing and model training [48].

3 Related work

This section explains the previous work related to the fileless malware detection, cloud computing, virtualization and memory forensics. A literature search performed in selected digital libraries such as ACM digital library, IEEE Explore and web search in google scholar. Search keywords such as fileless, file-less, malware-less, malware less have been used to find the related papers. Manual web search used to find the relevant articles and publications from reputed sources. Snowballing method has been used to find out associated researches.

Carlin *et al.* [49] presented research related to dynamic opcode analysis tool to detect fileless browser-based crypto-mining engines. Based on the author, this is the first dynamic opcode analysis for fileless crypto mining using machine learning. For the experimental setup, dedicated machines were used for the experiments, using a fresh image of Windows 7 64-bit, an Intel Celeron 2.90GHz G3930 CPU and 4GB RAM. The open-source OllyDbg v2 debugger used to trace the dynamic opcodes of each runtime, with StrongOD v0.4.8.892 used to cloak the running debugger. Firefox 54 was used as the browser to execute all HTML files. Bespoke Python scripts were employed to automate the execution process. This methodology was followed constantly during the dataset creation process. Random Forest (RF) machine learning model used with WEKA 3.9, for all classification tasks. As per the result presented in this paper, dynamic opcode analysis is an effective way to detect crypto-mining behaviours in the browser and detect malicious activities. Furthermore, this model can be used to determine crypto mining sites, weaponized benign sites, de-weaponized crypto mining sites and legitimate real-world sites. However, this study only addressed the browser-based crypto-mining malware for opcode analysis.

Nahmias *et al.* [50] presented a solution which is named as “Trustsign” a novel, trusted automatic malware signature generation method based on features transferred from a pre-trained VGG-19 model. VGG-19 is a convolutional neural network. This is one of the well-known models to classify the real-world images. “Trustsign” is trusted and fully capable of signing file-less malware. “Trustsign” produces signature based on the malicious process in the volatile memory (rather than the file representation on the local drive), thus overcoming packing and obfuscation techniques which are usually applied by

malware developers in order to avoid detection. Moreover, TrustSign does not require feature engineering or time-consuming model training, as it leverages transfer learning, thus minimizing the critical time interval between the malware's analysis and the distribution of its signature. "TrustSign" leverages virtualization to produce signatures based on the presence of a malicious process in the volatile memory in a trusted manner. To achieve trusted analysis, execute the malware on a designated virtual machine. Then, while the malicious process is active, snapshot has been taken from the system's volatile memory by querying the hypervisor. TrustSign's methodology is trusted since it analyses the volatile memory dump taken from a virtual machine, and the malware that is running on the machine cannot evade, interfere with, or shut down TrustSign. However, as per the author, TrustSign encounters difficulty in producing a signature for malware that injects code into memory. Additionally, there is no such malware program involved for analysis in this study. However, in this study, we overcome these limitations by addressing fileless threats that injects malicious code into memory, such as *Astaroth* attack.

Cohen *et al.* [11] Most companies are migrating to cloud infrastructure, and due to that reason, most attackers target virtual servers in the cloud. Most of the antivirus software cannot detect advance threats, and most of the advance prevention tools not yet developed to secure the cloud environment. This study presented a methodology to detect known and unknown ransomware in virtual machines on an organization's private cloud using volatile memory. For the implementation, VMware vSphere has been used to create and manage virtual machines for evaluation. This allows to take snapshots of the VM and easily extract the volatile memory. Using these snapshots, collected memory dumps after the execution of multiple legitimate and malicious programs. Conducted analysis on volatile memory dumps that are taken from virtual machines. Volatility framework has been used to extract general descriptive meta-features from memory dumps. Thereafter leveraged the meta-features using machine learning algorithm to capture unknown ransomware. The main limitation of this study is only ransomware attacks have been encountered for the analysis. Apart from that, only five ransomware attacks have been involved in the analysis. In our study, we addressed more than fifteen fileless threats in the analysis.

Hăjmăşan *et al.* [51] presented a scoring mechanism for dynamic evaluation of the behaviour of potential malware processes. Most of the dynamic behaviour malware detection platforms are based on learning method which prone to more false positives and

not accurate enough for real-time detection. This study presented a scoring engine based on dynamic behaviour evaluation of the machine. Proposed methodology monitor the actions performed by processes, kernel and assign a separate score for each action based on the proposed scoring mechanism. The primary theoretical contribution of this research is the scoring mechanism for dynamic behaviour evaluation of malware. Proposed solution tested against with malicious files, processes running in the machines. Detection test performed using an automated analysis platform that executed the samples in multiple virtual machines with the Windows 7 x64 OS installed. By analyzing the results, 14949 malicious files detected and 2120 were not. This can be translated into a detection rate of 87.57%. The tested files have been obtained from various sources like spam email attachments, URLs used to spread malware and infected web sites. However, based on the study result, this proposed solution has a significant amount of false-positive rate, which is more than 13%. Apart from that, most of the advance attacks are capable of manipulating the monitoring systems. Hence, this solution monitors each action performed by the kernel module and process it may not be a trusted solution against sophisticated attacks.

Handaya *et al.* [52] proposed a methodology to detect fileless cryptocurrency mining malware using a machine learning approach. Proposed methodology analyzes benign and malicious cryptocurrency malware using different machine learning models such as random forest, support vector machine and kNN. This research mainly focused on Monero crypto-mining malware during the analysis. For the analysis, EMBER data set has been extracted upon execution of each malware and legitimate programs in the machine. EMBER refers to benchmark dataset used for train machine learning models in order to statically detect malicious Windows executable files [53]. This dataset contains features extracted from binary files and malicious and benign training samples. As per the analysis, the main limitation of this research is they analyzed only Monero crypto-mining malware for the evaluation.

Gadgil *et al.* [54] presented an analytical approach to hunting advance volatile threats using memory forensics. Memory forensics become critical as memory contains many forensic artifacts that cannot obtain from traditional disk forensics. This study mainly checks for the indicator of compromise (IOC's) exist in the memory to detect malware/threats in the memory. Indicator of compromise represents data, log entries or files that are related to the malicious activities. In this study, infected machines analyzed

with the Volatility framework using a different set of plugins. Different plugins have been categorized to identify different characteristics. For example, to identify rouge process, it is able to use pslist, pstree plugins. These two plugins list down the process list in the memory so that analyst can check for unusual suspicious processes. Besides, examine the open connections to verify the malicious connections.

Tsuda *et al*. [55] proposed a HIDS based on process generation pattern. Proposed HIDS system, at first, periodically collects lists of active processes from hosts on managed networks. The system extracts process paths from process trees which the system builds by using the collected lists. Finally, the system detects anomaly processes considering process paths' uniqueness and lifetime. In order to find anomaly behaviours on well-managed networks, it is effective to observe changes in executing applications and processes. The proposed system has implemented in an actual organization to evaluate machines. It has collected 2,403,230 process paths in total from 498 hosts for two months. HIDS system could extract 38 anomaly processes. Most of the processes are created by benign applications which were used for maintenance and daily works. Among the anomaly processes, there is a PowerShell process created by a macro in Microsoft Excel. It also detected by using an antivirus software running on the organization. The other 18 PowerShell processes were benign, which were related to updating programs for maintaining hosts, and the anti-virus software. Based on the author, HIDS system should be improved using the sanctioned and unsanctioned application method to minimize the false positives.

As per the literature, there are certain amount studies have been conducted on malware detection based on dynamic analysis and machine learning. Among them, most of the researches focused on unique malware type such ransomware, fileless crypto-mining malware and dynamic analysis based on opcode analysis, volatile memory analysis and signature matching etc. There is no proper study found related to the detection of fileless threats using machine learning model. Additionally, no proper study found to provide a detailed analysis of how to select memory features to detect such advanced attacks. In this paper, we presented a novel and trusted methodology to detect fileless malware in virtual machines and a private cloud. Additionally, we presented a set of feature list that can be used for analysis and correlation characteristics between those features in order to detect fileless threats. As mentioned in the literature, Volatile memory feature analysis has been used in previous studies for ransomware detection. However, in this study, we

used a combination of both volatile memory and network-level feature analysis using machine learning model to detect fileless threats in the cloud. Additionally, proposed methodology mainly developed in hypervisor level. Since malware does not have access to the hypervisor level, it cannot manipulate or evade the detection process. Therefore, the proposed methodology can be considered as trusted also.

4 Methodology

This section explains the approach that has been used in this study, including infrastructure, data collection and evaluation procedure. Workflow of this master thesis can be categorized into four main phases: Acquisition of volatile memory, data collection, feature extraction and data analysis. Machine learning algorithm has been trained to determine legitimate and fileless threats by analyzing volatile memory features.

4.1 Acquisition of volatile memory

Microsoft released Hyper-V in 2016 to introduce virtualization to Microsoft end users. Hyper-V is virtualization software that provides the ability to deploy and manage virtual resources such as virtual client machines, virtual servers. Additionally, end-user can customize resources and choose the operating system while deploying virtual machines. All the virtual resources can be managed via Hyper-V manager which automatically installed upon the enabling of Hyper-V. In this study, Microsoft Hyper-V environment and VMware used to demonstrate private cloud computing platform. All the virtual machines are also deployed in the Hyper-V environment. Below figure shows the virtualization infrastructure that has been used in this study.

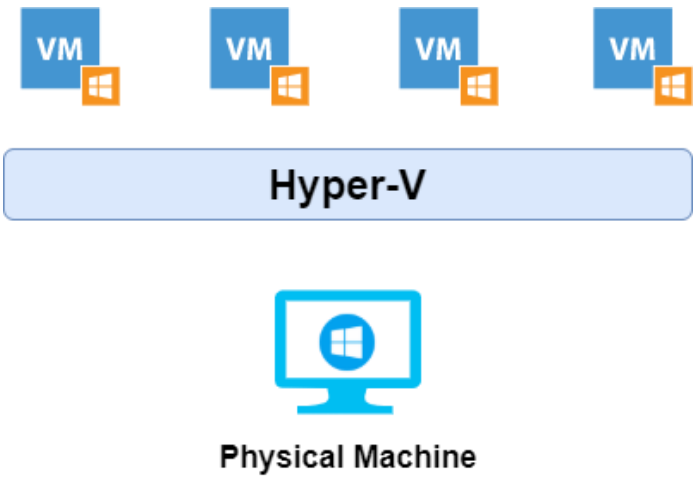

Figure 1: Hyper-V virtualization infrastructure

Hyper-V allows to take snapshot from virtual machines at the specific state and save to snapshot file. The data of the snapshot contains the virtual machine's power state (power on, power off or suspended), hard drive data, volatile memory information and device information. Necessary volatile memory data can be extracted from snapshot files using the Volatility framework. All the snapshots have been stored in to separate external hard drive for analysis. Below diagram illustrates the three main steps followed to collect volatile memory dumps. The process initiates from a fresh virtual machine which considers as baseline state.

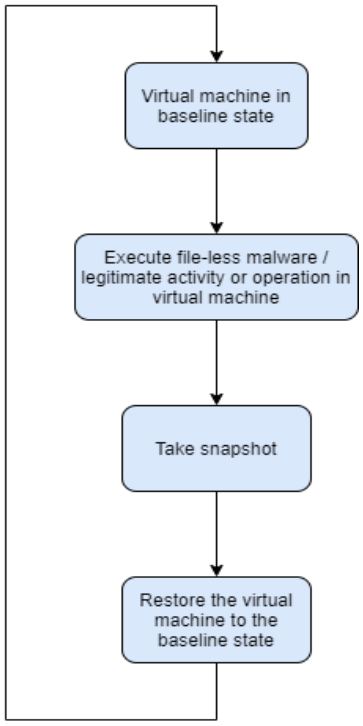

Figure 2: Process flow of collecting snapshots

Since the infrastructure has been deployed in the Hyper-V environment, virtualization technology ensures the segregation between the hypervisor and virtual machines. This methodology considered as trusted as a program running in the virtual machines cannot reach hypervisor level. Therefore, malicious programs running inside virtual machines cannot manipulate or interfere with the snapshot acquisition process.

4.2 Data collection

As described in section 4.1, new virtual machines have been deployed on the Hyper-V infrastructure.

Specifications of the virtual machines as below,

Operating Systems	Windows 7, Windows 10
Memory	2 GB
Number of processors	4
Hard Disk	20 GB
Network Connection	NAT

Table 2: Virtual machine specifications

As illustrated in Figure 2, snapshot data has been collected by executing both legitimate applications and fileless malware one at a time. During the default state of the virtual machine, snapshot has taken. Upon the execution of each program (both legitimate and malicious) and snapshot process, VM roll back to the baseline state, which considered as the default state of the client machine. For each snapshot, memory dump has been saved to an external hard drive for analysis. Additionally, the snapshotting process has taken approximately four minutes to make each 2 GB size snapshot. In this study, altogether, 40 snapshots have been analyzed, and 24 of them are legitimate while 16 are malicious fileless threats. These threat samples contain fileless threat features and categorized as fileless threats [56].

Below figure illustrates the snapshot collection process upon the execution of each program at a time. Benign application represented by green while malicious fileless malware represented by red colour. All benign programmes are related to the organizational Windows client programs and software that use a regular basis such as Microsoft office, run full antivirus, Wireshark, Procmon etc.

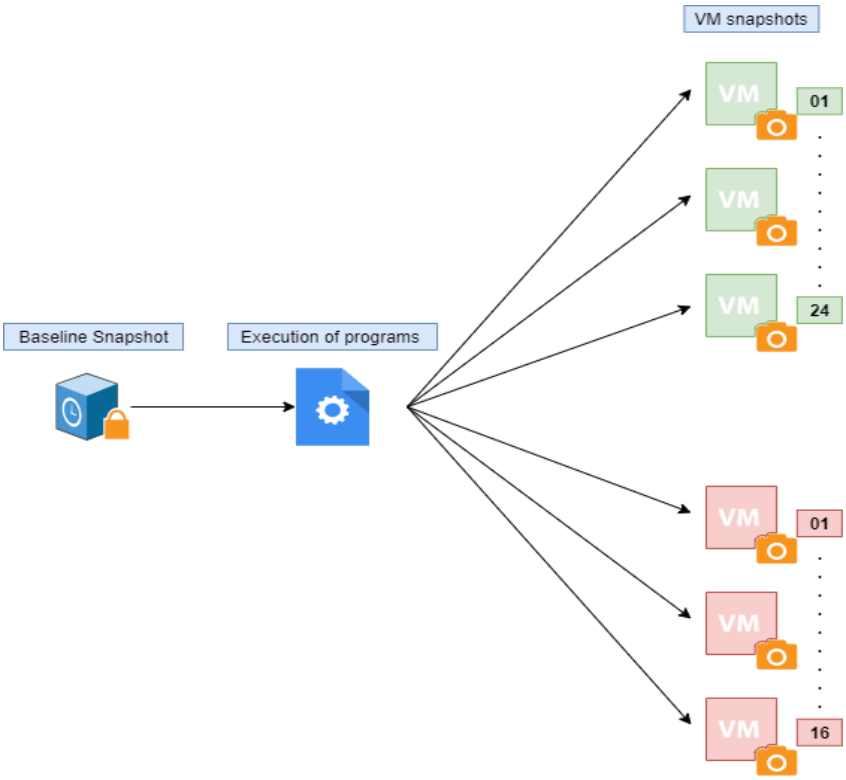

Figure 3: Snapshot creating process

Below table contains the programs executed in virtual machines before taking the snapshot. Details about fileless malware and malware samples are collected from "chenerlich" GitHub page [56], AnyRun [57] and Hybrid-Analysis.

Program	**Description**	**OS**	**Type**
Baseline	Baseline	Windows 10	Benign
Legitimate word document	Microsoft Office tool	Windows 10	Benign
Wireshark	Network monitoring tool	Windows 10	Benign
Procmon	Process monitoring tool	Windows 10	Benign
Avast antivirus	Antivirus engine	Windows 10	Benign
MS word doc with macro	Microsoft Office tool with legitimate macro	Windows 10	Benign
Spotify	Music application	Windows 10	Benign
7Zip	File archiver	Windows 10	Benign
Zoom	Video conferencing tool	Windows 10	Benign
Google chrome	Internet browser	Windows 10	Benign
WhatsApp web	Messaging and calling application	Windows 10	Benign
Outlook	Mail client	Windows 10	Benign
Adobe Reader	PDF file	Windows 10	Benign
Microsoft store	Microsoft store	Windows 10	Benign
Firefox	Internet browser	Windows 10	Benign
Skype	Messaging and calling application	Windows 10	Benign
Microsoft Excel	Microsoft Office tool	Windows 10	Benign
VMware	Virtualization software	Windows 10	Benign
iTunes	Apple devices management panel	Windows 10	Benign
Microsoft Edge	Internet Browser	Windows 10	Benign
KeePass	Password manager	Windows 10	Benign
Windows Defender scan	Microsoft firewall	Windows 10	Benign
Notepad++	Text and source code editor	Windows 10	Benign
PowerShell	Execute PowerShell script	Windows 10	Benign
Emotet	Emotet is a banking trojan malware	Windows 10	Malware

GZipDe	GZipDe malware drops backdoor	Windows 10	Malware
Macros	Malicious automation script	Windows 7	Malware
Valyria	Malicious visual basic script	Windows 7	Malware
LokiBot	Macro malware steals sensitive information	Windows 7	Malware
August	Steals credentials and sensitive documents	Windows 10	Malware
JS_POWMET	Trojan JS_POWMET is downloaded via an auto-start registry entry	Windows 10	Malware
Keybase	Macro based malware	Windows 7	Malware
Kovter	Pervasive click-fraud trojan	Windows 10	Malware
Rozena	Malicious script	Windows 10	Malware
Phase Bot	Fileless rootkit	Windows 7	Malware
Silence	Malicious script	Windows 7	Malware
CryptoWorm	Fileless Crypto-mining malware	Windows 7	Malware
CodeFork	Fileless malware by CodeFork hacker group	Windows 10	Malware
PowerWare	A novel approach to ransomware	Windows 10	Malware
Poweliks	Malware resides in the Windows registry	Windows 7	Malware

Table 3: Programs executed in virtual machines

All the snapshots have been collected to an external hard drive for later analysis. Since this study is focused explicitly on volatile memory, only volatile memory dump has been extracted from each snapshot. The size of each dump is equal to RAM size, which is 2 GB. After that, features have been obtained from each memory dump using the Volatility framework. Feature extraction method has been explained in section 4.3 Feature Extraction

4.2.1 Fileless malware samples

In this study fileless threat samples have been collected upon certain requirements such as, malware should able to execute in Windows 7 or Windows 10 environment and malware that can able to execute offline or able to communicate with command and Control servers (to retrieve the payload). Most of the available malware samples cannot be analyzed as their C&Cs are no longer active. Hence those type of threats cannot complete the entire execution in the victim machine. Additionally, some fileless threat samples that are available on internet specifically built to address certain vulnerabilities in the applications. Those type of threats can only execute upon the existence of such vulnerability in the victim machine. The main difficulty we found in this study is to collect fileless malware samples as there is only a limited number of platforms to download samples in the community. In order to download fileless malware samples, we used Hybrid analysis sandbox special educational package and full API package that provide for research purposes and Any-Run special researches. Apart from that VirusShare website [58] also used to acquire fileless scripts. In this study, multiple malware samples have been tested, and only 16 samples met the above requirements.

4.3 Feature Extraction

4.3.1 Volatility Framework

Volatility is an open-source platform used for digital forensic investigations. Raw memory dumps, virtual machine snapshots, Microsoft crash dumps, VMware dumps can be analyzed using this framework. Volatility is a well-known and frequently used platform among security experts and forensic investigators [59]. This framework supports both 32-bit and 64-bit operating systems. Besides Volatility framework support all flavours of Windows, Linux and macOS and Android. Furthermore, raw memory dumps, hibernation files, virtual box memory dumps, Microsoft crash dumps and VMware dumps can be analyzed using Volatility framework. Volatility contains over 35 plugins to analyze different characteristics of the volatile memory [60]. Prior to initiate memory dump analysis, the appropriate profile should be defined in the command. The primary objective of this process is to synchronize with frequent feature enhancement applied by the operating system. Volatility allows identifying the appropriate profile once after analyzing the metadata of the memory sample. Apart from that, this framework is able to provide active process status, hidden processes, DLL loaded, socket information, network connection information upon the analysis of the memory file. Present-day, most malware such as rootkits, fileless threats are capable of hiding their existence during the period of attack. Most of these attacks are hide their malicious processes in the volatile memory instead of writing to the hard disk. However, run time memory analysis is capable of detecting such behaviour. Hence in this research, we used the Volatility framework to inspect the memory dumps.

4.3.2 Feature extraction for dataset creation

In this research, Shell script has been developed to extract the predefined set of features from the volatile memory dumps. As explained in section 4.1, volatile memory contains data and instructions related to running programs. Developed shell script utilizes Volatility framework 2.6 to extract the necessary information from the memory dumps. In this study, different Volatility plugins have been used to obtain system information such as running processes, DLLs, Network connections [61]. Extracted features exported to the excel file. Python program has been developed for the extract only the necessary features from the excel files that are created by shell script in the previous process. For instance, avoid the duplicate DLL file in the memory dump and get the count of unique

DDL file in the memory. These set of features extracted and exported to CSV file, as shown in Appendix 1. In the CSV, columns represent the features while rows represent the memory dump. Furthermore, additional features such as network-level characteristics, file actives and registry activities are extracted from a live sandbox environment upon the execution of each malware. This allows to obtain file and registry activity features during the execution of malware.

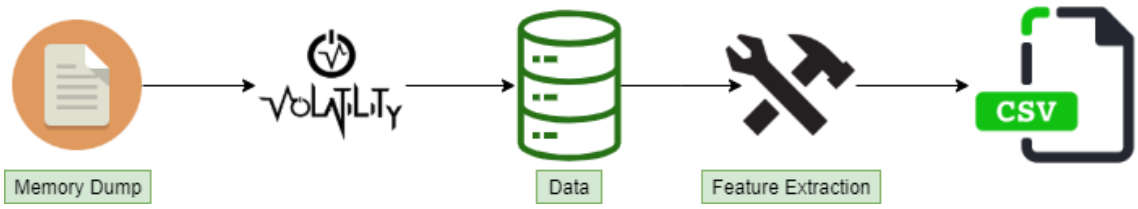

Figure 4: Feature Extraction Process

The duration for extract features from memory dump mainly depends on the size of the dump file. In this study, for 2 GM memory dump file, it has taken approximately 5 minutes to complete the feature extraction process. In this study, altogether, 40 memory dumps have been analyzed. Therefore, entirely 3 hours and 20 minutes spent for memory extraction process.

Below table contains the list of plugins used in this study to extract features from memory dumps [62]. More details regarding the Volatility plugins can be found in the Volatility Command Reference GitHub page [63].

Volatility plugin	**Description**
psxview	List of plugins utilizing in the machine
thrdscan	List of thread objects using pool scanner
svcscan	List of windows services
dlllist	List of loaded DLLs for each process
ldrmodules	List of unlinked DLLs
modules	List of modules loaded
privs	List of process privileges
callbacks	List of kernels callbacks (notification routines)
handles	List of open handles for each process
mutantscan	List of mutex objects using pool scanner
connections	List of open connections
hivelist	List of registry hives
pslist	List all running processes

Table 4: List of Volatility Plugins

4.3.3 Set of features

Table 4 explained the volatile memory features extracted from the memory dumps. List of these memory features obtained into CSV file for analysis. For each element, feature name, description, data type and Volatility plugin name has been provided in the table. In this study, list of features that are used for analysis decided based on a previous research article on ransomware detection [10], and study on an analytical approach to hunting advance threats using memory forensics [54]. Specific Volatility plugins have been used multiple times to obtain a different set of features. Additionally, list of these features contains not only the characteristics of memory but also various system operations such as file modification, registry modification during the execution of the program. These set of features are obtained by executing each program in the sandbox environment. Table 4 represents the features that are extracted using Volatility plugins.

#	Feature	Description	Type	Volatility Plugin/Source
1	handles_num	Number of handles	Integer	handles
2	hiveList	Number of registry hives	Integer	hivelist
3	dlls_ldrmodules_num	Number of DLLs used by all processes	Integer	ldrmodules
4	dlls_ldrmodules_unique_mappedpaths_num	Number of unique DLLs used by all processes	Integer	ldrmodules
5	dlls_ldrmodules_InInit_fales_num	Number of DLLs with InInit false	Integer	ldrmodules
6	dlls_ldrmodules_InLoad_false_num	Number of DLLs with InLoad false	Integer	ldrmodules
7	dlls_ldrmodules_InMem_False_num	Number of DLLs with InMem false	Integer	ldrmodules
8	dlls_ldrmodules_all_false_num	Number of DLLs with all false	Integer	ldrmodules
9	modules_num	Number of modules	Integer	modules
10	callbacks_num	Number of kernels callbacks	Integer	callbacks
11	processes_privs_enabled_not_default_num	Number of processes with enable and without default	Integer	privs
12	processes_psxview_exited_num	Number of processes completed before taking the snapshot	Integer	psxview
13	processes_psxview_false_columns_num	Number of process listing techniques that do not detect at least one process	Integer	psxview
14	processes_psxview_false_rows_num	Number of processes that are not detected by at least one process listing techniques	Integer	psxview
15	processes_psxview_num	Number of processes detected by psxview	Integer	psxview
16	processes_psxview_pslist_true_num	Number of processes detected by pslist	Integer	psxview
17	processes_psxview_psscan_true_num	Number of processes detected by psscan	Integer	psxview
18	services_svcscan_num	Number of services	Integer	svcscan
19	services_svcscan_running_num	Number of running services	Integer	svcscan
20	services_svcscan_stopped_num	Number of stopped services	Integer	svcscan
21	dlls_dlllist_unique_paths_num	Number of dlls	Integer	dlllist

22	mutex_mutantscan_num	Number of mutexes	Integer	mutantscan
23	threads_thrdscan_num	Number of threads	Integer	thrdscan
24	pslist	Number of processes	Integer	hivelist
25	tcp/udp_connections	Number of TCP/UDP connections	Integer	sockets

Table 5: List of features from Volatility

Table 6 explains the features that are extracted from the sandbox environment by executing the malware. These set of features have been extracted for each sample by executing each malware in the sandbox environment at a time. Important network features such as domain name services (DNS) requests cannot be extracted using Volatility. Therefore, we used sandbox to extract these additional important features for analysis.

26	total_reg_events	Number of registry events	Integer	sandbox
27	read_events	Number of read operations	Integer	sandbox
28	write_events	Number of write operations	Integer	sandbox
29	del_events	Number of delete operations	Integer	sandbox
30	executable_files	Number of executable files	Integer	sandbox
31	unknown_types	Number of unknown files	Integer	sandbox
32	http(s)_requests	Number of HTTP requests	Integer	sandbox
33	dns_requests	Number of DNS requests	Integer	sandbox

Table 6: List of features from Sandbox report

4.4 Data processing using machine learning algorithm

4.4.1 Classification machine learning algorithms

In this study, we applied four commonly used machine learning classification algorithms on the dataset. In machine learning, classification is a process of predicting the class of the specific given data point [64]. For instance, in this study, fileless malware detection can be identified as a classification problem. This can be categorized as binary classification since it consists of only two classes as malicious and non-malicious. In this case, known fileless threats and legitimate applications have been used as training data to train the classifier. After that, it can be used to detect the unknown fileless threats. This is known as supervised learning as output based on the input dataset. In this study, the following classification algorithms have been used: Random Forest (RF), Decision Tree (DT), K-Nearest Neighbour (K-NN) and Support Vector Machines (SVM). More information about these algorithms explained in section 2.4.1.

5 Evaluation

In this section, we evaluate the accuracy of detecting fileless threats by using the proposed methodology.

In this research, decision tree, random forest, k-nearest neighbour and support vector classifier algorithms have been chosen since our test data set has binary classifications. Additionally, based on previous studies, those algorithms widely used due to their accuracy and efficiency measures. Most suitable classification algorithm mainly depends on the nature of the dataset [64]. However, in this study cross-validation has been used to select the most appropriate classification algorithm for the available data set.

5.1 K-Fold cross-validation

K-Fold cross-validation allows to compare the different machine learning classification algorithms to pick up the most suitable method. In this research, we used *scikit-learn* python machine learning library to calculate the k-fold cross-validation for each selected classification algorithms. Below table illustrates the behaviour of each classification algorithm when K=10, training sample size: 70% and testing sample size: 30%.

Classification Algorithm	**Decision Tree**	**Random Forest**	**k-Nearest N**	**SVM**
Cross validation score	0.933	0.966	0.625	0.566

Table 7: K-Fold cross-validation performance

Based on the cross-validation score, random forest classification model scored 96%, which is highest from other evaluated models. Despite the decision tree is comparably faster than the random forest, deep decision trees experience overfitting, which decreases the performance and accuracy [65]. Overfitting refers to when the model gets trained with too much of data including noisy data, the model learned from those inaccurate data entries. In this case, the model does not classify data accurately because of inaccurate learning data. This caused overfitting in machine learning [45]. One of the advantages of random forest is, it can handle missing values. Besides, random forest classification has high efficiency because of the number of decision trees contribute to the prediction. Therefore, in this research, we used random forest as the machine learning classification algorithm.

5.2 Random Forest

Random forest is one of the most popular machine learning algorithms that belong to supervised learning algorithm. As explained in section 2.4.1**,** Random forest contains large number of individual decision trees. Each individual tree predicts the class. Instead of relying prediction on a single decision tree, random forest checks each decision tree prediction and obtain the final prediction based on the majority votes of prediction [66]. Besides random forest use bootstrap aggregating method, also known as bagging method to tree learners. Bagging is applying base classifier on random subsets of the original dataset and aggregate individual predictions of each of them [67]. Since bootstrap aggregation decreases the variance of the model, it increases the performance of the base model. Furthermore, bagging reduces the overfitting, which considered a limitation of the decision tree. Also, in the random forest method, a larger number of decision trees capable of giving more accurate result.

5.3 Fileless threat detection: performance

Below table illustrates the confusion matrix values retrieved using random forest machine learning model. Python *scikit-learn* library used for the calculation. 30% of the data set has been used as testing data for this evaluation.

		Predicted class	
		Malware	**Legitimate**
Actual class	**Malware**	4	0
	Legitimate	1	7

Table 8: Confusion matrix derived from the model

Altogether twelve samples (four malware samples and eight legitimate applications) have been involved with the confusion matrix evaluation. As per the table result, there is one false-positive result which is one legitimate application detected as malware by machine learning mode. As per the test samples, execution of Avast antivirus engine has been labelled as malware by machine learning mode.

5.4 Correlation matrix

In this study, correlation strength has been calculated against each feature and illustrated using correlation matrix heatmap. Matplotlib and Python plotting utility has been used to generate the below graph.

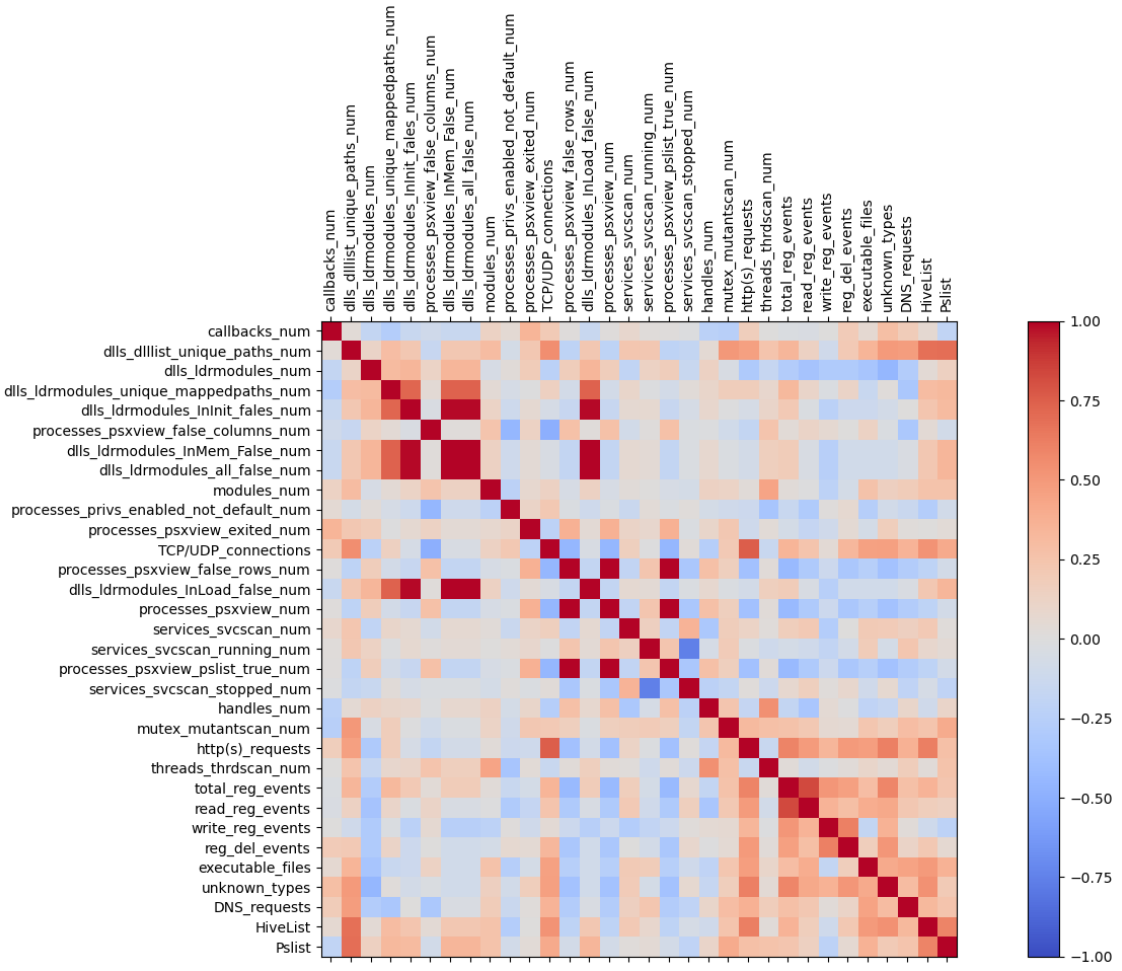

Figure 5: Correlation between features of the dataset

In Figure 5, blue denotes negative and red indicates the positive correlation coefficient values. Grey indicates that there is no correlation between features. Also, stronger in colour correspond to the magnitude of the correlation value. Diagonal correlation coefficient is always 1.0 as each feature perfectly correlates with itself. As per the figure 05 illustration, *dlls_ldrmodules_InLoad_false_num* feature highly correlated with *dlls_ldrmodules_InInit_false_num* and *dlls_ldrmodules_InMem_false_num* features. This indicates that number of DLLs that were not found in the InLoad highly correlated with number of DLLs that were not found in the InInit and InMem. In other words, count

of DLLs, including hidden DLLs that failed to load is correlated with DLL count that are not initialized. As per the figure, dark blue square indicates that service svcscan stopped count is negatively correlate with running service svcscan. In other words, there is a negative correlation between services running in the machine and services stopped. Furthermore, the TCP/UDP connection count strongly correlate with HTTP(S) request count. However, these network features not avoided from the dataset as these two features operate in two different layers in OSI seven layers (TCP/UDP belongs to transport-layer, and HTTP is an application-layer protocol). Therefore, these two features have different visibility.

5.5 Analysis of feature importance

In this study feature importance method has been used to extract the top ten features in the dataset. Below figure illustrates the ten best features that contribute to increase the accuracy of the machine learning model we use in this study. Python feature_importance_ library has been used to evaluate the relative importance of each element. Relative importance is the feature importance divided by the highest feature importance value so that values remain between 0 and 1. Feature importance calculation is a method of assign scores to input features in the dataset based on their contribution to the model accuracy [68].

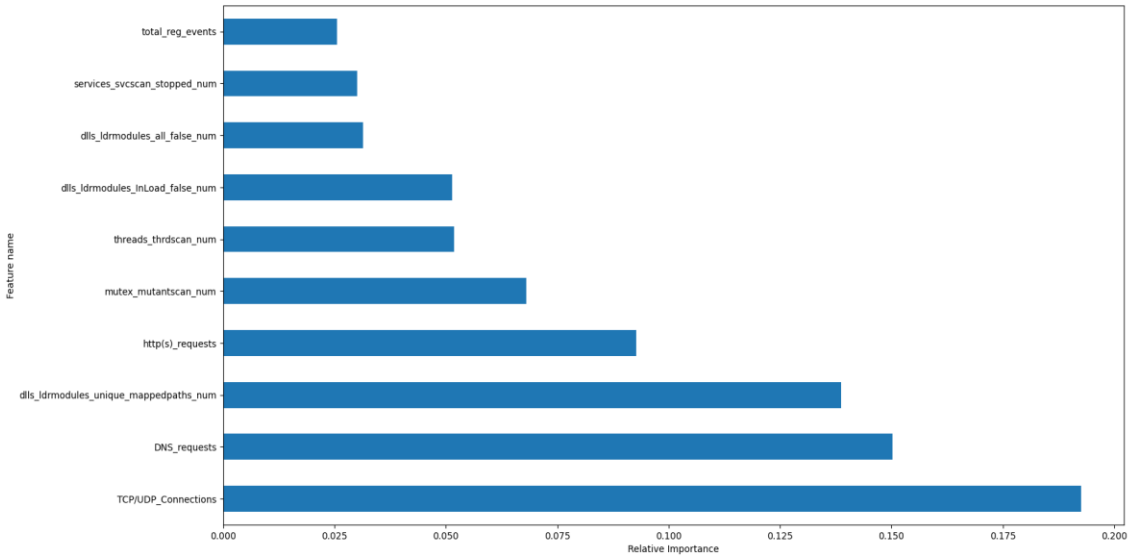

Figure 6: Top ten features according to feature importance

These top ten features are contributing to increase the model accuracy in different levels.

As per the above figure, the TCP/UDP connection count significantly contributes to the accuracy of the model. Network connection analysis is one of the essential tasks during the advance threat hunting. According to *Lockheed Martin, the Cyber Kill Chain,* most adversaries need network connectivity to achieve their objective [69]. For instance, the malicious script invokes PowerShell and connect with the C&C server to download secondary payload or exfiltration of data after the exploitation. For achieve, these task malware needs internet connectivity. Therefore, analysis of TCP/UDP connections are significantly contributing to thereat detection process. DNS requests also one of the most important features to analyze during the malware investigations. The most recent example for DNS fileless attack is *DNSMessenger* that use DNS requests to receive malicious PowerShell commands. DNSMessenger malware distributed via Microsoft word document embedded with a malicious macro. This macro leaves a backdoor which enables the communication between C&C and victim machine. Thereafter, DNSMessenger sends DNS TXT records which contains malicious PowerShell commands for further exploitation [70]. Besides, it is common that malicious domain takes down upon the user report. Therefore, cybercriminals programme malware to try out multiple abused or malicious domains. In this study, we collected DNS requests data as it is more relevant to the modern malware behaviour and as per the feature importance, it is the second most feature contributes to the model accuracy.

Apart from that, the third most important feature is DLL enumeration. DLL stands for Dynamic Link Library. As name implies "A DLL is a library that contains code and data that can be used by more than one program at the same time" [71]. This reduces memory consumption as a set of codes share among multiple programs instead of duplicating the same code. Adversaries misuse this technique by loading DLLs into a legitimate program, leading to normal process conduct malicious task and evade the detection [72]. Also, there are hidden DLLs in Windows that not show up by default. Those DLLs mostly interact with malicious activities to carry out the attack anonymously. Therefore, DLL analysis is one of the critical tasks in forensic investigations and threat hunting. In this study, we used Volatility dlllist and ldrmodules plugins to get all the DLLs in the system, including hidden DLLs. As per the figure, mutant scan also one of the essential functionalities in the malware hunting process.

In Windows, mutant is a kernel object that allows programs to synchronize between different events [73]. Adversaries often misuse this feature to avoid re-infect the same machine over time as to decrease chances of detection. For example, malicious macro embedded in the word document infect the victim machine. Each time user opens the word document, it re-infects the computer over again. This makes unnecessary potential to detect the attack. To overcome this situation, adversaries use mutants to avoid reinfection once it successfully infected the victim machine [73]. In this study, we used Volatility mutantscan to extract the created mutants in the memory.

Below table illustrates the accuracy of the model based on the selected important features. Training size is 70% of the dataset; testing size is 30% and K=10.

Number of features	Recall	Precession	F-Score	Accuracy
Best Feature	0.80	0.66	0.72	0.83
5 best features	0.80	1.00	0.88	0.86
10 best features	1.00	0.83	0.90	0.91
All features	0.80	1.00	0.88	0.96

Table 9: Performance-based on selected features

As per Table 10:

- Best feature, which is TCP/UDP connection count denotes the highest contributing feature to increase the model accuracy. By analyzing this individual feature, machine learning model is capable of distinguishing malware and legitimate applications with over 83% of overall efficiency.

- Random forest machine learning model can be trained to with best five features to achieve over 86% of accuracy in terms of fileless threat hunting

- 91% of efficiency achieved only using top ten important features

- Using all features, the machine learning model can be trained to detect fileless threats with 96% accuracy.

5.6 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised machine learning method that is used for dimensionality reduction. Figure 7 is the visualization of data distribution in this

study. PCA has been used to reduce 33 dimensions into two dimensions in order to represent in a 2D graph. Python's *Scikit-Learn* library used to draw the graph.

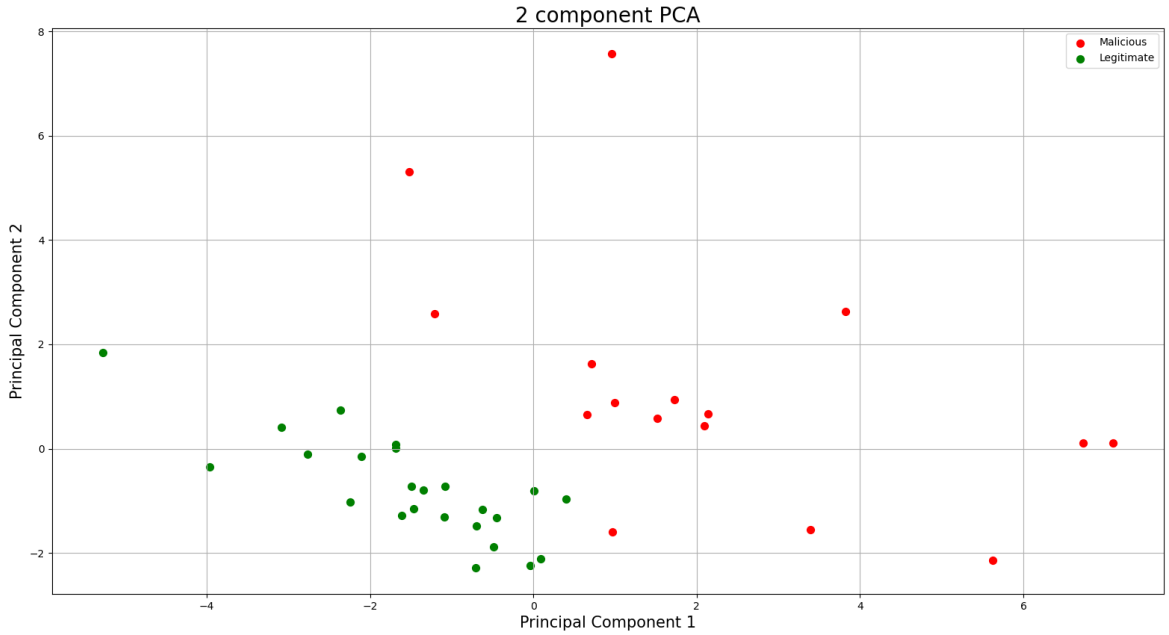

Figure 7: 2D principal component graph

As per the figure, principal component 1 (PC1) and principal component 2 (PC2) are two dimensions derived from the dataset using PCA. These two dimensions are selected based on the variance. Variance refers to the statistical measurement, that measures “how far each number in the set is from the mean and therefore, from every other number in the set” [74]. Total variance is the sum of the variance of all individual variables. So, the component which has higher value covers a higher part of the original dataset. In figure 7, the highest variance is the PC1 while the second-highest variance considers as PC2. Also, these principal components do not have any correlation between each other.

Apart from the data visualization, PCA can be used to perform analysis and rebuild the model after reducing features then train model and finally make predications and evaluate. Furthermore, dimensional reduction in PCA reduces computational power to run the model as it has only a few variables to process.

5.6.1 Performance evaluation using PCA

In this study, we applied PCA to reduce features in the dataset while reducing computational power to run the machine learning model. Since proposed methodology of malware detection should be run real-time in the hypervisor, minimal computational power consumption is much more important to provide better end-user experience. In this section, random forest classification has been used as classification model and evaluated based on predictions. Below table illustrates the performance of the model along with principal components. Training size is 70% of the dataset; testing size is 30% and K=10.

Principal Components	Cumulative Variance	Recall	Precession	F-Score	Accuracy
PC1	0.22	0.80	1.00	0.88	0.91
PC1 and PC2	0.35	0.80	1.00	0.88	0.91
PC1, PC2, PC3, PC4 and PC 5	0.64	1.00	0.41	0.58	0.93

Table 10: Accuracy based on PCA model

As table 10 illustrates, the accuracy achieved with only one principal component is higher than accuracy achieved three principal components. This emphasizes that the accuracy of the classifier doesn't necessarily improve with increased number of principal components [75]. As per the table 10, cumulative explained variance of fist five principal components is around 64% which is more than half of the overall variance. However, in some cases, all features are equally contributing to the overall variance where none of the components can be ignored theoretically. But in this study, as graph represents principal components doesn’t equally contribute to the variance. Therefore, cumulative explained variance ratio against with number of components graph has been used to ignore the principal components which make lower contribution to variance.

Below graph illustrates the cumulative explained variance ratio against with number of components. This can be used to determine number of principal components need to feed into the model.

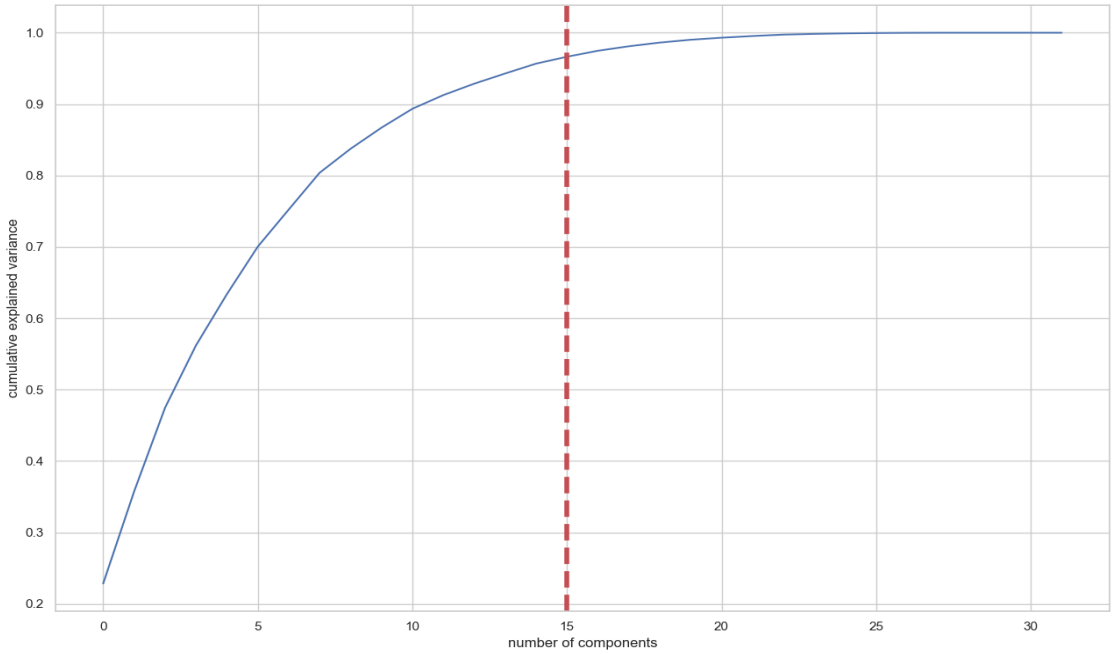

Figure 8: Cumulative explained variance with the number of components

This curve illustrates how much of the total, 32-dimensional variance is accommodated within the first *N* number of components [76]. As per the figure, first 15 components contain more than 95% of the overall variance, while other17 components explain around 5% of the variance. Also, this graph shows that after 15^{th} components, there is no huge gain of explained variance. In this study, we used PCA to reduce the number of components from 32 to 15.

Below table illustrates the random forest model performance evaluation with fist 15 principal components after dimension reduction. Training size is 70% of the dataset; testing size is 30% and K=10.

Principal Components	**Cumulative variance**	**Recall**	**Precession**	**F-Score**	**Accuracy**
15 principal components	0.97	0.80	1.00	0.88	0.96

Table 11: Accuracy based on the selected number of principal components

As per the table, 96 % accuracy achieved using 15 principal components in random forest classification model.

6 Conclusion

Fileless attacks leave no traces as these threats programmed to use legitimate Windows tools to carry out the attack. Additionally, fileless threats capable of manipulating Windows legitimate process and alter system registry files to hide their presence. In this study, we presented a methodology to detect fileless threats in virtual machines deployed on the cloud. Proposed methodology mainly based on feature analysis by using machine learning in order to detect fileless threats. The initial data set was built using analysis of volatile memory features and network features that derived from memory dumps and sandbox threat reports. In this experiment, the first task was training the machine learning classification model with the appropriate dataset. Thereafter we evaluated the detection accuracy across different machine learning algorithms such as random forest, decision tree, k-nearest neighbour, SVM. As per the cross-validation score, random forest achieved the highest accuracy, which is around 96 % upon analysis of the entire dataset. Additionally, random forest results for other sub-experiments performed: precession=1.0, recall=0.80 and F-score=0.88.

Feature selection methods used to reduce the number of features by merging correlated features. Besides, the feature importance method has been integrated to choose the best features that contribute to the accuracy of the classification model. Correlation coefficient has been calculated against all 33 features to evaluate the relationship between each feature. Based on the correlation matrix heatmap, all the DLLs, including hidden DLLs that were not found loaded DLL list highly correlated with DLLs that were not initialized. This indicates that the number of DLLs that were not found in the InLoad highly correlated with the number of DLLs that were not found in the InInit. Based on these characteristics, 'DLLs not initialized count' removed from the analysis as to increase the performance by reducing the data amount to be processed.

As per the study of feature importance, network features such as TCP/UDP connection count, DNS request count significantly contribute to distinguish between malware and legitimate applications. This finding points out, similar to typical malware, fileless malware also make unusual patterns of network traffic during the attack. Besides, according to the feature importance bar chart, ldrmodules feature which lists down the DLLs including hidden DLLs significantly contribute to the accuracy of the model.

However, DLL enumeration using dlllist plugin doesn't show such contribution to the overall efficiency emphasize that fileless threat we analyzed in this research, are programmed to use hidden DLL strategy to conceal their existence during the attack. Based on the overall result with all features, it was able to achieve random forest accuracy more than 96 %, and with the best ten features, it is around 91 %. Based on these results, it shows that even though there are a smaller number of features, it is still able to perform a significant level of accuracy with the most essential and appropriate features.

Apart from these methods, we used a statistical model to speed up the training and predictions in the machine learning model. Since threat detection should operate in real-time, minimal computational power consumption is much more critical for the proposed methodology to provide high accuracy and better end-user experience. Therefore, in this study, we used principal component analysis (PCA) to perform dimensionality reduction and speedup the random forest model. Also, here we used a statistical approach to determine the number of components involved in our classification model. After PCA dimensionality reduction, 96 % random forest accuracy achieved using only 15 principal components.

This research shows that despite the strategy of the typical malware and fileless threats have significant differences in the operational process, in network-level behaviour and volatile memory features indicate certain similarities. Combination of these features can be used to train proper machine learning classification model and use it to distinguish between malicious operations and legitimate operations in machines. Finally, this set of feature analysis can be used to optimize the performance of detecting widely spread fileless threats.

References

[1] Flexera, “flexera.com,” Flexera, 16 11 2019. [Online]. Available: https://www.flexera.com/blog/cloud/2019/02/cloud-computing-trends-2019-state-of-the-cloud-survey/. [Accessed 28 03 2020].

[2] Microsoft Corporation, “Microsoft Security Intelligence Report,” June 2016. [Online]. Available: https://download.microsoft.com/download/E/B/0/EB0F50CC-989C-4B66-B7F6-68CD3DC90DE3/Microsoft_Security_Intelligence_Report_Volume_21_Protecting_Cloud_Infrastructure_English.pdf. [Accessed March 2020].

[3] E. Zhang, “What is Fileless Malware (or a Non-Malware Attack)?,” 12 September 2018. [Online]. [Accessed 13 January 2020].

[4] vmware Carbon Black, “Let’s Define Fileless Malware,” vmware Carbon Black, [Online]. Available: https://www.carbonblack.com/resources/definitions/what-is-fileless-malware/. [Accessed 12 February 2020].

[5] PROOFPOINT STAFF, “August in November: New Information Stealer Hits the Scene,” 07 December 2016. [Online]. Available: https://www.proofpoint.com/us/threat-insight/post/august-in-december-new-information-stealer-hits-the-scene. [Accessed 02 January 2020].

[6] N. K. E. G. Carey S. Nachenberg, “Reducing malware signature set size through server-side processing”. US Patent US 8.239,944 B1 , 07 Aug 2012.

[7] Security Response, “What is Living off the Land?,” 03 Oct 2018. [Online]. Available: https://medium.com/threat-intel/what-is-living-off-the-land-ca0c2e932931. [Accessed 28 March 2020].

[8] ThreatVector, “Threat Spotlight: The Truth About Fileless Malware,” 04 April 2017. [Online]. Available: https://threatvector.cylance.com/en_us/home/threat-spotlight-the-truth-about-fileless-malware.html. [Accessed 23 March 2020].

[9] The Cylance Threat Research Team, “Threat Spotlight: Kovter Malware Fileless Persistence Mechanism,” ThreatVector, 23 January 2018. [Online]. Available: https://threatvector.cylance.com/en_us/home/threat-spotlight-kovter-malware-fileless-persistence-mechanism.html. [Accessed 29 March 2020].

[10] Y.-L. Wan, J.-C. Chang, R.-J. Chen and S.-J. Wang, “Feature-Selection-Based Ransomware Detection with Machine Learning of Data Analysis,” in *IEEE*, Nagoya, Japan, 2018.

[11] AviadCohen, Nir Nissim, “Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory,” in *ELSEVIER*, Israel, 2018.

[12] F. O'CONNOR, “WHAT YOU NEED TO KNOW ABOUT POWERSHELL ATTACKS,” 05 December 2017. [Online]. Available: https://www.cybereason.com/blog/fileless-malware-powershell. [Accessed 28 March 2020].

[13] Microsoft, “Fileless threats,” 09 April 2019. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/fileless-threats. [Accessed 04 April 2020].

[14] MITRE ATT&CK, “Windows Management Instrumentation,” MITRE ATT&CK, 17 July 2019. [Online]. Available: https://attack.mitre.org/techniques/T1047/. [Accessed 03 May 2020].

[15] MITRE , “MITRE ATT&CK,” MITRE, 16 July 2019. [Online]. Available: https://attack.mitre.org/techniques/T1067/. [Accessed 03 May 2020].

[16] McAfee, “What Is Fileless Malware?,” McAfee, [Online]. Available: https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-fileless-malware.html. [Accessed 04 April 2020].

[17] W. Williamson, “Lateral Movement: When Cyber Attacks Go Sideways,” SecurityWeek, 2016.

[18] A. ZAHARIA, “Understanding Fileless Malware Infections – The Full Guide,” 03 Feb 2016. [Online]. Available: https://heimdalsecurity.com/blog/fileless-malware-infections-guide/. [Accessed 03 May 2020].

[19] MITRE, “Net,” MITRE ATT&CK, 31 May 2017. [Online]. Available: https://attack.mitre.org/software/S0039/. [Accessed 14 May 2020].

[20] MITRE, “Regsvr32,” MITRE, 31 May 2017. [Online]. Available: https://attack.mitre.org/techniques/T1117/. [Accessed 10 May 2020].

[21] D.-Y. KAO, S.-C. HSIAO and R. TSO, “Analyzing WannaCry Ransomware Considering the Weapons and Exploits,” in *IEEE*, PyeongChang Kwangwoon_Do, 2019.

[22] McFee, “What Is Fileless Malware?,” McFee, [Online]. Available: https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-fileless-malware.html. [Accessed 10 04 2020].

[23] RedHat, “What is virtualization?,” [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-virtualization. [Accessed 23 04 2020].

[24] Microsoft Azure, “What is virtualization?,” Microsoft, [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-virtualization/. [Accessed 23 April 2020].

[25] R. Andreas, “Virtualization vs. Cloud Computing: What's the Difference?,” *Grow Your Business ,* 04 Feb 2019.

[26] J. FRANKENFIELD, “Cloud Computing,” Investopedia, 18 May 2019. [Online]. Available: https://www.investopedia.com/terms/c/cloud-computing.asp. [Accessed 23 April 2020].

[27] NirNissim, “Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining,” vol. 153, pp. 147-175, 2018.

[28] K. Amari, Techniques and Tools for Recovering and Analyzing Data from Volatile Memory, SANS Information Security Reading Room, 2009.

[29] S. M. J. F. EricConrad, “Chapter 3 - Domain 2: Asset Security (Protecting Security of Assets),” in *CISSP Study Guide (Third Edition)*, ScienceDirect, 2016, pp. 81-101.

[30] A. H. R. panel Nihad, “Data Hiding Techniques in Windows OS,” in *Data Hiding Forensics*, ScienceDirect, 2017, pp. Pages 207-265.

[31] Microsoft, “Investigate entities on devices using live response,” Microsoft, 29 April 2020. [Online]. Available: https://docs.microsoft.com/en-

us/windows/security/threat-protection/microsoft-defender-atp/live-response. [Accessed 13 May 2020].

[32] MathWorks, “What Is Machine Learning?,” MATLAB, [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html. [Accessed 03 April 2020].

[33] ayushgangwar, “Getting started with Machine Learning,” GeekForGeek, [Online]. Available: https://www.geeksforgeeks.org/getting-started-machine-learning/. [Accessed 03 April 2020].

[34] M. Sanjeevi, “Decision Trees Algorithms,” Medium, 2017.

[35] O. Harrison, “Machine Learning Basics with the K-Nearest Neighbors Algorithm,” 10 Sept 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed 15 May 2020].

[36] G. Gahukar, “Classification Algorithms in Machine Learning,” 08 Nov 2018. [Online]. Available: https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4. [Accessed 15 April 2020].

[37] A. Sharma, “Cross Validation in Machine Learning,” *Methods of Cross Validation,* p. 2, 13 April 2020.

[38] MLK, “Confusion Matrix in Machine Learning with EXAMPLE,” MakingAISimple, 04 May 2019. [Online]. Available: https://machinelearningknowledge.ai/confusion-matrix-and-performance-metrics-machine-learning/. [Accessed 29 March 2020].

[39] B. Schnarzo, “Understanding Type I and Type II Errors,” 18 August 2018. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/understanding-type-i-and-type-ii-errors. [Accessed 27 April 2020].

[40] K. P. Shung, “towards data science,” Medium , [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. [Accessed 20 April 2020].

[41] Wikipedia, “F1 score,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/F1_score. [Accessed 30 April 2020].

[42] J. Brownlee, “An Introduction to Feature Selection,” 27 April 2020. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/. [Accessed 30 April 2020].

[43] T. Bock, “Display R,” 17 June 2019. [Online]. Available: https://www.displayr.com/what-is-a-correlation-matrix/. [Accessed 06 May 2020].

[44] S. NICKOLAS, “What Does it Mean if the Correlation Coefficient is Positive, Negative, or Zero?,” Investopedia, 07 Dec 2019. [Online]. Available: https://www.investopedia.com/ask/answers/032515/what-does-it-mean-if-correlation-coefficient-positive-negative-or-zero.asp. [Accessed 05 May 2020].

[45] W. Koehrsen, “Overfitting vs. Underfitting: A Conceptual Explanation,” 28 Jan 2018. [Online]. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-conceptual-explanation-d94ee20ca7f9. [Accessed 03 05 2020].

[46] H. Goonewardana, “PCA: Application in Machine Learning,” Medium, 28 Feb 2019. [Online]. Available: https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db. [Accessed 10 May 2020].

[47] A. M. S. Madhu S. Advani, “High-dimensional dynamics of generalization error,” in *HARVARD*, 2017.

[48] M. Galarnyk, “PCA using Python (scikit-learn),” twards data science, 05 Dec 2017. [Online]. Available: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60. [Accessed 07 May 2020].

[49] P. O. S. S. a. J. B. D. Carlin, “Detecting Cryptomining Using Dynamic Analysis,” in *IEEE*, Belfast, 2018.

[50] A. C. N. N. a. Y. E. D. Nahmias, “Trusted Malware Signature Generation in Private Clouds Using Deep Feature Transfer Learning,” in *International Joint Conference on Neural Networks (IJCNN)*, Budapest, 2019.

[51] A. M. a. O. C. G. Hăjmăşan, “Dynamic behavior evaluation for malware detection,” in *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*, Tirgu Mures, 2017.

[52] M. N. Y. A. J. W B T Handaya, “Machine learning approach for detection of fileless cryptocurrency,” in *Journal of Physics: Conference Series*, Bali, 2020.

[53] P. R. Hyrum S. Anderson, “EMBER: An Open Dataset for Training Static PE Malware,” 2018.

[54] S. N. Priya B Gadgil, “Hunting advanced volatile threats using memory forensics,” *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 4, no. 4, pp. 943-948, 2018.

[55] J. N. Y. T. D. I. K. N. K. T. Yu Tsuda∗, “A Lightweight Host-Based Intrusion Detection,” in *Asia Joint Conference on Information Security*, Kanagawa, 2018.

[56] C. Erlich, *Fileless Command Lines*, Github, 2018.

[57] anyrun, *Interactive malware hunting platform*, any.run.

[58] VirusShare, “VirusShare.com,” VirusShare, [Online]. Available: https://virusshare.com/. [Accessed 11 03 2020].

[59] S. Chakkaravarthy, “Volatility: The open source framework for memory forensics,” Opensource, 12 Oct 2016. [Online]. Available: https://opensourceforu.com/2016/10/volatility/. [Accessed 18 April 2020].

[60] S. Chakkaravarthy, “Volatility: The open source framework for memory forensics,” 16 October 2016. [Online]. Available: https://opensourceforu.com/2016/10/volatility/. [Accessed 03 May 2020].

[61] M. K. A, “Finding Advanced Malware Using Volatility,” eForensics, 29 June 2016. [Online]. Available: https://eforensicsmag.com/finding-advanced-malware-using-volatility/. [Accessed 18 April 2020].

[62] M. K. A, “Finding Advanced Malware Using Volatility,” [Online]. Available: https://eforensicsmag.com/finding-advanced-malware-using-volatility/. [Accessed 23 February 2020].

[63] Volatility, “Command Reference,” GitHub, 2019.

[64] S. Asiri, “Machine Learning Classifiers,” Towards Data Science, 2018.

[65] A. Navlani, “Understanding Random Forests Classifiers in Python,” DataCamp, [Online]. Available: https://www.datacamp.com/community/tutorials/random-forests-classifier-python#comparison. [Accessed 15 April 2020].

[66] javaTpoint, “Random Forest Algorithm,” javaTpoint, [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm. [Accessed 21 April 2020].

[67] D. Dey, “ML | Bagging classifier,” [Online]. Available: https://www.geeksforgeeks.org/ml-bagging-classifier/. [Accessed 29 April 2020].

[68] A. Chauhan, “What Is Variable Importance and How Is It Calculated?,” 15 June 2017. [Online]. Available: https://dzone.com/articles/variable-importance-and-how-it-is-calculated. [Accessed 14 May 2020].

[69] Lockheed Martin, “The Cyber Kill Chain,” [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. [Accessed 30 April 2020].

[70] S. Khandelwal, “New Fileless Malware Uses DNS Queries To Receive PowerShell Commands,” The Hacker News, 06 March 2017. [Online]. Available: https://thehackernews.com/2017/03/powershell-dns-malware.html. [Accessed 25 April 2020].

[71] Microsoft, “What is a DLL?,” 18 Dec 2019. [Online]. Available: https://support.microsoft.com/en-us/help/815065/what-is-a-dll. [Accessed 02 May 2020].

[72] PC Matic Malware Research, “Running DLL Files for Malware Analysis,” 30 November 2017. [Online]. Available: https://techtalk.pcmatic.com/2017/11/30/running-dll-files-malware-analysis/. [Accessed 02 May 2020].

[73] M. Cohen, “Hunting Malware using Mutants,” [Online]. Available: https://medium.com/velociraptor-ir/hunting-malware-using-mutants-ea08e86dfc19. [Accessed 02 May 2020].

[74] A. HAYES, “Variance,” Investopedia, 02 Sept 2019. [Online]. Available: https://www.investopedia.com/terms/v/variance.asp. [Accessed 03 May 2020].

[75] U. Malik, “Implementing PCA in Python with Scikit-Learn,” 04 Nov 2017. [Online]. Available: https://stackabuse.com/implementing-pca-in-python-with-scikit-learn/. [Accessed 12 May 2020].

[76] J. VanderPlas, “In Depth: Principal Component Analysis,” in *Python Data Science Handbook*, O'Reilly Media, 2016, p. 541.

[77] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[78] N. N. Aviad Cohen, “Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory,” *Expert System with Application,* vol. 102, no. ScienceDirect, pp. 158-178, 2018.

[79] tutorialspoint, “ML - Support Vector Machine(SVM),” [Online]. Available: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm. [Accessed 26 April 2020].

[80] A. S. P. V. Shijoa, “Integrated static and dynamic analysis for malware detection,” in *International Conference on Information and Communication Technologies (ICICT 2014)*, India, 2014.

Appendix 1 – Data set used for the analysis

These tables table illustrates the dataset used for the analysis in this research.

#	Program	Label	Feature List										
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	Baseline	0	100	568	2635	467	130	78	78	78	156	10	7
2	Legitimate word document	0	97	497	1484	504	103	68	68	68	155	9	1
3	Wireshark	0	98	502	2133	492	117	75	75	75	156	12	2
4	Procmon	0	97	523	2242	476	114	62	62	62	156	9	2
5	Avast antivirus	0	98	514	2746	496	144	94	94	94	155	9	1
6	MS word doc with macro	0	98	512	2585	519	131	87	87	87	154	10	1
7	Spotify	0	97	509	2432	506	106	79	79	79	154	9	1
8	7Zip	0	98	504	2114	483	137	81	81	81	156	9	2
9	Zoom	0	98	521	2493	502	121	98	98	98	154	10	2
10	Google chrome	0	98	516	2123	523	129	83	83	83	154	10	1
11	WhatsApp web	0	97	507	2653	502	132	78	78	78	156	9	1
12	Outlook	0	98	493	1728	521	118	73	73	73	154	9	2
13	Adobe Reader	0	97	526	2343	483	138	89	89	89	154	10	2
14	Microsoft store	0	97	573	2534	443	119	63	63	63	154	9	1
15	Firefox	0	98	512	2013	424	138	71	71	71	156	9	1
16	Skype	0	97	501	2404	502	129	63	63	63	154	10	1
17	Microsoft Excel	0	97	523	2371	478	118	76	76	76	154	10	2

18	VMware	0	98	507	2236	521	136	69	69	69	155	9	2
19	iTunes	0	98	546	2115	496	109	73	73	73	155	9	1
20	Microsoft Edge	0	97	512	2132	501	131	59	59	59	156	10	2
21	KeePass	0	97	513	2112	523	112	78	78	78	154	10	1
22	Windows Defender scan	0	98	486	1833	496	136	83	83	83	154	9	1
23	Notepad++	0	97	521	2341	467	127	66	66	66	156	10	2
24	PowerShell	0	98	532	2058	504	106	71	71	71	154	10	1
25	Emotet	1	98	609	2337	538	111	70	70	70	155	9	1
26	GZipDe	1	95	671	2561	675	152	99	99	99	155	10	2
27	Macros	1	98	532	2482	551	134	90	90	90	154	10	7
28	Valyria	1	98	526	2534	538	140	95	95	95	156	9	1
29	LokiBot	1	98	506	1776	115	113	72	72	72	156	9	2
30	August	1	99	576	1261	490	104	74	74	74	157	10	2
31	JS_POWMET	1	98	608	2514	502	127	79	79	79	157	9	1
32	Keybase	1	97	525	2936	1016	795	755	755	755	156	9	2
33	Kovter	1	98	579	2316	492	126	84	84	84	154	12	0
34	Rozena	1	98	623	2418	556	133	73	73	73	156	9	2
35	Phase Bot	1	98	556	2569	543	136	93	93	93	156	9	2
36	Silence	1	98	513	1767	498	196	89	89	89	154	10	2
37	CryptoWorm	1	97	569	1892	623	113	90	90	90	157	9	1
38	CodeFork	1	98	591	2317	517	172	69	69	69	154	9	2
39	PowerWare	1	98	616	1483	639	118	70	70	70	155	9	1
40	Poweliks	1	98	668	2463	548	126	86	86	86	155	10	2

#	Program	Label	Feature List										
			12	13	14	15	16	17	18	19	20	21	22
1	Baseline	0	5	60	60	60	0	393	139	254	15453	446	791
2	Legitimate word document	0	5	47	47	47	0	393	134	256	16060	443	965
3	Wireshark	0	5	45	45	45	0	392	139	253	13222	420	637
4	Procmon	0	5	47	47	47	0	393	134	259	14352	423	702
5	Avast antivirus	0	5	52	52	52	0	391	132	253	16122	413	769
6	MS word doc with macro	0	5	47	47	47	0	393	137	258	13900	410	722
7	Spotify	0	5	43	43	43	0	393	134	259	13432	412	672
8	7Zip	0	5	53	53	53	0	393	139	254	12257	463	712
9	Zoom	0	5	47	47	47	0	393	137	258	14463	421	748
10	Google chrome	0	5	55	55	55	0	393	140	253	13874	411	698
11	WhatsApp web	0	5	44	44	44	0	393	137	258	13921	416	643
12	Outlook	0	5	56	56	56	0	393	134	259	13280	412	709
13	Adobe Reader	0	5	52	52	52	0	393	137	258	12421	428	631
14	Microsoft store	0	5	46	46	46	0	392	139	253	12318	434	718
15	Firefox	0	5	52	52	52	0	393	134	259	13426	463	728
16	Skype	0	5	49	49	49	0	393	137	256	12798	421	673
17	Microsoft Excel	0	5	43	43	43	0	393	134	259	13248	410	702
18	VMware	0	5	59	59	59	0	393	140	253	12287	449	673
19	iTunes	0	5	46	46	46	0	393	137	256	14321	407	724
20	Microsoft Edge	0	5	51	51	51	0	393	134	259	13871	421	697
21	KeePass	0	5	58	58	58	0	392	139	253	16548	422	713
22	Windows Defender scan	0	5	49	49	49	0	393	137	258	12874	417	648
23	Notepad++	0	5	55	55	55	0	392	133	259	15489	443	721
24	PowerShell	0	5	47	47	47	0	393	134	259	13651	491	786
25	Emotet	1	5	43	43	43	0	393	136	257	12624	469	635

26	GZipDe	1	5	56	56	56	0	393	140	253	16741	552	790
27	Macros	1	5	52	52	52	0	393	135	258	13301	488	679
28	Valyria	1	5	47	47	47	0	393	136	257	15073	446	822
29	LokiBot	1	5	46	46	46	0	393	140	253	13878	445	809
30	August	1	5	40	40	40	0	393	134	259	12917	407	798
31	JS_POWMET	1	5	50	50	50	0	393	138	255	14536	427	793
32	Keybase	1	5	43	43	43	0	393	137	256	14245	429	793
33	Kovter	1	4	43	43	43	0	393	136	257	12259	419	609
34	Rozena	1	5	47	47	47	0	393	138	255	13876	443	798
35	Phase Bot	1	5	50	50	50	0	393	137	256	14423	479	806
36	Silence	1	4	43	43	43	0	393	137	256	15646	484	688
37	CryptoWorm	1	5	47	47	47	0	393	138	255	13775	492	811
38	CodeFork	1	5	48	48	48	0	393	140	253	13423	434	613
39	PowerWare	1	5	43	43	43	0	393	136	257	12624	469	720
40	Poweliks	1	5	51	51	51	0	393	136	257	12241	413	673

#	Program	Label	Feature List										
			23	24	25	26	27	28	29	30	31	32	33
1	Baseline	0	25	24	1	0	0	0	0	0	0	12	44
2	Legitimate word document	0	1881	915	911	55	0	2	0	0	0	12	47
3	Wireshark	0	1709	812	796	63	0	0	1	4	2	12	45
4	Procmon	0	396	377	19	0	0	0	0	0	0	14	48
5	Avast antivirus	0	1251	1035	579	0	0	0	1	0	0	13	52
6	MS word doc with macro	0	1015	1033	525	57	0	0	0	0	0	12	47
7	Spotify	0	1123	1025	503	41	0	0	0	0	0	12	45
8	7Zip	0	1286	1196	623	0	0	0	0	0	0	12	48

9	Zoom	0	1344	789	478	0	0	0	0	0	0	13	45
10	Google chrome	0	1886	1504	342	0	0	0	0	0	0	12	47
11	WhatsApp web	0	1621	1128	421	6	0	0	0	0	0	12	44
12	Outlook	0	1104	1016	445	3	0	0	0	0	0	13	48
13	Adobe Reader	0	1498	996	512	0	1	0	0	0	1	13	48
14	Microsoft store	0	1023	894	517	0	0	0	0	0	0	12	45
15	Firefox	0	1115	1021	529	0	0	1	0	0	0	12	44
16	Skype	0	1017	653	322	3	1	0	0	0	0	13	44
17	Microsoft Excel	0	1151	1006	536	0	0	0	0	0	0	12	44
18	VMware	0	1089	1034	486	0	0	0	0	0	0	13	45
19	iTunes	0	991	967	511	13	1	0	1	3	2	12	48
20	Microsoft Edge	0	1610	1542	394	0	0	0	0	0	0	12	45
21	KeePass	0	336	114	756	62	0	0	0	0	0	12	44
22	Windows Defender scan	0	1249	1428	313	42	1	0	0	0	0	13	44
23	Notepad++	0	1076	776	602	0	0	0	0	0	0	13	48
24	PowerShell	0	1468	1026	416	4	0	0	0	0	0	12	45
25	Emotet	1	2414	1639	703	72	2	5	13	15	5	16	51
26	GZipDe	1	1271	926	344	1	0	1	1	1	0	14	56
27	Macros	1	1955	1236	649	70	1	5	1	1	1	14	52
28	Valyria	1	1808	1149	604	55	1	2	3	6	5	15	49
29	LokiBot	1	1165	1039	121	5	3	2	0	1	25	14	53
30	August	1	2063	1310	697	56	1	11	4	8	5	14	47
31	JS_POWMET	1	310	231	79	0	1	0	0	2	2	16	50
32	Keybase	1	2006	877	124	5	0	0	0	1	1	15	54
33	Kovter	1	393	302	90	1	0	0	1	17	0	13	52
34	Rozena	1	1844	1183	604	57	0	2	0	0	1	13	50
35	Phase Bot	1	1424	923	588	13	0	0	3	7	17	14	49

36	Silence	1	2191	1317	728	41	0	3	7	8	21	13	46
37	CryptoWorm	1	1871	1631	113	7	4	2	4	12	7	14	52
38	CodeFork	1	2023	1121	623	12	1	7	5	4	6	16	46
39	PowerWare	1	2413	1643	721	73	2	5	14	12	5	16	51
40	Poweliks	1	1637	1061	453	31	0	2	3	8	2	15	52

In the above three tables, feature list denoted by numeric values from 1 to 33. Feature name corresponding to each numeric value mentioned in the below table.

Number	**Feature**	**Description**	**Type**
1	handles_num	Number of handles	Integer
2	hiveList	Number of registry hives	Integer
3	dlls_ldrmodules_num	Number of DLLs used by all processes	Integer
4	dlls_ldrmodules_unique_mappedpaths_num	Number of unique DLLs used by all processes	Integer
5	dlls_ldrmodules_InInit_fales_num	Number of DLLs with InInit false	Integer
6	dlls_ldrmodules_InLoad_false_num	Number of DLLs with InLoad false	Integer
7	dlls_ldrmodules_InMem_False_num	Number of DLLs with InMem false	Integer
8	dlls_ldrmodules_all_false_num	Number of DLLs with all false	Integer
9	modules_num	Number of modules	Integer
10	callbacks_num	Number of kernels callbacks	Integer
11	processes_privs_enabled_not_default_num	Number of processes with enable and without default	Integer
12	processes_psxview_exited_num	Number of processes completed before taking the snapshot	Integer
13	processes_psxview_false_columns_num	Number of process listing techniques that do not detect at least one process	Integer

14	processes_psxview_false_rows_num	Number of processes that are not detected by at least one process listing techniques	Integer
15	processes_psxview_num	Number of processes detected by psxview	Integer
16	processes_psxview_pslist_true_num	Number of processes detected by pslist	Integer
17	processes_psxview_psscan_true_num	Number of processes detected by psscan	Integer
18	services_svcscan_num	Number of services	Integer
19	services_svcscan_running_num	Number of running services	Integer
20	services_svcscan_stopped_num	Number of stopped services	Integer
21	dlls_dlllist_unique_paths_num	Number of dlls	Integer
22	mutex_mutantscan_num	Number of mutexes	Integer
23	threads_thrdscan_num	Number of threads	Integer
24	pslist	Number of processes	Integer
25	tcp/udp_connections	Number of TCP/UDP connections	Integer
26	total_reg_events	Number of registry events	Integer
27	read_events	Number of read operations	Integer
28	write_events	Number of write operations	Integer
29	del_events	Number of delete operations	Integer
30	executable_files	Number of executable files	Integer
31	unknown_types	Number of unknown files	Integer
32	http(s)_requests	Number of HTTP requests	Integer
33	dns_requests	Number of DNS requests	Integer