PROJECT REPORT

On

"Comprehensive Study of Anti-Detection Techniques in Windows Malware"

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Computer Applications (BCA)

In the department of

Computer Science and Engineering

Submitted by:

Sahil Ahamed (UG/02/BCA/2022/016)

Soumyajit Maji (UG/02/BCA/2022/053)

Arnaa Das Burman (UG/02/BCA/2022/008)

Under the Guidance of

Mr. Sayantan Singha Roy

(Assistant Professor)

School of Engineering & Technology

ADAMAS University, Kolkata, West Bengal

July 2024 – December 2024

CERTIFICATE

This is to certify that the project report entitled ***"*Comprehensive Study of Anti-Detection Techniques in Windows Malware*"***, submitted to the School of Engineering & Technology (SOET), **ADAMAS UNIVERSITY, KOLKATA** in partial fulfilment for the completion of **Semester – 5th** of the degree of **Bachelor of Computer Applications** in the department of **Computer Science and Engineering**, is a record of bonafide work carried out by **Sahil Ahamed, UG/02/BCA/2022/016, Soumyajit Maji, UG/02/BCA/2022/053, Arnaa Das Burman, UG/02/BCA/2022/008** under our guidance.

All help received by us from various sources have been duly acknowledged.

No part of this report has been submitted elsewhere for award of any other degree.

Mr. Sayantan Singha Roy

(Assistant Professor)

Mr. Sayantan Singha Roy

(Project Coordinator)

Dr. Sajal Saha

(HOD CSE)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our project, which is the result of a studied blend of both research and knowledge.

We express our earnest gratitude to our project guide **Mr. Sayantan Singha Roy (Assistant Professor)**, **Department of CSE**, for his constant support, encouragement and guidance. We are grateful for his cooperation and valuable suggestions.

Finally, we express our gratitude to all other members who are involved either directly or indirectly for the completion of this project.

DECLARATION

We, the undersigned, declare that the project entitled 'Comprehensive Study of Anti-Detection Techniques in Windows Malware', being submitted in partial fulfillment for the award of Bachelor of Computer Applications Degree in Computer Science and Engineering, affiliated to ADAMAS University, is the work carried out by us.

Sahil Ahamed
(UG/02/BCA/2022/016)

Soumyajit Maji
(UG/02/BCA/2022/053)

Arnaa Das Burman
(UG/02/BCA/2022/008)

ABSTRACT

In this project, our focus is on developing a new method for fileless malware detection that is specifically targeting the Windows operating system. Anti-detection techniques for malware, which have been studied, will find ways through which the present challenges posed by the security of such malware threats can be handled. The methodology employs an image-based machine learning framework to convert memory dump snapshots from virtual machines into grayscale images, and later these undergo enhancement techniques like CLAHE and Wavelet Transform for effective feature extraction.

The project methodology and dataset address future research endeavors while taking into consideration scalability, accuracy, and reproducibility. Thus, it attempts to redefine the traditional detection methodologies with the incorporation of the integrated modern memory forensics and proficient machine learning techniques in order to play an effective role in this ever-evolving field of cybersecurity.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

The project centers around the background of malware's increasing complexity and sophistication, which today is exemplified by fileless malware. This kind of malware has become the foremost challenge in the field of cybersecurity in that it does not depend on executable files. It lives directly in the memory of the system, makes illegal use of processes considered trusted, and works with legitimately available systems tools such as PowerShell, Registry, WMI etc affecting malicious activities. Therefore, classic antivirus and intrusion detection systems are not effective in identifying and mitigating threats from such kinds of malware.

Rising Threat of Fileless Malware: Fileless malware has dramatically grown in prevalence because of its stealthy nature and its ability to bypass both signature based as well as heuristic detection mechanisms. These are Attacks that exploit system vulnerabilities and "living off the land binaries" (LOLBins) for persistence and malicious payload execution.

Challenges in Detection: Traditional static and dynamic analysis approaches frequently have been unable to detect fileless malware because such malware has a minuscule footprint on disk.

Behavioral analysis and memory forensics show promise, but these need to be enhanced to cope with increasingly complex and evolving malware strategies.

Project Interest: It branches from memory forensics and image processing for the improvement of detection accuracy from fileless malware. It identifies the existing research gaps, including the imbalance of classes in the datasets and inadequate feature extraction techniques, and emphasizes the necessity for reproducible studies.

1.2 Purpose of the Project

The purpose of the project is to develop an advanced and efficient framework for detecting fileless malware on Windows operating systems. Fileless malware is a sophisticated cybersecurity threat that evades traditional detection methods by operating entirely in memory without leaving discernible traces on the file system. The project aims to:

Enhance Malware Detection Accuracy: Utilize image-based machine learning techniques to detect malicious activities more effectively by analyzing memory dumps.
Address Anti-Detection Techniques: Study and counteract the strategies employed by fileless malware to avoid detection.
Contribute to Cybersecurity Research: Create a reproducible methodology and dataset for future studies in malware analysis.
Bridge Technology Gaps: Improve upon existing memory forensics and machine learning techniques to overcome limitations such as feature extraction inefficiencies, scalability, and handling evolving threats.

By achieving these goals, the project provides a robust tool to enhance cybersecurity defenses against increasingly complex and evasive malware.

1.3 Problem Statement

The growing prevalence of entryway malware has become one of the severe threats to cybersecurity. In contrast to normal malware, which finds its way into files and commands, entryway malware resides entirely in memory and works by leveraging legitimate system processes. Because of this property, it becomes more difficult to detect by conventional signature-based and heuristic detection techniques. Existing detection frameworks face the following challenges:

Evasion of Signature-Based Techniques: Fileless malware avoids leaving detectable traces in the file system, which makes the static and signature-based methods of detection ineffectual.

Inadequate Memory Forensics Techniques: At present, tools for memory forensics fail to automate detection of complex fileless malware behavior with mostly poor accuracy and scalability.

Feature Extraction Challenges: Extracting meaningful features from dumps of volatile memory, thus ensuring an effective classification of malware, is a complicated task requiring advanced preprocessing and visualization techniques.

Emerging Threat: File load less malware continually evolves and employs very sophisticated techniques to resist detection, techniques that already supersede present counter security measures.

The project tries to solve these problems by employing image base machine learning techniques as a means of detecting malicious behaviors in memory dumps. This represents an original and efficient approach to handling the issues presented by fileless malware.

1.4 Objective

The project here therefore looks towards developing a highly effective and scalable partial machine-learning framework for fileless malware detection. To be specific therefore, the project intends:

Creating an image-based machine learning framework: Develop a methodology for detecting fileless malware by converting memory dumps into grayscale images for analysis.
Generate and preprocess memory dump images: Extract memory dumps from snapshots of virtual machines and produce high-quality images for use in machine learning.
Enhance image features: Techniques such as CLAHE (Contrast Limited Adaptive Histogram Equalization) and Wavelet Transform.
Train and validate the machine learning models: Train models, like CNN, on enhanced images for identifying malicious memory access. Validate the performance of the model using unseen datasets to ensure reliability and generalization.

Addressing anti-detection techniques: Analyze and counteract evasion strategies of fileless malware that are designed to circumvent typical detection methods.
Assisting Future Research Efforts: A dataset and methodology that can be reproduced are provided for further research in fileless malware detection based on machine learning applications in cybersecurity.

1.5 Structure of Project

The outline of this project is shown as follows -

In Chapter 2, Literature Reviews of some of the previous related studies are provided.

In Chapter 3, Technology used

In Chapter 4, the methodology of the proposed system will be provided.

In Chapter 5, the hardware and software requirements will be provided.

In Chapter 6, the implementation and results will be provided.

In Chapter 7, the conclusion and recommendation will be provided.

CHAPTER 2
LITERATURE REVIEW

[1] Afreen et al. (2020) offered an analysis of the fileless malware (as known as Advanced Volatile Threats, AVT), narrowing on the possible evasive nature and the challenges in detection. In contrast to traditional malware, fileless malware gets stored in memory whereas it exploits legitimate system tools like PowerShell (PS) and Windows Management Instrumentation (WMI), all which renders them difficult to detect by traditional signature-based antivirus-solution systems. Behavioral analysis, memory forensics, and process-monitoring are key detection methods the authors stress and propose layering these approaches in order to improve such advanced threats of detection and prevention

The paper did not provide any individual datasets used for the analysis of fileless malware but rather focused on techniques, methods, and associated challenges for detecting and mitigating fileless malware.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Arrival Methods, Fileless Malware, Living off the Land Binaries (LOLBins), Memory-Based Execution.	Behavioral Analytics Algorithms, Signature-Based Detection, Memory Forensics Tools, Process Injection Detection, Anomaly Detection, Logging and Auditing Algorithms.	Systems like endpoint protection platforms (EPP) and intrusion detection/prevention systems (IDS/IPS) use this method. Also Volatility framework, FTK Imager is used.	Effective Differentiation Between Legitimate and Malicious Use of Dual-Use Tools, Advanced Memory Forensics Techniques , Automated Detection of Fileless Malware.

Table 2.1 : Key concepts of Afreen et al.

[2] B N et al.(1) discussed file-less malware in their work. This is malware that operates in the memory space evading traditional detection by only utilizing legitimate system tools such as PowerShell. The paper portrays the entire lifecycle of this sort of malware, its evasion methodology, challenges in detection, and proposes certain mitigation strategies such as sandboxing, heuristic analysis, and behavior-based methods for a robust defense.

This is the kind of research paper that does not highlight any specific dataset for the research. It talks about the detection and possible mitigation in terms of certain tools and strategies for example PowerShell, Windows Management Instrumentation (WMI number one, and behavioral analysis. In other words, it describes certain techniques and approaches rather than reflecting on specific datasets used for experimentation or testing purposes.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**REASEARCH GAPS**
Heuristic Based Detection, Execution Emulation.	No particular algorithms have been used.	Powershell, WMI, YARA.	Evolving Threat Landscape, Comprehensive Frameworks.

Table 2.2 : Key concepts of B N et al.

[3] Catak et al. (1) have designed a new malware detection filter that uses GDA. It showed higher reliability in terms of detecting and mitigating network malware attacks compared to existing algorithms. The authors tested their filter extensively and validated its effectiveness and recommended its deployment for the protection of Internet of Things devices and additional research into other attack models to enhance its application.

The context has not stated which datasets used in the research paper. Details about datasets can be obtained when one would have to refer back to the full text of the paper or methodology section where such details are usually placed.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**REASEARCH GAPS**
Algorithm Development, Implementation and Testing, Comparative Analysis(MSE) and Regression analysis Validation, Recommendations for Deployment.	Convolutional Neural Networks (CNNs), Support Vector Machines (SVM), Random Forests, Decision Trees, K-Nearest Neighbors (KNN), Deep Learning Techniques.	TensorFlow, Keras, PyTorch, Scikit-learn, Wireshark, Snort, Jupyter Notebook, MATLAB.	Detection of Multi-Family Dynamic Malicious Behaviors, Model Sustainability, Accurate Selection of PE File Fragments, Detection of Virtual Machine Escapes, Dynamic Analysis Limitations, Generalization Across Different Systems and Environments.

Table 2.3 : Key concepts of Catak et al.

[4] Demmese et al. (2023) proposed a new approach to fileless malware traffic detection based on image visualization and CNNs. The research work focused on the conversion of Cobalt Strike beacon payloads into grayscale images with an impressive accuracy rate of 99.48% for identifying evasive malicious traces within network traffic. This study, in summary, suggests that image-based methods can be integrated with machine learning toward better detection of malware that evades the more traditionally used methods.

The datasets used in the research by Demmese et al. (2023) are as follows:

- **Benign Dataset**
- **Malicious Dataset**
- **Testing Dataset**

METHODOLOGY	ALGORITHMS	TOOLS	REASEARCH GAPS
Image Visualization, Convolutional Neural Networks (CNNs), Data Preprocessing, Resampling Techniques.	Adam Optimization Algorithm, Dropout Regularization, Max Pooling.	TensorFlow, OpenCV, NumPy, Cobalt Strike.	Class Imbalance and Scarcity of Labeled Fileless Malware, Handling Temporal Dependencies in Network Traffic, Exploration of Advanced Image Conversion Techniques, Enhancing Image Contrast for Better Detection.

Table 2.4 : Key concepts of Demmese et al.

[5] Handaya et al. (2020) proposed a machine learning-based approach towards fileless cryptocurrency mining malware detection. The work utilizes the EMBER dataset and classifies conventional malware and fileless cryptocurrency mining malware, based on Monero mining. Algorithms like k-Nearest Neighbors, SVM, and Random Forest were applied in order to improve the accuracy and efficiency of detection against the constantly evolving nature of malware obfuscation and signature evasion.

This dataset contains more than 1 million samples of SHA-256 hashes from PE files that were scanned in 2018. Of which, about 900K samples will be for training and about 200K will be for testing. The dataset will provide static features from malware and benign files as a basis for the proposal's different versions of malware detection and classification models.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**REASEARCH GAPS**
Static and Dynamic Malware Analysis, Feature Extraction from EMBER Dataset, Outlier Detection Techniques, Fileless Attack Detection.	k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Random Forest.	PowerShell, Windows Management Instrumentation (WMI), EMBER dataset.	Need for More Accurate Machine Learning Models, Limited Feature Extraction for Fileless Attacks, Inadequate Classification of Cryptocurrency Mining Malware.

Table 2.5 : Key concepts of Handaya et al.

[6] Khushali et al. (2020) recommend a review on fileless malware with a focus upon various detection and mitigation techniques. The authors discuss fileless malware that operates within the memory entirely, depending on tools such as PowerShell and Windows Management Instrumentation (WMI) to avoid detection. The paper discusses the challenges it poses to fileless malware, its lifecycle, detection methods such as behavioral analysis, memory analysis, process injection techniques, and several others.

The research paper does not mention specific datasets used for the analysis in the fileless malware detection or mitigation. Instead, it offers a review of techniques and methods used to detect and mitigate fileless malware based on various behaviors and characteristics of malware.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAPS**
Signature-Based Detection, Heuristics Based Detection.	Static Analysis, Dynamic Analysis, Hybrid Analysis, Memory Analysis.	PSDEM, YARA, Volatility Framework etc.	Powershell and WMI-specific threat detection, Insufficient focus on Memory Forensics Automation.

Table 2.6 : Key concepts of Khushali et al.

[7] More et al. (1) have proposed a simulation framework for detecting and analyzing fileless malware, which, using many tools and techniques ranging from static analysis to dynamic analysis and even log analysis, shall provide more comprehensive understanding in the way of fileless malware's behavior and give effective strategies for mitigation.

The research paper does not explicitly mention any specific datasets that can be used for fileless malware detection and analysis. However, the research focuses on the creation and execution of fileless malware scripts using the Metasploit framework, followed by analysis through various methods of detection. The framework has controlled virtual environments for testing and analysis, such as process monitoring tools, PowerShell logging, and network traffic inspection; however, no external datasets were mentioned for the study.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**REASEARCH GAPS**
Signature Based Detection, Behaviour Based Detection, Heuristic Based Detection, Entropy Analysis.	Static Analysis, Dynamic Analysis, Log Analysis.	Sandbox, Process Explorer, Metasploit Framework, CyberChef, Spunk, Windbg, Netstat, Dumpit.	Complexity of Analysis, Evolving Threat Landscape, Comprehensive Frameworks.

Table 2.7 : Key concepts of More et al.

[8] Saad et al. (2019) have proposed a comprehensive analysis of the challenges faced by machine learning-based malware detection systems in real-world scenarios. Paper limits static analysis and shows a need for dynamic behavioral analysis to address emerging threats. It identifies challenges, such as the retraining cost of models, interpretability issues, and susceptibility to adversarial attacks, and then outlines innovative solutions, including disposable micro-detectors and improved techniques for interpretability, enhancing the resilience and effectiveness of next-generation malware detection systems.

The paper mentions numerous datasets used in prior studies for evaluating malware detection techniques. Some of the mentioned datasets are:

- **Hassen et al. (2017)**
- **Naeem et al. (2018)**
- **Su et al. (2018)**
- **Kilgallon et al. (2017)**

METHODOLOGY	ALGORITHMS	TOOLS	REASEARCH GAPS
Hybrid Machine Learning Approaches, Behavioral Analysis, Static and Dynamic Analysis, Adversarial Training, Feature Evolution and Confusion Exploitation, Image Recognition, Interpretability in Machine Learning.	Decision Trees, Random Forests, Support Vector Machines (SVM), Neural Networks, k-Nearest Neighbors (k-NN), Naive Bayes, Ensemble Methods Anomaly , Detection Algorithms.	TensorFlow, Keras, Scikit-learn, Malware Analysis Tools, Apache Spark, OpenCV, Weka, Pandas and NumPy, Jupyter Notebooks.	Scalability to Big Data, Feature Selection and Evolution, Inexpensive Training Methods.

Table 2.8 : Key concepts of Saad et al.

[9] Zhang et al.(1) proposed a CNN-based neural network model for the detection of malicious code in memory PE file fragments in 2023. The research clearly indicates that the model was capable of detecting malicious samples through the analysis of fragments of various lengths and extraction locations with high accuracy rates, especially for 4096-byte fragments. Dynamic analysis of fileless malware is highlighted in the study as an essential aspect of memory forensics and malicious code detection methodologies.

The research paper used datasets created by collecting static samples from VirusShare and Malshare. The authors ran the samples in a virtual machine, dumped memory information, and extracted processes and DLL files from the memory data. The dataset they created contains both benign and malicious samples of in-memory PE files.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**REASEARCH GAPS**
Dataset Creation, Neural Network Model Construction, Input Length Calculation, Training and Evaluation, Dynamic Analysis.	Long Short-Term Memory (LSTM), Conventional CNN, XGBoost (XGB), Random Forest (RF), Support Vector Machine (SVM), Decision Tree, Deep Forest (DF).	PyTorch, Python, Anaconda, VMWare.	Detection of Multi-Family Dynamic Malicious Behaviors, Improving Model Sustainability, Accurate Selection of PE File Fragments, Detection of Virtual Machine Escapes, Dynamic Analysis Limitations, Generalization Across Different Systems and Environments.

Table 2.9 : Key concepts of Zhang et al.

[10] Khalid et al. (2023) proposed a machine learning-based fileless malware detection technique as traditional methods fail to detect them, as they run in direct memory without any dependency on files. They collected the memory dump from virtual machines and used the Volatility memory forensics tool for the key extraction of features. These features were analyzed using multiple machine learning algorithms, and Random Forest outperformed other classifiers with an accuracy of 93.33% in identifying fileless malware across datasets from VirusShare, AnyRun, and others .

Khalid et al. (2023) used five widely recognized datasets for their fileless malware detection research. These datasets include:

- **VirusShare**
- **AnyRun**
- **PolySwarm**
- **Hatching Triage**
- **JoESandbox**

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Feature Extraction with Memory Forensics, Machine Learning Classification, Cross Validation and Feature Scaling, Dataset Generation..	Random Forest, Decision Tree, Support Vector Machine, Logistic Regression, K-Nearest Neighbours, XGBoost, Gradient Boosting.	VMware Workstation, AnyRun, Jupyter Notebook, VMWare Tool - vmss2core, scikit-learn.	Feature Selection and Extraction, Integration of Behavioural Indicators.

Table 2.10 : Key concepts of Khalid et al.

[11] Bucevschi et al. (2019) designed an entry-level anomaly detection methodology for the detection of file-less attacks, identifying such activities by analyzing command line arguments using a modified version of the Perceptron algorithm. Their methodology is a feature extraction-based methodology from common system tools command lines which can differentiate benign from malicious activities with a high sensitivity and accuracy in detecting these attacks.

The datasets included in the research are an initial collection of 500,551 command lines in PowerShell scripts, WMI scripts, Windows tasks, LNK files, batch scripts, and other types of files containing command lines. These datasets were shared by Bitdefender Cyber Threat Intelligence Lab and Virus Total Intelligence and were collected from February 2019 to May 2019. After applying some filters and removing inconsistencies, the authors retained 499,550 command lines.

METHODOLOGY	ALGORITHMS	TOOLS	REASEARCH GAPS
Feature Extraction, Conditional Mutual Information Maximization, Training and Testing.	One Side Class Perceptron (OSC) Natural Language Processing (NLP) Techniques Convolutional Neural Networks (CNNs) Kernel Support Vector Machine (SVM) and Gradient Boosted Trees.	Bitdefender Cyber Threat Intelligence Lab VirusTotal Intelligence Machine Learning Framework Data Preprocessing Tools Performance Evaluation Metrics.	Limited Focus on Complex File-less Attacks, Ineffective Differentiation Between Benign and Malicious Activities, Insufficient Exploration of Feature Extraction, Inefficiency of Traditional Detection Approaches, Need for Comprehensive and Balanced Datasets.

Table 2.11 : Key concepts of Bucevschi et al.

[12] Dewan et al. (1) have proposed a comprehensive review of fileless malware, including its evolution, characteristics, propagation methods, detection techniques, and mitigation strategies, which makes it clear that traditional antivirus solutions face challenges in the detection of fileless malware and requires proactive defense strategies to mitigate this evolving threat landscape.
The research paper does not clearly indicate the datasets used for the analysis, but it mentions that the authors derived malware samples from a 'reliable' source called "Malware Bazaar", and for the analysis, employed tools such as Any.Run and VirusTotal.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Signature Based Detection, Behaviour Based Detection, Machine Learning approaches.	Supervised Learning Algorithms, Unsupervised Learning Algorithms, Anomaly Detection Algorithms, Deep Learning Algorithms.	Any.Run, VirusTotal.	Complexity of Analysis, Evolving Threat Landscape, Comprehensive Frameworks.

Table 2.12 : Key concepts of Dewan et al.

[13] Shah et al. (2022) have developed a memory-forensics-based malware detection based on computer vision and ML techniques. This approach involves the use of RGB images in place of converting memory dump files. This contrasts enhancement along with wavelet transform may be applied for feature extraction, followed by model building using SVM and XGBOOST classifiers. The method obtained 97.01% accuracy and still proved to be efficient in computation with massive improvements over other methods both in terms of precision, recall, and memory utilization.

The paper uses the following datasets for malware detection and classification: **Microsoft Malware Classification Challenge (BIG2015), Malimg Dataset, Malevis Dataset, Memory Dump Dataset (used for their experiment).**

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Data Collection, Image Transformation, Feature Extraction, Machine Learning Classification, Evaluation and Comparison	Image Processing Algorithms, Data Conversion and Feature Engineering Algorithms, Evaluation Metrics and Confusion Matrix, Support Vector Machine, Random Forest, Decision Trees, XGBoost	Python, Python Libraries, binary2image, Memory Forensics Tools, Virtual Environment for Data Collection	Ineffectiveness of Traditional Malware Detection Techniques, Limitations in Existing Memory Forensics-Based Approaches, Lack of Efficient Feature Extraction Methods, Computational and Resource Limitations, Generalization and Scalability Challenges,

Table 2.13 : Key concepts of Shah et al.

[14] Kara et al. (2022) proposed a memory-based approach to detect and analyze fileless malware. The approach provides numerous benefits in identifying and understanding the behavior of this complex and hazardous type of malware.

The dataset for the research paper uses 1249 samples of fileless malware pieces collected by a cybersecurity company in Turkey, while one named "Kovter" is an exemplar of such malware considered in the paper.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Signature Based Detection, Memory Dump collection.	Memory analysis, Process Analysis, Network Traffic Analysis, Registry analysis, Behavioural Analysis.	Volatility Framework, FTK Imager, Process Monitor, Wireshark, Netstat.	Automated analysis methods are unable to find fileless malware signatures. The behaviour of fileless malware is unclear Fileless malware has numerous qualities that are unrelated to one another. Inadequate fileless malware next-generation detection.

Table 2.14 : Key concepts of Kara et al.

[15] Sudhakar et al. (2020) suggested a detailed survey on fileless malware threats, which bypass the traditional detection because they don't use any executable files and instead exploit trusted system tools such as PowerShell and WMI. The study provides a description of the malware, detection techniques, and presents a process model for an incident response process, highlighting memory forensics and challenges of investigation while outlining future gaps in research toward mitigating fileless attacks.

It doesn't mention specific datasets in the document to be used in the study. It has a comprehensive analysis on fileless malware mechanisms, persistence techniques, and methods of detection through literature review and existing research. This is more of a conceptual and methodological study in which theoretical frameworks and challenges have been put forward, not empirical analysis with datasets.

METHODOLOGY	**ALGORITHMS**	**TOOLS**	**RESEARCH GAP**
Classification and Analysis, Comparison, Detection Techniques, Process Model Proposal, Research Gap Identification	Rule-Based Detection, Behavioral Monitoring, Machine Learning Approaches, Pattern Recognition and Anomaly Detection	PowerShell, Windows Management Instrumentation (WMI), Memory Forensics Tools, Registry Analysis Tools, Network Analysis Tools, Security Event Monitoring	Ineffectiveness of Traditional Malware Detection Techniques, Limitations in Existing Memory Forensics-Based Approaches, Lack of Efficient Feature Extraction Methods, Computational and Resource Limitations, Generalization and Scalability Challenges,

Table 2.15 : Key concepts of Sudhakar et al.

CHAPTER 3
TECHNOLOGY USE

3.1 TECHNOLOGY SPECIFICATION

Processor: Intel(R) Core(TM) i5-1035G1 - it has a good number of computing resources for both the host and the virtual machines. This quad-core architecture will not cause problems in multitasking.

OS: The host OS is Windows 10 Pro, Version 22H2. This operating system gives stability for all the applications, the virtualization software, and the rest of the tools that can be needed.

Architecture: It's 64-bit x64-based processor architecture. Therefore, it is efficient to run modern 64-bit applications and virtualization.

Virtualization Software: Oracle VirtualBox 7.0.14 - You can create, manage, and run virtual machines, providing an isolated environment for testing and analysis. [17]

Guest OS: The virtual machine runs Windows 10 (Version 22H2), offering a sandboxed environment for tasks like debugging, scripting, or forensic analysis. [18]

Base Memory: 2048 MB (2 GB) is allocated to the VM, ensuring it has sufficient RAM to operate smoothly without impacting the host system too much.

Processors: Two CPU cores are allocated to the VM for it to process its activities in a balanced manner.

Virtual Storage: The VM has been allocated 30 GB of virtual storage, which acts as its primary disk for the guest OS and files related to the guest OS.

Actual Storage Used: 10.77 GB stands for the actual space consumed by the VM's OS and installed software.

Network: The Bridged Adapter network setting makes the VM connect to the same network as the host, so that it can act like another separate physical device for network-related testing or browsing.

VirtualBox Guest Additions: Installed to make VM run smoother and more user-friendly. It provides features like clipboard sharing, improved graphics, and seamless integration with the host. [17]

Memory Dump Tool: Comae Toolkit (DumpIt) is used to capture memory dumps from the VM for debugging, forensics, or analysis. [19]

PowerShell: This scripting and automation framework simplifies system management and repetitive tasks within the VM.

7zip: A file compression and extraction tool to efficiently manage large files, saving storage and simplifying file transfers. [26]

Python 3.13: A versatile programming language used for running scripts and tools like binary data converters or forensic analyzers. [24]

binary2image Script : converts binary information like memory dumps to images helping visualize raw data. [25]

AI assistance: With this, it provides help related to technical support wherein it guides for setting up a problem through answers to questions.

3.2 TECHNOLOGIES USED

CATEGORY	TECHNOLOGY	PURPOSE
Virtualization	Oracle VirtualBox 7.0.14 [17]	Hypervisor for creating and running virtual machines.
Memory Analysis	Comae Toolkit v20230117 (DumpIt) [19]	Tool for capturing memory dumps for forensic analysis.
File Management	7zip [26]	Compression and extraction of large files.
Scripting & Automation	PowerShell	Task automation and management.
Programming	Python 3.13 [24]	General-purpose programming for tasks and automation.
Visualization	binary2image Python script [25]	Converts binary data (e.g., memory dumps) into images for analysis.
Networking	Bridged Adapter	Allows the VM to use the host's network as if it were a separate machine.
System Enhancement	VirtualBox Guest Additions	Improves VM performance (e.g., graphics, shared folders, clipboard).

Table No 3.1 : Technologies Used

CHAPTER 4
METHODOLOGY

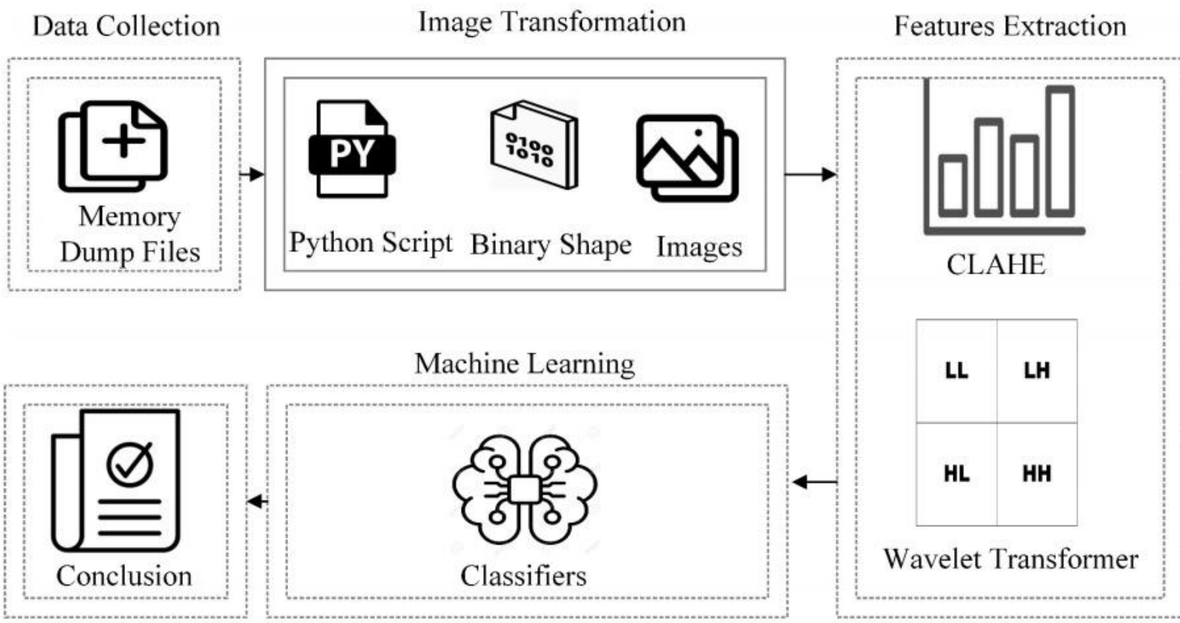


Figure 4.1 : Methodology Diagram [13]

4.1 Setup and Preparation

VirtualBox 7.0.14: Oracle VirtualBox was selected as the virtualization platform to create and manage the virtual environment. This software allows flexibility in the configuration of virtual machines, including options for snapshots and hardware resource allocation.

Windows 10 Pro: Installed the guest operating system in a virtual environment, using Windows 10 Pro. This was selected as it was compatible with tools to be used and as an environment that

simulates a standard user to test on.
Downloaded Comae-Toolkit-v20230117: Downloaded Comae Toolkit, in particular, the DumpIt tool, which facilitates memory dump collection. The DumpIt tool simplifies creating full-memory dumps, which is critical for forensic analysis.

Guest Additions : Installed to make the virtual machine more usable. It featured clipboard sharing, drag-and-drop between host and guest, and enhanced performance, especially in graphics and input.
Snapshot Taken (Base): After setting up, a base snapshot was taken. This base snapshot saves the clean state of the virtual machine so that, after experimenting, it can be quickly restored to its original state with consistent testing conditions.

4.2 Benign Memory Dump Collection

Run PowerShell Commands: PowerShell was run in the benign environment. The scripts were run from the PowerShell to interact with the process being dumped. Thus, in the context, the required specific program powershell.exe was in a running state and would be targeted for memory analysis.
Ran DumpIt: DumpIt was run to capture a full memory dump of the VM with an emphasis on benign processes. DumpIt is very simple and well-suited for this task, providing reliable memory dump files that can be further analyzed.
Modified Drag-and-Drop Option: VirtualBox was configured to drag-and-drop from guest to host. This modification enabled the gathered memory dumps to be dragged over to the host for safe storage and further analysis.
Collected Memory Dump: The memory dump file was copied over to the host system. In this way, the data was ensured to be saved and manipulated without altering the state of the VM.
Renamed with Program Name: The dump file was renamed with the name of the specific program being executed, namely powershell.exe. This would help later in the analysis.
VM Shut Down: Once all the data was collected, the virtual machine was shut down to save resources and prevent any accidental modification.
Restored to base snapshot: The VM was restored to its original base snapshot so that any residual effects from the previous session were removed, ensuring that the environment was clean for subsequent tests.

4.3 Malicious Memory Dump Collection

Same Machine Configuration: The same VM configuration was used, maintaining consistency across tests for comparable results.

Disabled Windows Defender: Windows Defender Antivirus was disabled using group policies and registry edits to prevent it from interfering with the execution of malware. This ensured that the test environment mimicked a compromised system.

Disabled Windows Defender Firewall: Group policies were also used to disable the Windows Defender Firewall to enable the malware to execute its actions without restraint, thereby simulating an actual system without security controls.

Disabled Security Notifications: All the security notifications were disabled so that there would be a focus on the testing environment rather than disruptions while the malware was running.

Set Execution Policy Bypass: Using the command ‘Set-ExecutionPolicy’, the execution policy of PowerShell was set to bypass mode. This enabled unrestricted script execution, a common tactic of malicious activities.

Created Snapshot: This stage took a new snapshot to preserve the state of the system after disabling security features, allowing repeated experiments if needed.

Connected to Local Network: The VM was connected to a local, isolated network for safety purposes. This allowed the network-based malware to perform actions without risking external systems.

Installed 7zip: The 7zip software was installed to extract or manage the malware package so that the necessary files were ready for execution.

Downloaded Malware Sample (Emotet): The Emotet malware, which is an.exe file, was downloaded to the VM as first sample. This is one of the most studied malware because of its relevance in memory dump analysis and forensic studies.

VM Restarted to Clear Memory: The VM was restarted to clear the memory before running the malware. This ensured that all the unnecessary processes that could interfere with the test were cleared.

Executed Malware: Malware was executed to view the behavior and interaction of malware in the system. This comprised the observation of changes that it made in memory and processes.
Captured Memory Dump: DumpIt was executed to collect a memory dump of the system after malware execution. The dump captured all the memory activity, including traces of malware.
Drag and Drop Memory Dump to Host: The memory dump file was transferred securely to the host system for further processing and analysis.
VM Restored to Base Snapshot: The VM was restored to the clean base snapshot to remove all traces of malware and reset the environment before future tests.

[16] Table 4.1 : Programs Executed In Virtual Machines

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

4.4 Image Generation

Installed Python 3.13: This was used for scripting and mainly for analyzing the collected memory dumps.

Installed Dependencies: All the required Python libraries were installed, including Pillow, to ensure the script was properly functional.

Downloaded binary2image Script: The binary2image script was downloaded to process memory dumps into visual images. This is a technique that helps visualize memory patterns and anomalies.

Modified Script with AI: The script was modified using AI to enhance functionality:

Allowed the generation of grayscale .png images to enhance contrast and clarity.

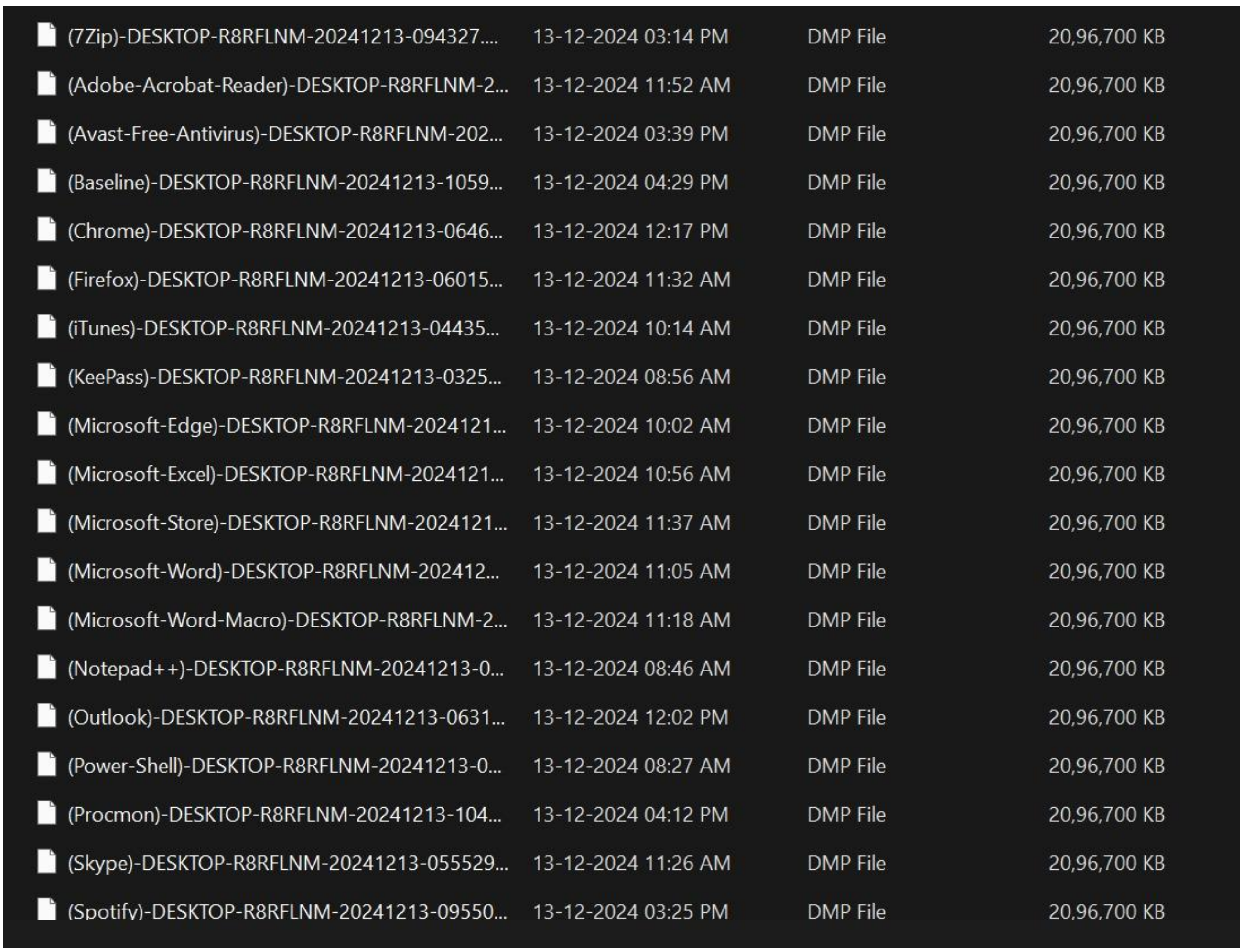
Added chunk-based memory processing to handle large files

Grayscale Images: The modified script was run to convert memory dumps to grayscale images, which then could be visually analyzed for patterns or anomalies caused by the malware.

CHAPTER 5
OUTPUT

5.1 Benign Memory Dump Collection Output

The clean memory dump taken from the benign environment served as a baseline for comparison against the malicious dump. It is an output of normal system behavior, with an emphasis on the powershell.exe process. The normal memory dumps are collected. This is important in identifying deviations that may be introduced by malware during the forensic analysis.

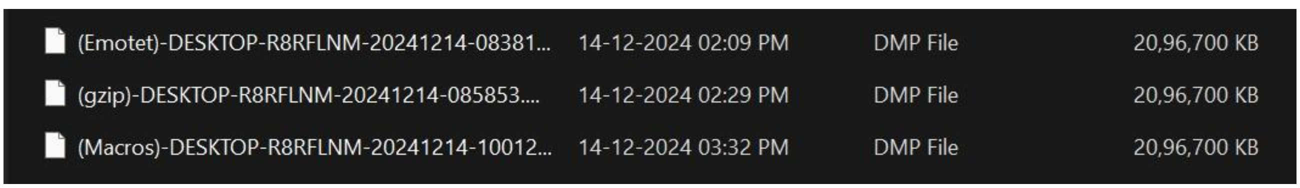


(7Zip)-DESKTOP-R8RFLNM-20241213-094327....	13-12-2024 03:14 PM	DMP File	20,96,700 KB
(Adobe-Acrobat-Reader)-DESKTOP-R8RFLNM-2...	13-12-2024 11:52 AM	DMP File	20,96,700 KB
(Avast-Free-Antivirus)-DESKTOP-R8RFLNM-202...	13-12-2024 03:39 PM	DMP File	20,96,700 KB
(Baseline)-DESKTOP-R8RFLNM-20241213-1059...	13-12-2024 04:29 PM	DMP File	20,96,700 KB
(Chrome)-DESKTOP-R8RFLNM-20241213-0646...	13-12-2024 12:17 PM	DMP File	20,96,700 KB
(Firefox)-DESKTOP-R8RFLNM-20241213-06015...	13-12-2024 11:32 AM	DMP File	20,96,700 KB
(iTunes)-DESKTOP-R8RFLNM-20241213-04435...	13-12-2024 10:14 AM	DMP File	20,96,700 KB
(KeePass)-DESKTOP-R8RFLNM-20241213-0325...	13-12-2024 08:56 AM	DMP File	20,96,700 KB
(Microsoft-Edge)-DESKTOP-R8RFLNM-2024121...	13-12-2024 10:02 AM	DMP File	20,96,700 KB
(Microsoft-Excel)-DESKTOP-R8RFLNM-2024121...	13-12-2024 10:56 AM	DMP File	20,96,700 KB
(Microsoft-Store)-DESKTOP-R8RFLNM-2024121...	13-12-2024 11:37 AM	DMP File	20,96,700 KB
(Microsoft-Word)-DESKTOP-R8RFLNM-202412...	13-12-2024 11:05 AM	DMP File	20,96,700 KB
(Microsoft-Word-Macro)-DESKTOP-R8RFLNM-2...	13-12-2024 11:18 AM	DMP File	20,96,700 KB
(Notepad++)-DESKTOP-R8RFLNM-20241213-0...	13-12-2024 08:46 AM	DMP File	20,96,700 KB
(Outlook)-DESKTOP-R8RFLNM-20241213-0631...	13-12-2024 12:02 PM	DMP File	20,96,700 KB
(Power-Shell)-DESKTOP-R8RFLNM-20241213-0...	13-12-2024 08:27 AM	DMP File	20,96,700 KB
(Procmon)-DESKTOP-R8RFLNM-20241213-104...	13-12-2024 04:12 PM	DMP File	20,96,700 KB
(Skype)-DESKTOP-R8RFLNM-20241213-055529...	13-12-2024 11:26 AM	DMP File	20,96,700 KB
(Spotify)-DESKTOP-R8RFLNM-20241213-09550...	13-12-2024 03:25 PM	DMP File	20,96,700 KB

Figure 5.1 : Benign Memory Dump Collection

5.2 Malicious Memory Dump Collection Output

The Malicious Memory Dump Collection Output is capturing the system's memory following the execution of malicious software. Here, the dump was generated using the Emotet, gzip as well as Macros malware used in this project. The dump contains critical indicators of compromise, such as altered memory structures, injected processes, or unusual memory patterns, that are direct results of the malware's behavior. Unlike the benign memory dump, the malicious dump shows how the malware interacts with system memory, whereby it can alter the sequence of data, create or hijack existing processes. Analyzing the output, we may identify specifics about traces of malicious behavior such as hidden processes, altering system files, or improper network connections. The malicious memory dump is a key element in understanding the impact of malware and provides valuable data for improving detection techniques and developing better security measures.



(Emotet)-DESKTOP-R8RFLNM-20241214-08381...	14-12-2024 02:09 PM	DMP File	20,96,700 KB
(gzip)-DESKTOP-R8RFLNM-20241214-085853....	14-12-2024 02:29 PM	DMP File	20,96,700 KB
(Macros)-DESKTOP-R8RFLNM-20241214-10012...	14-12-2024 03:32 PM	DMP File	20,96,700 KB

Figure 5.2 : Mallicious Memory Dump Collection

5.3 Image Generation Output

Grayscale Image Generation is a technique used to give a visual representation of the memory dumps, where the binary data is converted into an image for easier analysis. In this project, benign and malicious memory dumps were presented as grayscale images to show differences in patterns of memory usage. A method using a binary-to-image conversion technique visually captures the system's memory state, in which each pixel within the image corresponds to some portion of memory. Images in grayscale are most valuable because they make for easy visualization of high contrast for data within the memory, allowing easy detection of irregularities and anomalies caused by malware. This approach enables quicker detection of malicious behavior by providing an intuitive way to compare normal and compromised memory states, thus supporting both manual analysis and machine learning model training.

5.4 Restored Base VM State

Restored Base VM State involves the reset of the VM at each stage of the phase to the original, clean state. It is essential that one test not influence another test so much so that residual data from an old run or some residual modifications from a prior test affect new tests. We bring the VM back to the base state to ensure that no matter how benign or malignant a test may be, they are all initiated from the same configuration and eliminate variables that could affect results. This ensures reproducibility, which is needed for the generation of data to be reliable and constant. It also allows for the equal comparison of different memory dumps by performing each collection under equivalent conditions so that differences actually observed are due to what has changed in system memory, not other factors.

5.5 Comparison Results

The comparison results will, therefore, bring out the differences between the memory dumps of benign and malicious environments. These differences are essential in understanding system behavior. Patterns that may point to normal or malicious activities are, therefore, identified from such differences. A benign memory dump is normally characterized by consistent and uniform memory usage without any indication of abnormal behavior. On the contrary, the malicious memory dump, which is usually driven by malware such as Emotet, has different anomalies in the form of unexpected memory injections, abnormal processes, or altered data structures. Such differences are critical indicators for detecting malicious activity. The graphical representation of these memory dumps in grayscale images further makes it easier to spot anomalies quickly and accurately. The comparison outcomes form the foundation for states of safe and compromised memories, and provide useful training data for training machine learning models in the quest to automatize future malware detection.

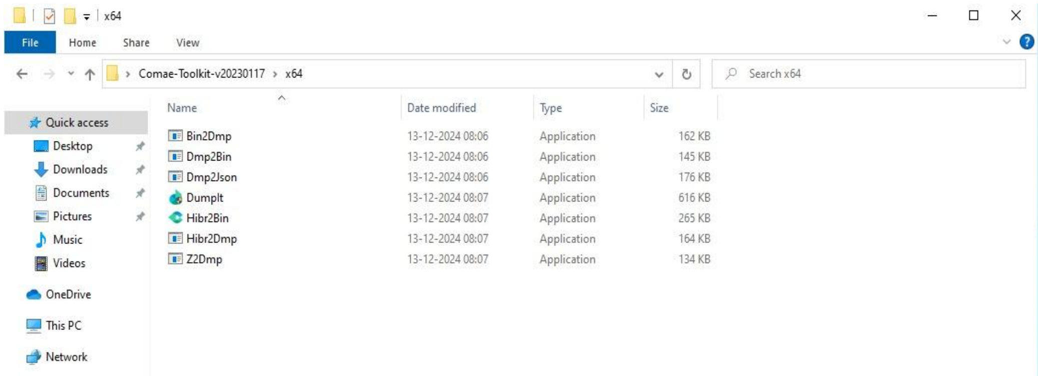


Figure 5.3 : DumpIt Tool

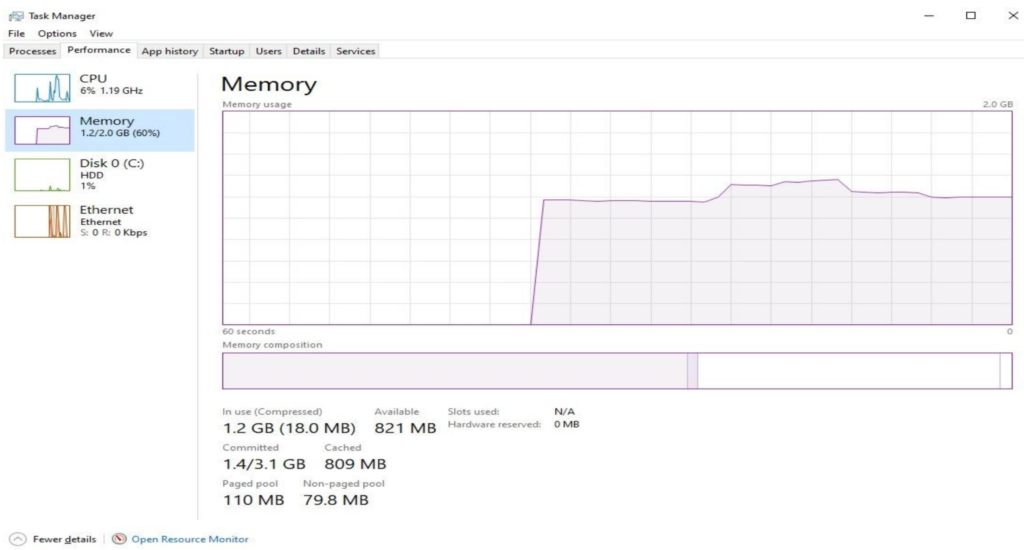


Figure 5.4 : Memory State

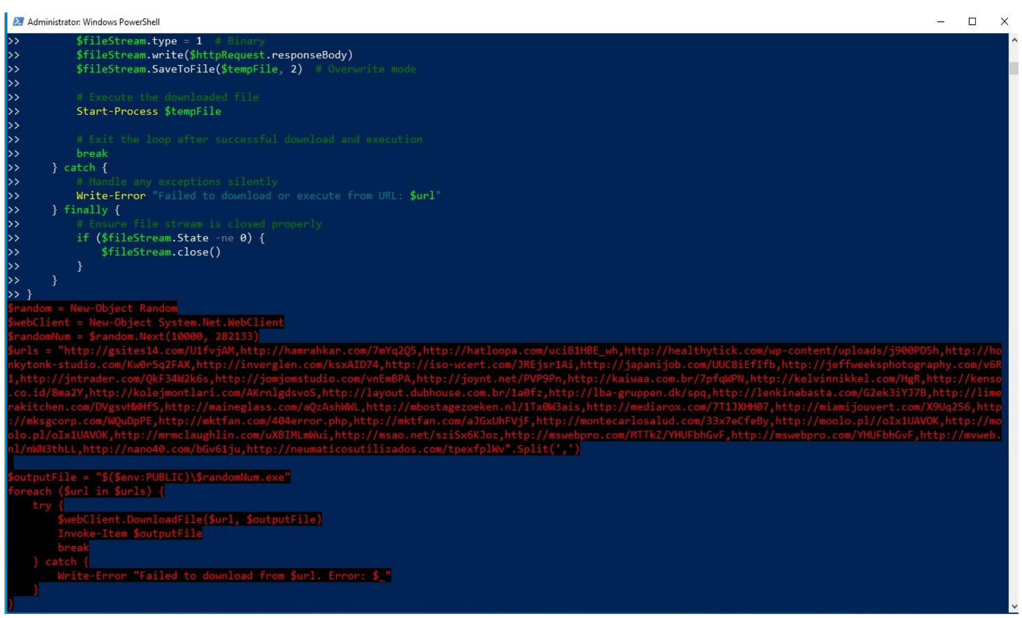


Figure 5.5 : Malicious Code Execution

CONCLUSION

It goes without saying that this project managed to integrate all of the techniques of memory dump collection, analysis, and visualization into both benign and malicious environments. It was pretty clear with the established methodology for capturing and analyzing memory dumps with tools such as Comae Toolkit (DumpIt), VirtualBox, and Python. The benign memory dump provided a baseline for normal system behavior, while the malicious memory dump, obtained after running the Emotet malware, provided key indicators of compromise. The innovative step of generating grayscale images from the memory dumps introduced a new means of visually detecting malware-induced anomalies in memory. Each experiment was run in a controlled, reproducible environment by using VirtualBox snapshots. This work points to the importance of memory forensics in cybersecurity and actually enhances the detection of malware through the visual presentation of memory states. The outcomes are a solid foundation for further research into automated anomaly detection and into memory-based analysis techniques within cybersecurity.

FUTURE WORK

Enhancement of Grayscale Images:

In the future, improving the quality of the produced grayscale images from memory dump will be our goal of focus. Increasing resolution ensures that finer details are captured with contrast adjustment making anomalies observable. Sharpening up the image will then get the key features and its patterns clear for analysis hence making the images visually more clearer and informative for further in-depth analysis.

Use of Enhanced Images for Machine Learning:

Once the grayscale images are enhanced, these processed images are to be inputs for the training of the machine learning models. With these higher quality data, models will be able to learn more complicated patterns and features; thus the differentiation between benign and malicious memory behaviors will increase. Using these enhanced images, one should be able to come up with a training dataset which would best represent the complexity of the memory data.

Training Machine Learning Models:

Trained machine learning models for recognizing malicious activities will use these enhanced grayscale images obtained after memory dump analysis. Patterns or anomalies are the target; these patterns point toward the presence of malware or any other malicious processes. They will be exposed to diverse images, such as both benign and malicious images to develop the capacity for precisely forecasting future memory dumps that have been labeled as potential threats automatically.

Automated Malware Detection:

After training the models, they will be incorporated in an automated system that shall analyze memory dumps in a real-time basis. Time and effort required to detect the malware will be minimized from this step. This enhances the speed of identification, response, and, accordingly, the security of a system. Since the technology is based on machine learning, the system will learn to improve as it continues to process more data.

Improving Memory Forensics and Cybersecurity:

This work will contribute to the general topic of memory forensics in a more efficient, effective approach to identifying such malicious activity. Memory forensics is critical for answering what happens in a system during and after an attack and automating the process by simplifying the detection of malware significantly. This is simply improving traditional forensics but goes to improve overall cybersecurity because detection occurs faster and is indeed more accurate.

Scalable and Efficient Detection System:

The long-term vision is to develop a system that scales from personal devices all the way to very large enterprise systems. Training on diverse data sets will help adapt the machine learning models to the variety of memory data types and make it work at scale. This scalability will be key to providing efficient detection across multiple platforms, ensuring that organizations can secure their systems against evolving threats in a fast and resource-efficient manner.

REFERENCE

[1] A. Afreen, M. Aslam and S. Ahmed, "Analysis of Fileless Malware and its Evasive Behavior," 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2020, pp. 1-8, doi: 10.1109/ICCWS48432.2020.9292376.

[2] B. N. Sanjay, D. C. Rakshith, R. B. Akash and D. V. V. Hegde, "An Approach to Detect Fileless Malware and Defend its Evasive mechanisms," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2018, pp. 234-239, doi: 10.1109/CSITSS.2018.8768769.

[3] Obini, U. C., Jeremiah, C., & Igwe, S. A. (2024). Development of a machine learning based fileless malware filter system for cyber-security. Journal of the Nigerian Society of Physical Sciences, 2192-2192.

[4] Demmese, F.A., Neupane, A., Khorsandroo, S. et al. Machine learning based fileless malware traffic classification using image visualization. Cybersecurity 6, 32 (2023). https://doi.org/10.1186/s42400-023-00170-z

[5] W. B. T. Handaya, M. N. Yusoff, and A. Jantan, "Machine learning approach for detection of fileless cryptocurrency mining malware," Journal of Physics: Conference Series, vol. 1450, p. 012075, Feb. 2020, doi: https://doi.org/10.1088/1742-6596/1450/1/012075.

[6] Vala Khushali, "A Review on Fileless Malware Analysis Techniques," ResearchGate, May 09, 2020. https://www.researchgate.net/publication/341870307_A_Review_on_Fileless_Malware_Analysis_Techniques

[7] A. More, K. Joshi, and K. Kumar, "Simulation Framework for Fileless Malware Detection and Analysis," Social Science Research Network, Jul. 12, 2023. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4488146

[8] Saad, S., Briguglio, W. and Elmiligi, H. (2019) The curious case of machine learning in

malware detection, arXiv.org. Available at: https://arxiv.org/abs/1905.07573 (Accessed: 15 December 2024).

[9] S. Zhang, C. Hu, L. Wang, M. J. Mihaljevic, S. Xu, and T. Lan, “A Malware Detection Approach Based on Deep Learning and Memory Forensics,” Symmetry, vol. 15, no. 3, p. 758, Mar. 2023, doi: https://doi.org/10.3390/sym15030758.

[10] O. Khalid et al., “An Insight into the Machine-Learning-Based Fileless Malware Detection,” Sensors, vol. 23, no. 2, p. 612, Jan. 2023, doi: https://doi.org/10.3390/s23020612.

[11] A. G. Bucevschi, G. Balan and D. B. Prelipcean, "Preventing File-Less Attacks with Machine Learning Techniques," 2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2019, pp. 248-252, doi: 10.1109/SYNASC49474.2019.00042.

[12] R. Dewan and SIVAKUMAR VENU, “A Deep Dive into Detecting and Investigating Fileless Malware,” Jan. 2024, doi: https://doi.org/10.2139/ssrn.4932008.

[13] S. S. H. Shah, A. R. Ahmad, N. Jamil, and A. ur R. Khan, “Memory Forensics-Based Malware Detection Using Computer Vision and Machine Learning,” Electronics, vol. 11, no. 16, p. 2579, Aug. 2022, doi: https://doi.org/10.3390/electronics11162579.

[14] I. Kara, “Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges,” Expert Systems with Applications, vol. 214, p. 119133, Mar. 2023, doi: https://doi.org/10.1016/j.eswa.2022.119133.

[15] Sudhakar and S. Kumar, “An emerging threat Fileless malware: a survey and research challenges,” Cybersecurity, vol. 3, no. 1, Jan. 2020, doi: https://doi.org/10.1186/s42400-019-0043-x.

[16] S. Abeydeera and A. G. Manzanares, “Failivaba pahavara tuvastamine pilveteenuses kasutades masinõpe tehnikaid,” Taltech.ee, May 28, 2020. https://digikogu.taltech.ee/en/Item/87cb2a3a-7ef5-43f0-89a5-ef4cb588b0d5 (accessed Dec. 15, 2024).

[17] Oracle Corporation. (n.d.). VirtualBox virtualization software. Retrieved from https://www.virtualbox.org/

[18] Microsoft. (n.d.). Windows 10 disk image (ISO) file. Retrieved from https://www.microsoft.com/en-us/software-download/windows10

[19] Magnet Forensics. (n.d.). Magnet DumpIt: A memory acquisition tool for Windows. Retrieved from https://www.magnetforensics.com/resources/magnet-dumpit-for-windows/

[20] Wireshark Foundation. (n.d.). Wireshark network protocol analyzer. Retrieved from https://www.wireshark.org/download.html

[21] Abuse.ch. (n.d.). Malware Bazaar: A platform for sharing malware samples. Retrieved from https://bazaar.abuse.ch/browse/

[22] ANY.RUN. (n.d.). Interactive malware analysis sandbox. Retrieved from https://app.any.run/submissions/

[23] Chenerlich. (n.d.). FCL Malware dataset repository [GitHub repository]. Retrieved from https://github.com/chenerlich/FCL/tree/master/Malwares

[24] Python Software Foundation. (n.d.). Python programming language. Retrieved from https://www.python.org/downloads/

[25] Carkaci, N. (n.d.). Binary-to-image conversion tool [GitHub repository]. Retrieved from https://github.com/ncarkaci/binary-to-image/blob/master/binary2image.py

[26] 7-Zip. (n.d.). 7-Zip software. Retrieved from https://www.7-zip.org/