National College of Ireland

ML Advancements in Malware Detection: Bridging Memory and Behavior

MSc Research Project
Cybersecurity

Ajay Kumar Oad
Student ID: 22183841

School of Computing
National College of Ireland

Supervisor: Dr. Arghir-Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ajay Kumar Oad
Student ID:	22183841
Program:	MSc Cybersecurity **Year**: 2023
Module:	MSc Research Project
Supervisor:	Dr. Arghir-Nicolae Moldovan
Submission Due Date:	14/12/2023
Project Title:	MSc Research Project
Word Count:	8602 **Page Count**.....22.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: .....

Date: .....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	□

Assignments that are submitted to the Program Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

ML Advancements in Malware Detection: Bridging Memory and Behavior

Ajay Kumar Oad
22183841

Abstract

Malware remains a persistent and evolving challenge in the digital landscape, often evading detection by traditional security measures. As new malware types emerge with increasing frequency, it is crucial to develop effective tools and techniques to combat these threats. This study investigates the potential of advanced machine learning (ML) techniques to enhance the detection and classification of malicious software. Employing a variety of ML models on three distinct datasets, we conducted AV detection on each dataset to assess its effectiveness in identifying and classifying malware within the datasets. Our findings suggest that hybrid detection models, which combine both memory and behavioral features, hold the most promise for improving malware detection accuracy and adaptability. We discovered that using both memory and behavioral features significantly increased detection accuracy from 73.72% to 81.02%, highlighting the effectiveness of ML models in detecting malware. Our next step is to investigate the stability of these methods when incorporating specific data features or employing optimization techniques.

Keywords: Malware detection, Fileless malware, Malware behaviors, Memory dumps.

1 Introduction

Malware, a program designed to harm computer systems, poses a significant threat to individuals and organizations. Malware can disrupt operations, damage systems, and steal sensitive data. Cybercriminals create malware to make money, often demanding ransom payments in exchange for restoring access to a victim's computer. Malware is spread through various methods, including infected software, phishing emails, and malicious websites.

Among the multitude of malware types, "fileless malware" stands out for its stealth. Unlike typical malware, it does not reveal itself as identifiable files as it avoids signature-based detection, file scanning, sandboxing, and others. Instead, it sneaks into a computer's memory, making detection quite challenging. It is related to a skilled intruder navigating hidden pathways to dodge security measures, evading detection by dwelling solely in a computer's memory. This covert malware infiltrates memory without leaving the usual traces, behaving like an undercover agent collecting personal information or causing disturbances without noticeable signs.

For instance, (Hendler et al., 2018) explains that **PowerShell Empire** and **Kovter** are notable fileless malware types tricky to detect as they do not rely on conventional files and lurk in less obvious parts of a computer, making their identification complex.

There are varies signs indicating a system is under a fileless malware attack, such as *suspicious network activity, unusual activity in the registry or WMI, excessive use of legitimate tools, abnormal spikes/behavioral in CPU or memory usage, unexpected changes to system settings or permissions and abnormal behavior from users.*

Both individuals and organizations face outcomes from these elusive fileless malware types. Individuals might suffer personal data breaches or financial losses, while organizations endure severe consequences like data breaches, operational disruptions, financial setbacks, and damaged reputations.

To counter these elusive threats, this research introduces novel contributions:

- ✓ A specialized dataset containing 215 fileless malware instances, focusing on unique API-based features, offering a fresh perspective on understanding fileless malware behavior—**a novel endeavor in this field.**
- ✓ Improvement of existing datasets through thorough analysis, leading to substantial enhancements in classification accuracy, particularly for detecting fileless malware.
- ✓ Integration of behavioral features from the initial dataset into commonly used memory dump datasets, allowing for a more comprehensive analysis, improving the capability to detect fileless malware within memory dumps.

The goal is to use new ways like spotting behavior and recognizing patterns to make our defenses stronger against tricky online dangers. It is like using smart plans to catch enemies and make sure our digital stuff stays safe.

1.1 Research Objectives

We studied how well computer programs can predict malware in different sets of data. We wanted to know if these programs can still do a better job when the information they're given is not strongly connected. Our investigation explores how to enhance the detection of fileless malware through a synergistic application of Memory features and behavioral features. This aligns with our overarching research question:

How the combination and separate use of Memory features and behavior features enhance the malware detection accuracy?

Our exploration delves into the potential impacts and implications of creating a hybrid detection model, as well as its separate utilization, on system performance and resource utilization.

- Our research contribution lies in combining memory and behavior features that boosted malware detection accuracy from 73.72% to 81.02%, showcasing the effectiveness of machine learning models.

1.2 Contribution to Academic Understanding

This study aims to contribute valuable insights to the academic discourse on cybersecurity by delving into the potential of machine learning methodologies in handling diverse malware datasets. The comparative analysis of these datasets, varying in correlation levels, promises to shed light on the predictive capacities of machine learning algorithms in detecting malware instances.

1.3 Structure of the Report

This section explains the structure of our report. By looking at what experts have already studied about this in Section II. Then, in Section III, it explains how we did our research, talking about the data we used, how we prepared it, and the computer programs we used. Sections IV, V, and VI go into the details of how we planned and designed our research, how we put it into action, and then how we checked if it worked well. Lastly, in Section VII, we wrap it up by

summarizing what we found and suggesting what could be explored next in the field of finding and stopping malware.

2 Related Work

The literature review section serves as a foundation for understanding the landscape of malware detection within the realm of behavioral and memory dump analysis, with most recent studies, aligning with our exploration using the three different types of datasets, see section 3.1.

2.1 Fileless malware: Detection Techniques and Challenges

(Sudhakar & Kumar, 2020), explores the changing landscape of cyber threats, shifting from traditional malware to more advanced fileless malware. Unlike usual malware, this type doesn't rely on regular executables, making it harder to detect with standard antivirus tools. Enterprises face challenges due to their persistence and ability to bypass security measures by using trusted operating system tools. This study also examines detection techniques and proposes an incident response model but notes gaps and associated difficulties. It stresses the urgency of focusing on detecting and preventing fileless malware attacks, especially as attackers exploit trusted applications to evade detection. Recent high-profile attacks on entities like SWIFT and the Ukraine power grid highlight the ongoing threat to critical infrastructure from such sophisticated cyber-attacks.

(Khushali, 2020), in this paper, examines malware a harmful software threatening computer systems. Traditional types like viruses, worms, and Trojans are known, but a new threat called fileless malware operates in system memory, leaving minimal trace. This type targets Windows tools like PowerShell and WMI, making detection challenging. This paper also researches into various detection and mitigation methods, aiming to clarify misconceptions. It concludes by stressing the critical nature of fileless malware, advocating for solutions that automatically analyze system behavior to prevent its infiltration into memory.

(Borana et al., 2021) examines fileless malware, a tricky threat as it avoids detection by not leaving traces. They propose a tool to help forensic experts spot unusual system activities linked to this malware. The study stresses how fileless malware persists and avoids detection, showing the growing risk it poses. It calls for more research on security methods to tackle this evolving threat, suggesting behavioral analysis despite its resource use. Additionally, they recommend using machine learning and AI-based systems to combat these attacks, highlighting the need for specific defenses against different types of fileless malware.

(Dang et al., 2019), highlighted the rising threat of cyber-attacks on Linux based IoT devices, particularly focusing on fileless attacks rather than traditional malware-based ones. Their research, utilizing IoT honeypots, revealed insights into the frequency, strategies, and impacts of these fileless attacks, often evading conventional defense methods. This shift in attention emphasizes the necessity to fortify IoT security beyond malware defense, stressing the importance of integrating heightened awareness and robust defense mechanisms against fileless attacks. The study underscores the critical need for updated defense tactics to safeguard widely used IoT systems.

In this research, (Bozkir et al., 2021) addressed countering fileless malware, which evades detection by not leaving traces on devices. They proposed a method using computer vision and machine learning to convert memory dumps into images for malware identification. Testing on diverse data showed a 96.39% accuracy in detection and employed UMAP, boosting recognition by 21.83%. This method proved both effective and speedy, taking only 3.56

seconds on standard computers. The future involves exploring neural networks for improved accuracy and leveraging UMAP's capabilities for synthetic image feature generation, promising advancements in malware defense strategies.

2.2 In Memory Detection Techniques and Mitigation Strategies

(Lee et al., 2021) studied ten recent cyberattacks using fileless malware in the last five years, aiming to understand these attacks better. Current defense systems struggle to catch these fileless attacks, leading to fragmented reports from security vendors. To bridge this gap, the study closely examined the specific methods and traits of these attacks using both real samples and published reports. They used Cuckoo Sandbox to analyze samples and proposed a classification method dividing attacks into evasion, attack, or collection categories based on techniques used. The goal is to create a solid framework for recognizing and categorizing fileless cyberattacks, helping to deal with future cybersecurity threats more effectively.

(Bucevschi et al., 2019), focuses on cybersecurity's changing landscape, especially in detecting evolving cyber threats. Traditional methods struggle with non-persistent attacks, prompting the proposal of entry-level anomaly detection using command line arguments from system tools. This approach, based on a modified Perceptron, aims to spot anomalies in files with PowerShell code often used by attackers. UAnomaly, the proposed solution, targets file-less attacks using legitimate Windows processes, offering a potential supplement to existing antivirus solutions. It could aid in creating new datasets for training, advancing defenses against file-less attacks and zero-day vulnerabilities that harm businesses.

(Varlioglu et al., 2022) critically examines the evolving threat landscape of "Fileless Cryptojacking," merging fileless malware and cryptojacking into a discreet and challenging issue post-2020. It researches academic and industry literature, emphasizing the lack of exploration around in-memory fileless cryptojacking despite attention to in-browser and in-host cryptojacking. The parallels between ransomware and cryptojacking, operating within computer memory using legitimate Windows processes for malicious activities, are highlighted. The paper proposes a novel threat-hunting approach within Digital Forensics and Incident Response (DFIR) to combat these challenges effectively, emphasizing shared patterns in Tactics, Techniques, and Procedures (TTPs).

(Zhang et al., 2023), tackle the significant challenge of detecting threats operating solely in computer memory. They propose using convolutional neural networks “CNN” and memory forensics to address this issue. By analyzing binary fragments from memory-based portable executable “PE” files, they create a dataset and train a CNN model, achieving a 97.48% accuracy in detecting fileless attacks, surpassing traditional methods. The study underscores the importance of deep learning in scrutinizing dynamic PE files housing malicious code, highlighting the need to improve the model's ability to identify diverse malicious behaviors and reduce false positives. This research has profound implications for memory forensics and sets the stage for future investigations into broader threats.

(Botacin et al., 2020), they point out how software-based antivirus struggles with scanning memory due to delays. They propose MINI-ME, a new hardware solution acting as an in-memory antivirus booster. This hardware method uses memory-based techniques, scanning for malware while data moves in memory, sidestepping software AV issues. MINI-ME is compact, using small bloom filters, and impressively spots 500 real threats without slowing down or making mistakes. This hardware-based approach looks promising for beefing up cybersecurity by strengthening memory-based threat spotting.

(Carrier, 2021), focuses on the tough task of spotting hidden and tricky malware, which can sneak past usual detection methods. Memory analysis is key here, but current techniques struggle to catch these smart forms of malware accurately. To tackle this, they upgraded VolMemLyzer, a memory tool, to better find obfuscated malware. Using a specific dataset (MalMemAnalysis2022) mimicking real tricky malware, they use machine learning and show a 99.00% accuracy and 99.02% F1-Score in finding hidden malware. The paper stresses the ever-changing nature of malware targeting Windows weaknesses and suggests a solution using memory tools to quickly find tricky malware. Their approach beats existing methods in both speed and accuracy, which is big for fighting smart malware like Spyware, Ransomware, and Trojan Horse types. This research paves a hopeful path for future detection methods.

(Dener et al., 2022), critically examines malware challenges and the limitations of traditional detection methods against advanced threats. Focusing on memory analysis, it employs cutting-edge deep learning techniques using the CIC-MalMem2022 dataset. By rigorously testing nine algorithms, Logistic Regression notably achieves 99.97% accuracy in detecting malware, highlighting memory analysis' significance. However, the study acknowledges dataset limitations and suggests refining models for diverse data. This research establishes a foundation for using machine learning in memory analysis for malware detection, encouraging future studies on multiclass classification and addressing data imbalances in this field.

(Talukder et al., 2023), present a new method blending machine learning and deep learning for stronger network intrusion detection systems (NIDSs). This special model deals well with data imbalance by using SMOTE and carefully selecting features with XGBoost. It outperforms regular methods by handling data issues, boosting accuracy, and defending against various cyber threats. Using SMOTE cuts down on mistakes, and XGBoost improves feature selection, making the whole model work better. This combo offers a dependable way to spot intrusions in real-time on internet-connected IDS devices. Future work wants to test it with new threats using updated data and compare it to other methods for even stronger systems.

(Louk & Tama, 2022), thoroughly assesses tree-based ensemble learning methods for analyzing PE malware. It compares techniques like random forest, XGBoost, CatBoost, GBM, and LightGBM using accuracy, precision, recall, and other metrics across datasets (BODMAS, Kaggle, CIC-MalMem-2022). Findings consistently favor tree-based ensembles, highlighting their superiority in malware detection. This study contributes by statistically evaluating these models and suggesting future exploration in interpretability and neural network models like TabNet. It critically engages with theories, emphasizing the potential of tree-based models in diverse malware detection systems. Room for future research in explainable models and broader neural network use is acknowledged.

(Roy et al., 2023), their study focused on the intricate challenge of detecting complex malware, highlighting gaps in current research and the need for more comprehensive detection methods. Their MalHyStack model merges traditional and deep learning, showcasing improved accuracy and faster processing by selecting critical features. Tested on the CIC-MalMem2022 dataset, this model surpasses existing techniques, excelling in spotting obfuscated malware. The paper stresses the urgency of early detection for system security, suggesting future research directions like automated tuning algorithms to enhance performance and reduce computational complexities. By critically evaluating theories and offering insightful analysis, this work shows a deep understanding of literature, providing valuable avenues for further exploration in the field.

(Mezina & Burget, 2022), this research dives into the ongoing challenge of spotting hidden malware in cybersecurity, recognizing the ever-changing threats. It suggests using artificial intelligence to detect disguised malware in computer memory, highlighting the struggle of antivirus programs to keep up. They used advanced neural networks, like the dilated convolutional network, with the latest CIC-MalMem-2022 dataset, achieving a 0.99 accuracy in detection. Traditional machine learning combined with optimization techniques showed the random forest model excelling in binary classification, but for multiclass issues, the proposed neural network performed better with 0.8353 accuracy. This emphasizes the need for better methods to accurately classify diverse malware types. Overall, this study urges advancements in identifying and categorizing evolving malware, summarizing the current state and future pathways in the field.

(Nugraha & Zeniarja, 2022) carefully examines how memory-based analysis helps detect evolving polymorphic threats in malware. It studied a Decision Tree-based classification method using a big dataset of 58,596 records containing both harmless and harmful data. The results were impressive: accuracy stood at 99.982%, with only a 0.1% false positive rate and a precision of 99.977%. This approach efficiently pinpointed key features, making computations faster. The model excelled at recognizing and signaling malware behavior, highlighting the Decision Tree's effectiveness. Overall, this study significantly boosts detection accuracy while reducing false alarms, emphasizing the importance of memory-based analysis against ever-changing malware threats.

(Naeem et al., 2023) introduces an advanced malware detection system using memory forensics and a deep ensemble model. It effectively detects elusive malware like fileless and memory-resident threats. The model achieved high accuracies (99.1% for Windows, 94.3% for Android, and 99.8% for obfuscated Windows malware) but faces complexity in training due to numerous features. It suggests a memory agent for streamlined training. Future research aims to explore additional memory characteristics, real-time monitoring during execution, and stronger defenses against attacks.

2.3 Advanced Techniques and Novel Approaches

(Al-Qudah et al., 2023), introduced a fresh way to tackle complex memory dump malware. They combined OCSVM and PCA to create the OCC-PCA model, boosting malware detection to an impressive 99.4%, a big jump from the 55% seen with older methods. Using the 'MalMemAnalysis2022' dataset, this method not only showed better performance but also highlighted PCA's strength in spotting irregularities in memory dump files. This combo of OCSVM and PCA stands out as a strong way to tell apart good and bad behaviors in the ever-changing malware scene.

(Keyes et al., 2021), focus on Android malware's swift growth, highlighting the need for stronger analysis methods. Current techniques focus on fixed and moving features like API calls but lack insight into malware behavior using various dynamic traits. The study introduces EntropLyzer, employing entropy-based behavioral analysis, successfully categorizing 12 Android malware types and 147 families from the CCCS-CIC-AndMal2020 dataset. It examines six dynamic traits but faces limitations in emulator-based analysis, reducing the sample size due to malware detection. To overcome this, employing real devices for analysis could broaden the study's scope. Despite strides, the study underscores ongoing challenges in battling evolving Android malware, calling for refined analysis approaches.

(Sihwail et al., 2019), introduced a new way to find malware, which works better than regular antivirus software. They mixed memory investigations and dynamic analysis using a support vector machine (SVM) model. This combo spotted malware with 98.5% accuracy, cutting down wrong alerts to just 1.7%. By blending memory clues with dynamic analysis, it gave a wider view of how malware acts, going beyond just one way it works. Even though using API calls was good for sorting malware, the study says we should also investigate registries and networks to find it even better. They plan to grow the dataset of malware and make the test areas stronger to fight against tricky tactics used to avoid detection.

2.4 Summary Table of previous work

Table 1 The following summary table lists previous studies in the field of malware detection.

Study	**Methods**	**dataset**	**Evaluation performance - Binary**	**Evaluation performance – multi-class**
(Khushali, 2020)	Survey of previous studies, in fileless malware detection techniques	N/A	N/A	N/A
(Sudhakar & Kumar, 2020)	Survey of previous studies of fileless malware and it's challenges	N/A	N/A	N/A
(Bozkir et al., 2021)	SMO (RBF)	Own dataset of 4294 samples (3686 malwares and 608 benign)	96.39%	N/A
(Bucevschi et al., 2019)	Monitoring suspicion, harmful and other hiding actions	Own dataset containing 500,551 command lines, WMI Scripts and others	N/A	N/A
(Lee et al., 2021)	Kuckoo Sandbox malware attack traced with MITRE	Samples from GitHub and Hybrid-analysis	N/A	N/A
(Dang et al., 2019)	By examining 4 software and honeypots across public cloud environments through profiling.	N/A	N/A	N/A
(Borana et al., 2021)	Network and System's run time behavior	N/A	N/A	N/A
(Varlioglu et al., 2022)	Fileless malware pattern, Techniques and Procedure, malware for Tactics	N/A	N/A	N/A
(Zhang et al., 2023)	Memory Forensics coupled with CNN-Based Classification	4896 memory dump samples.	97.48%	N/A
(Botacin et al., 2020)	MINI-ME: Hardware AV using memory techniques	21 thousand in-the-wild malware samples	100%	N/A
(Carrier, 2021)	Stack ensemble	CICMalMem-2022	99.00	N/A
(Dener et al., 2022)	LR	CICMalMem-2022	99.97	N/A
(Mezina & Burget, 2022)	Dilated CNN	CICMalMem-2022	99.00	N/A
(Nugraha & Zeniarja, 2022)	DT	CICMalMem-2022	99.98	N/A
(Louk & Tama, 2022)	GBM/ XgBoost/RF	Kaggle, BODMAS, CICMalMem-2022		N/A
(Naeem et al., 2023)	Deep stacked ensemble (MLP+CNN)	CICMalMem-2022	99.8	N/A
(Al-Qudah et al., 2023)	SVM	CICMalMem-2022	99.4	N/A
(Keyes et al., 2021)	Decision Tree	CIC-AndMal-2020	N/A	98.3 (12 class)
(Roy et al., 2023)	Hybrid Stack Ensemble	CICMalMem-2022	99.98	85.04 (4 Class) 70.29 (16 Class
(Sihwail et al., 2019)	SVM	VirusTotal	98.5%	N/A
Our proposed (2023)	**RF**	**CICMalMem-2022 and Behavioral**	**100% with 55 features**	**81.02% with 69 features**

3 Research Methodology

In this study, our aim is to enhance malware detection using three distinct datasets our own behavioral-based, (Khalid et al., 2023) and CICMalMem-2022 consisting of memory dumps information. The journey towards refining these datasets and improving detection accuracy involves a structured methodology inspired by the KDD which is also known as "Knowledge Discovery in Databases" framework.

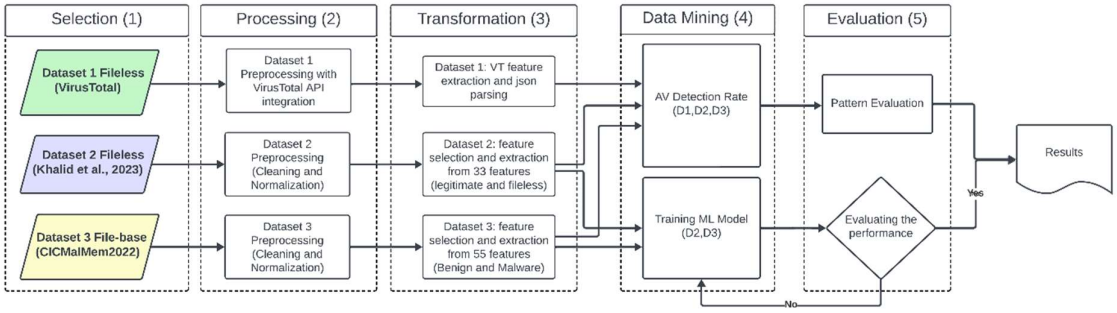

Figure 1. The process of KDD methodology being followed in our research study.

3.1 Data Selection

We gathered diverse datasets including our own fileless malware samples collection of 215 from VirusTotal, dataset obtained from (Khalid et al., 2023), and CICMalMem2022, obtained from the Canadian Institute for Cybersecurity. These datasets cover a wide spectrum of sources and types, offering a comprehensive foundation for our research.

Table 2 Lists our all datasets used in this study.

Dataset name	**Samples**	**Dataset Containing**	**Source**	**Dataset Source**
Dataset 1 (Own dataset)	215 malware samples	Malware samples returned when searched for "fileless" on VirusTotal	VirusTotal	***Created***
Dataset 2 (Khalid et al., 2023)	45 (24 legitimate and 21 fileless)	33 features extracted from memory dumps	(Khalid et al., 2023)	Obtained
Dataset 3 (CICMalMem2022)	58596 (29298 benign, 29298 malware)	55 features extracted from memory dumps	Canadian Institute for Cybersecurity	Downloaded

Dataset 1: Our search for fileless malware samples began by leveraging VirusTotal's unique capabilities. Initially, we used VirusTotal's search engine, finding 215 SHA-256 hashes specifically associated with fileless malware, searching with 'fileless' keyword. VirusTotal stood out as a primary source, notably surpassing other platforms due to its proficiency in providing fileless malware samples, a relatively new and challenging type of malware to find.

We chose VirusTotal, because other platforms such as Hybrid-Analysis, Any Run, and VirusShare did not provide the required and enough information on fileless malware. Where VirusTotal offered 215 labeled as 'fileless' samples when searched on VirusTotal, which was crucial for my research, additionally it has many benefits such as easy to use and integrate API (Kantola, 2022).

Dataset 2: This dataset for exploring malware detection is obtained from the work of (Khalid et al., 2023), as detailed in their research paper. It's a collection of memory dumps categorized

into two groups: one holds data on legitimate applications, while the other comprises fileless malware samples. They used this dataset to learn how to distinguish fileless malware from normal applications using information from memory dumps.

Originally, the dataset was published with 40 samples which includes memory dumps dataset of legitimate application fileless malwares by (Abeydeera, 2020), which later modified by (Khalid et al., 2023), as they argued that the original dataset was unbalanced and wanted to add more to get better classification results, to do that they added additional 26 fileless malware samples but they dropped 21 samples as they did not execute as per needs and added 5 which were executed successfully. This way by adding 5 fileless malware samples (Khalid et al., 2023) used 45 malware samples in total in their study.

Table 3 shows the sample distribution for dataset 2.

Category	Percentage	Count
File-base	53.33%	24
Fileless	46.66%	21

Dataset 3: CICMalMem-2022, which is released by the Canadian Institute of Cybersecurity also known as, serves as a tool to test how well systems can detect hidden malware. It contains two types of data: one representing harmful software and the other representing harmless software. This collection mirrors real-life situations by including a variety of malicious programs commonly encountered. (CICMalMem2022 is available at https://www.unb.ca/cic/datasets/malmem-2022.html).

Table 4 shows data distribution of dataset 3.

Category	Percentage	Count
Benign	50%	29298
Malware	50%	29298

Table 5 shows details about all malware families for dataset 3.

Malware Category	Family name	Malware Percentage	Count
Ransomware	Ako	6.83%	2000
	Conti	6.79%	1988
	MAZE	6.68%	1958
	Pysa	5.86%	1717
	Shade	7.26%	2128
Spyware	Coolwebsearch	6.83%	2000
	Gator	7.51%	2200
	Transponder	8.23%	2410
	TIBS	4.81%	1410
	180Solutions	6.83%	2000
Trojan Horse	Emotet	6.71%	1967
	Refroso	6.83%	2000
	Reconyc	5.36%	1570
	Scar	6.83%	2000
	Zeus	6.66%	1950

3.2 Data Preprocessing

Dataset 1:

- *Accessing Behavioral Information*: As our dataset 1 is related to behavioral information of Fileless malwares, we utilized VirusTotal's API *'Get objects related to a file'* to access API-based behavioral information associated with 215 identified fileless malware SHA-256 hashes. The reason we selected the behavioral section, as it has

comprehensive behavioral information as compared to other sections such as overview, details which contains general and identifications information accordingly.

- *Automated Retrieval*: Integrated the VirusTotal API into a Python script to automatically download and collect the behavioral information in JSON files.
- *Data Validation and Cleaning*: The integration of API to our own python script to automate the downloading of JSON files gave us chances to validation procedures within the Python script to ensure data integrity through followings:
    - Removed duplicate sample hashes to maintain dataset accuracy.
    - Excluded samples lacking behavioral information from the dataset to ensure completeness.
- *Security Measures*: Conducted the iterative retrieval process within a Kali Linux VM for a secure environment while handling sensitive data.

Dataset 2:

1. *Label Refinement*: Binary labels representing legitimate applications and fileless malware were transformed into descriptive strings **non-malware** for legitimate apps and **malware** for fileless malware.
2. *Focused Feature Selection*: We pinpointed six features highly significant in memory dumps, enhancing our ability to predict outcomes based on crucial system characteristics.

Dataset 3:

1. *Refining Data Fields*: The removal of the **'Category'** column containing extensive malware names and hashes allowed for a more normalized dataset. Additionally, the class column was renamed to **'label'** for ML clarity and ease of understanding.
2. *Removed Duplicates:* As we identified that each sample was run 10 times as per the need of research (Carrier, 2021), there were many duplicated malwares. To maintain the integrity and data validation we removed the duplicates and worked with unique malware only.
3. *Focused Feature Selection*: We identified six specific characteristics related to memory dumps; we also extracted our 6 features as explained in Table 8.

3.3 Feature Selection and Extraction

In this section, we extracted the features programmatically and manually for each dataset using our own C# written scripts.

Dataset 1: Upon acquiring behavioral details, the obtained JSON files were securely transferred to our primary environment securely. Following Tables 5 lists the features for our own dataset.

Table 6 List of 14 features extracted based on VirusTotal API behavior data for all 3 datasets.

No.	Feature	Explanation
1	Detections	Number of antivirus engines that detected the file as malicious, in network end.
2	MitreSignatures	Number of signatures from the MITRE ATT&CK Framework that the file matched.
3	IDS_Rules	Number of IDS (intrusion detection system) rules that the file triggered.
4	Sigma_Rules	Number of Sigma detection rules that the file triggered.
5	DroppedFiles	Whether the file created or dropped any files on the system.
6	NetworkComms	Whether the file established any network connections.
7	FileSystemActions	Whether the file performed any file operations, such as creating, modifying, or deleting files.
8	RegistryActions	Whether the file performed registry operations, such as creating, modifying, or deleting registry keys or values.
9	ProcessServiceActions	Whether the file spawned any new processes or services.
10	SyncMechanisms	Whether the file used any synchronization mechanisms, such as mutexes or semaphores.
11	MutexesCreated	Whether the file created any new mutexes.
12	MutexesOpened	Whether the file opened any existing mutexes.
13	ModulesLoaded	Whether the file loaded any additional modules.
14	RuntimeModules	Whether the file loaded any specific runtime modules, such as rundll32.exe or kernel32.dll.

Dataset 2: This dataset contains 33 features, as listed below.

Table 7 lists all features from datasets 2.

No	Feature name	No		No	Feature name
1	handles_num	12	processes_psxview_exited_num	23	threads_thrdscan_num
2	hiveList	13	processes_psxview_false_columns_num	24	pslist
3	dlls_ldrmodules_num	14	processes_psxview_false_rows_num	25	tcp/udp_connections
4	dlls_ldrmodules_unique_mappedpaths_num	15	processes_psxview_num	26	total_reg_events
5	dlls_ldrmodules_InInit_fales_num	16	processes_psxview_pslist_true_num	27	read_events
6	dlls_ldrmodules_InLoad_false_num	17	processes_psxview_psscan_true_num	28	write_events
7	dlls_ldrmodules_InMem_False_num	18	services_svcscan_num	29	del_events
8	dlls_ldrmodules_all_false_num	19	services_svcscan_running_num	30	executable_files
9	modules_num	20	services_svcscan_stopped_num	31	unknown_types
10	callbacks_num	21	dlls_dlllist_unique_paths_num	32	http(s)_requests
11	processes_privs_enabled_not_default_num	22	mutex_mutantscan_num	33	dns_requests

Table 8 lists Six specific features selected for Dataset 2.

No	Feature name	Description
1	handles_num	Number of handles used by the system
2	dlls_ldrmodules_num	Count of loaded DLLs
3	callbacks_num	Number of callback functions registered in the system
4	services_svcscan_num	Count of services scanned by the system
5	mutex_mutantscan_num	Number of mutex (mutant) objects scanned
6	threads_thrdscan_num	Total count of threads scanned within the system

Justification of Feature Selection: During our research for datasets 2 and 3, a correlation analysis of their memory dumps unveiled above six common features (Table 8). Leveraging these identified features for machine learning applications led to notable improvements in predictive outcomes across both datasets. This empirical observation underscores the significance of these features in contributing to enhanced model performance. Thus, the selection of these six features was rooted in empirical evidence, consolidating their relevance and utility in improving the accuracy and efficiency of our machine learning approach.

Dataset 3: This dataset contains 55 memory-based features, as listed below.

Table 9 lists all features from dataset 3.

No.	Feature name	No.	Feature name	No#	Feature name
1	pslist.nproc	20	handles.nmutant	39	psxview.not_in_eprocess_pool_false_avg
2	pslist.nppid	21	ldrmodules.not_in_load	40	psxview.not_in_ethread_pool_false_avg
3	pslist.avg_threads	22	ldrmodules.not_in_init	41	psxview.not_in_pspcid_list_false_avg
4	pslist.nprocs64bit	23	ldrmodules.not_in_mem	42	psxview.not_in_csrss_handles_false_avg
5	pslist.avg_handlers	24	ldrmodules.not_in_load_avg	43	psxview.not_in_session_false_avg
6	dlllist.ndlls	25	ldrmodules.not_in_init_avg	44	psxview.not_in_deskthrd_false_avg
7	dlllist.avg_dlls_per_proc	26	ldrmodules.not_in_mem_avg	45	modules.nmodules
8	handles.nhandles	27	malfind.ninjections	46	svcscan.nservices
9	handles.avg_handles_per_proc	28	malfind.commitCharge	47	svcscan.kernel_drivers
10	handles.nport	29	malfind.protection	48	svcscan.fs_drivers
11	handles.nfile	30	malfind.uniqueInjections	49	svcscan.process_services
12	handles.nevent	31	psxview.not_in_pslist	50	svcscan.shared_process_services
13	handles.ndesktop	32	psxview.not_in_eprocess_pool	51	svcscan.interactive_process_services
14	handles.nkey	33	psxview.not_in_ethread_pool	52	svcscan.nactive
15	handles.nthread	34	psxview.not_in_pspcid_list	53	callbacks.ncallbacks
16	handles.ndirectory	35	psxview.not_in_csrss_handles	54	callbacks.nanonymous
17	handles.nsemaphore	36	psxview.not_in_session	55	callbacks.ngeneric
18	handles.ntimer	37	psxview.not_in_deskthrd		
19	handles.nsection	38	psxview.not_in_pslist_false_avg		

3.4 Data Mining

Our study seeks to detect several types of malwares using following two techniques:

- *AV Detection Rates*
- *ML classifications*

Machine Learning Algorithms for Malware Detection: A Comparative Analysis

- *Random Forest (RF):* Random Forest is an excellent method for malware detection. By merging multiple decision trees, it produces more accurate results and reduces the risk of overfitting. It's applicable to both binary (malicious vs. benign) and multi-class classification tasks, making it a versatile tool for malware analysts. Random Forest effectively handles noisy and outlier data, high-dimensional data with numerous features, and its non-parametric nature makes it adaptable to diverse malware types. It's also highly accurate, efficient, and scalable, making it an asset for security analysts.
- *J48:* This is a well-structured decision tree algorithm that efficiently separates data points based on their attributes. It's widely used in classification tasks, including malware detection. J48's decision tree structure is easy to comprehend, making it simpler for security analysts to interpret the classifier's predictions and gain insights into the malware's behavior. Its ability to handle correlated features, making it suitable for analyzing malware datasets, and its efficiency in learning from small datasets make it a valuable tool for security analysts.
- *Naive Bayes (NB):* Naive Bayes is a probabilistic classifier that utilizes Bayes' theorem to assign class probabilities. It's flexible for both binary and multi-class classification tasks, making it a versatile tool for malware analysts. Naive Bayes is computationally efficient, making it ideal for large-scale malware detection. It effectively handles imbalanced data, where benign samples significantly outnumber malicious ones, and its probabilistic nature provides insights into the relative importance of features in determining the class label. These features make it a valuable tool for security analysts.
- *Sequential Minimal Optimization (SMO):* SMO is an algorithm for training support vector machines (SVMs), which are supervised learning models that distinguish between data points based on their attributes. SMO is used for classification tasks, including malware detection. Its ability to handle high-dimensional data, which is common in malware analysis, its relative robustness to noise and outliers, and its ability to provide insights into the decision boundaries and feature importance makes it an effective tool for security analysts.
- *Instance-Based Learning (IBk):* IBk is a lazy learning algorithm that classifies new data points by comparing them to previously stored instances. It's used for classification tasks, such as malware detection. IBk is highly scalable, making it efficient for processing large-scale malware datasets. It's also versatile for both binary and multi-class classification tasks, making it capable of handling both benign and malicious samples without requiring explicit labels. IBk effectively handles imbalanced data, making it suitable for malware detection where benign samples outnumber malicious ones, and its nearest neighbor approach provides insights into the similarity between malware samples. These features make it a valuable tool for security analysts.

3.5 Evaluation

- *AV Detection:* The effectiveness of an anti-virus (AV) program in detecting and blocking malware is crucial for cybersecurity. By effectively detecting and blocking malware at the initial point of entry, AV programs act as a frontline defense against malware attacks, protecting devices from potential damage and data breaches. It

measures how well an AV program can identify and prevent malware from infecting a device. A high AV detection rate is essential for effective cybersecurity, as it ensures that malicious software is detected and blocked before it can cause harm. By serving as a first line of defense against malware attacks, AV programs play a critical role in protecting sensitive data and maintaining system integrity.

- *Precision:* Precision assesses a classifier's accuracy in predicting positive samples. It indicates the proportion of correctly identified malware samples among all samples predicted to be malware. A high precision score implies that the classifier is less likely to mistakenly label benign samples as malware, minimizing false positives.
- *Recall:* Recall evaluates a classifier's completeness in identifying positive samples. It measures the fraction of actual malware samples accurately identified as malware. A high recall score indicates the classifier's ability to effectively detect most malware samples, minimizing false negatives.
- *F1-score:* The F1-score balances precision and recall, providing a single metric to evaluate a classifier's overall performance. It is calculated by averaging precision and recall, considering their reciprocals. A high F1-score suggests the classifier strikes a good balance between not falsely labeling benign samples and identifying malware.
- *Accuracy:* Accuracy measures the overall correctness of a classifier, indicating the proportion of correctly predicted samples among all samples. It provides a straightforward assessment of the classifier's ability to correctly classify samples. However, accuracy can be misleading when dealing with imbalanced datasets, where one class significantly outnumbers the other.
- *ROC-AUC*: The ROC-AUC curve visualizes a classifier's performance over a range of sensitivity (recall) and specificity (1 - false positive rate) trade-offs. It provides a comprehensive view of a classifier's performance, considering both precision and recall simultaneously. A high ROC-AUC score indicates a classifier's ability to effectively identify both true positives and true negatives across different operating points.

4 Design Specification

This section describes and lists the technologies and designs used in our research.

Technologies and Tools Utilized:

We researched and utilized 2 different types of VirusTotal APIs by creating a free account and acquiring an API key necessary for accessing the public API endpoints.

- *Programming Languages and Environments:*
  - **Python**: Used for data retrieval, utilizing libraries such as *os*, *json*, *time*, *requests*, and *hashlib* for data manipulation and API interaction.
  - **C# in Visual Studio 2022**: Utilized for further processing and extraction of specific features from the retrieved data. Utilizing libraries such as *Newtonsoft.Json*, *GemBox.Spreadsheet* and *OfficeOpenXml*;

Dataset Composition:

- *Features Obtained:* Detailed the specific features extracted from the VirusTotal API data (e.g., file hashes, scan results, timestamps).

Privacy and Ethical Considerations:

- *API Key Usage:* Stressed the responsible and ethical use of the API key in compliance with VirusTotal's terms of service and privacy policies.

Limitations and Challenges:

- *Rate Limit Constraints*: Highlighted challenges faced due to rate limitations on public APIs and strategies employed (like delay implementation) to manage these constraints.

For which we implemented a *20-second delay* between requests due to limitations on the public API (4 requests per minute) to avoid exceeding the rate limit.

Class Diagram: The following class diagram shows a map of different classes and how they are associated to one another. This diagram helps to understand the basic and logical structure of code, which was created for Dataset 1.

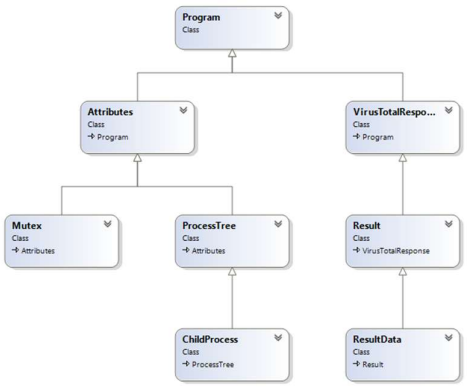

Figure 2 shows the class diagram of C# program.

5 Implementation

This section foces on breif description of implementation of our 3 datasets and technolgoies used along with their justifications.

Dataset 1:

Following figure 3 gives breif overview of all steps taken in creating dataset 1.

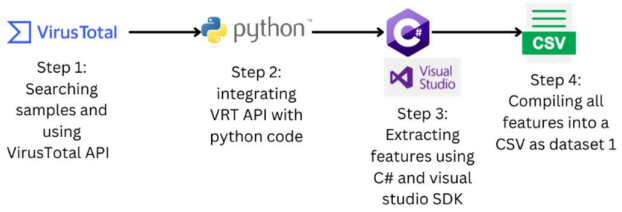

Figure 3 shows he process flow of creating our own Dataset 1.

Step 1: VirusTotal API Identification & Parameters Setup

Our initial step involved pinpointing the relevant VirusTotal API. This API required specific parameters file identification (SHA-256, SHA-1, or MD5), relationship type ('behavior'), a limit on related objects and an API key for authentication.

Table 10 Lists parameters used in API.

Parameter	Description
Id	A string type value representing the file's identification (SHA-256, SHA-1, or MD5).
relationship	A string denoting the relationship type, such as 'behavior' in this case.
Limit	An int32 type representing the maximum number of related objects to retrieve, set at 40 for this extraction.
x-apikey	A string type representing the user's API key necessary for API access and authentication.

Justification: We connected an API to a Python program to automatically fetch large number Json files containing malware data. Specifically, we focused on the behavior section of the malware because the important information we needed was only found there. This helped us gather the necessary data efficiently.

Step 2: Python Code Integration (API Access)
To interact with the identified API, a Python script was developed. This script successfully accessed the VirusTotal API, retrieving behavioral details for the 215 fileless malware samples in the form of json extension, on a VM environment.

- *Json files sizes: 1 kb to 663 kb*
- *Python version: 3.11.2*
- *Kali Lunix version: 2023.1*

Justification: The reason for running python script on VM was simple. Our VM was set up on Kali Linux, which has pre-installed python libraries which were up to date.

Step 3: C# Code Development (Behavioral Data Processing)
Transitioning to our primary environment, we thoroughly constructed a C# codebase within Visual Studio 2022. This program was developed to validate, process, and extract the identified 15 crucial behavioral features from the obtained 215 JSON files.

- *Visual Studio version: 2022*
- *C# version: 12*

Justification: We used C# programming as it provides a wide variety of libraries to manage files easily. The debugging was much better and easy to work with.

Step 4: CSV Compilation (Data Integration)
The final phase involved compiling the extracted behavioral features into a comprehensive CSV file. This file condensed crucial data points for each malware sample, offering a consolidated overview of their behavioral characteristics. Additionally, to maintain the validation and integrity of malware detection, we removed the malware which had no data.

- *15 malware samples did not have behavioral information.*
- *Final CSV file compiled with 200 malwares with size of 20 kb.*

Justification: We compiled final data into a CSV file because we will use this file into Weka tool, for ML classification as Weka accepts this file extension and Weka is easy to use too.

Dataset 2:
We refined Dataset 2 by transforming labels into clearer descriptions, removing irregular data, and focusing on significant features found in memory dumps. This helped enhance the dataset for better analysis.

Technologies Used:

- We developed C# programing within Visual Studio 2022 for automating the refinement process due to the dataset's size.

Justification: Visual Studio 2022: Its automation capabilities expedited the process, crucial when dealing with a large dataset. Its user-friendly interface also facilitated manual interventions where needed.

Dataset 3:
For Dataset 3, we refined the data by normalization, renaming columns for clarity, and selecting critical memory dump-related features. This ensured a more organized and useful dataset for analysis.

Technologies: Like Dataset 2, we developed C# code through Visual Studio 2022 for refinement. This choice was due to the consistency needed between the datasets and the efficiency required for handling memory dump-related information.

Justification: Visual Studio 2022: Its automation support was pivotal. The need for manual intervention was efficiently managed within this environment, streamlining the overall process.

6 Evaluation

During the evaluation, we aimed to do AV detections on 3 datasets and thoroughly check how well different models can spot malware in all datasets. We did many tests, trying out different features and classification setups to really understand how good our models are at this task.

6.1 Dataset 1

In this section we performed statistical analyses using AV detection methods.

6.1.1 AV Detection Rate

We looked at Dataset 1 and checked 215 fileless malware samples. We found some important trends in how these malware things get noticed, which helped us understand them better.

Table 11 shows the detection rate information for Dataset 1.

Statistic	Number Detections	Total Number of AVs	Detection Rate
Mean	51.2	67.2	0.759
Min.	21	55	0.370
Max.	64	73	0.900
Std.	6.2	2.7	0.078

Overall, these malwares identify such instances 76% of the time, meaning roughly three quarters of the cases. We also found differences from a 37% detection rate to a high of 90%. The standard deviation, a measure of how much detection rates deviate from the average, was around 0.078, showing a moderate level of fluctuation around the mean detection rate.

6.2 Dataset 2

In our second dataset, we conducted AV detection Rates and binary classification tests with a **67% split**, dividing the dataset into training and testing sets. Binary classification experiments aimed to distinguish between legitimate applications and fileless malware.

6.2.1 AV Detection Rates

The following table shows the AV detection rates on malware samples of dataset 2.

Table 12 shows the detection rate information for Dataset 2.

Statistic	Number Detections	Total Number of AVs	Detection Rate
Mean	45.6	63.8	0.706
Min.	0	51	0
Max.	65	72	0.9
Std.	14.2	6.1	0.185

Overall, the systems spotted around seven out of ten malware instances, with an average detection rate of 70.6%. Some were completely missed (0% detection), while the best detection rate hit 90%. The slight difference of around 0.19 shows that the systems varied in how well they spotted these malwares. This suggests big differences in their identification abilities.

6.2.2 ML Results on Original 33 Features

This experiment involved utilizing the complete set of 33 original features dataset 2.

Table 13 shows Binary classification results on 33 features.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.893	0.867	0.863	86.66	1.000
J48	0.867	0.867	0.867	86.66	0.884
Naive Bayes	0.941	0.933	0.933	93.33	1.000
SMO	0.893	0.867	0.863	86.66	0.857
IBk	0.893	0.867	0.863	86.66	0.857

6.2.3 ML Results on Selected 6 Features

This test is focused on a dataset comprising only 6 critical features identified as significant in memory dump analysis, see Table 8.

Table 14 shows Binary classification results on 6 specific features.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.941	0.933	0.933	93.33	1.000
J48	0.941	0.933	0.933	93.33	0.929
Naive Bayes	0.795	0.667	0.614	66.66	0.821
SMO	0.795	0.667	0.614	66.66	0.643
IBk	0.804	0.800	0.798	80	0.795

6.3 Dataset 3

For our third dataset, experiments encompassed binary and multi-classification tasks using a **70% split**. We started our analysis on Dataset 3, from AV Detection and then to ML training.

6.3.1 Detection rates and Statistical Analysis

Table 15 shows the detection rate information for Dataset 3.

Statistic	Number Detections	Total Number of AVs	Detection Rate
Mean	55.4	68.3	0.811
Min.	10	40	0.14
Max.	70	74	0.97
Std.	8.8	4.6	0.113

The Following Table compares AV detection of malware families from 1945 malwares.

Table 16 Lists the AV Detection by 3 Malware families.

Statistic		Number Detections	Total Number of AVs	Detection Rate
Ransomware	Mean	55.5	69	0.804
	Std	8.5	2.3	0.120
	Min	10	55	0.14
	Max	69	72	0.97
Spyware	Mean	57.8	68.9	0.838
	Std	7.1	2.3	0.099
	Min	17	54	0.24
	Max	70	73	0.97
Trojan Horse	Mean	53.8	68.4	0.785
	Std	10.2	2.8	0.139
	Min	15	49	0.22
	Max	69	73	0.96

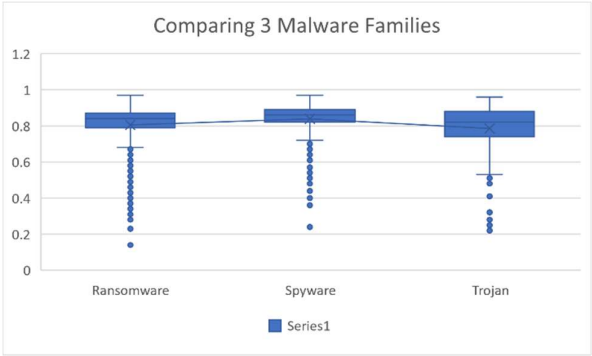

Figure 4 shows visual comparison of 3 Malwares by AV Detection Rates

6.3.2 ML Results on Binary classification with original 55 feature

The following table shows the results of a binary classification intended to discern between benign and malicious software instances within Dataset 3.

Table 17 shows Binary classification results on 55 features of CICMalMem2022.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	1.000	1.000	1.000	100	1.000
J48	0.999	0.999	0.999	99.92	1.000
Naive Bayes	0.991	0.991	0.991	99.14	0.995
SMO	0.998	0.998	0.998	99.83	0.998
IBk	1.000	1.000	1.000	99.98	1.000

6.3.3 ML Results on Binary classification with Refined 6 feature

The following ML test focused on 6 specific features.

Table 18 shows Binary classification results on 6 specific features of Dataset 3.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	1.000	1.000	1.000	99.97	1.000
J48	0.999	0.999	0.999	99.92	1.000
Naive Bayes	0.984	0.984	0.984	98.40	0.994
SMO	0.991	0.991	0.991	99.13	0.991
IBk	1.000	1.000	1.000	99.95	1.000

6.3.4 ML Results on Multi-classification with original 55 feature

The following table shows the results of a multi-classification test designed to categorize diverse types of malwares within Dataset 3.

Table 19 shows multi-classification results on 55 features of CICMalMem2022.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.738	0.737	0.737	73.72	0.892
J48	0.724	0.724	0.724	72.38	0.827
Naive Bayes	0.486	0.390	0.308	39.03	0.601
SMO	0.475	0.398	0.324	39.77	0.605
IBk	0.625	0.624	0.624	62.41	0.719

6.3.5 ML Results on Multi-classification with Refined 6 feature

The following table shows the results of a multi-classification using a reduced set of 6 specific features. The result has some missing data as shown below Table 20.

Table 20 shows Binary classification results on 6 specific features of CICMalMem2022.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.527	0.526	0.527	52.63	0.717
J48	0.516	0.511	0.512	51.10	0.689
Naive Bayes	0.437	0.353	0.250	35.30	0.545
SMO	N/A	0.354	N/A	35.36	0.505
IBk	0.475	0.474	0.474	47.43	0.617

6.3.6 ML Results on Multi-classification [Memory dumps 55 + Behavior]

In this test combined the 14 behavioral features with 55 memory dumps features *(our original research question as mentioned in Section 1.1),* which gave us 69 features in total. As we have 3 malware families, we did the multi-classification on this dataset.

By adding 14 f the accuracy has increased from 73.72% to 81.02%.

Table 21 shows the ML results of 69 features.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.812	0.810	0.810	81.02	0.924
J48	0.797	0.792	0.792	79.19	0.862
Naive Bayes	0.537	0.468	0.430	46.83	0.735
SMO	0.716	0.696	0.691	69.58	0.802
IBk	0.741	0.741	0.741	74.08	0.804

6.3.7 ML Results on Multi-classification [Memory dumps 6 + Behavior]

In this test, we used 6 memory features and 14 behavioral features, the results decreased significantly, from 73.72% to 73.47%.

Table 22 shows the ML result of 20 features.

Classifier	Precision	Recall	F1-Score	Accuracy (%)	ROC-AUC
RF	0.738	0.735	0.736	73.47	0.879
J48	0.734	0.727	0.730	72.74	0.834
Naive Bayes	0.563	0.487	0.448	48.66	0.738
SMO	0.582	0.572	0.516	57.17	0.721
IBk	0.702	0.698	0.700	69.82	0.768

6.3.8 Comparing AV Detection Rates

This section compares the AV detection of 3 datasets by their detection rates.

Table 23 Compares three datasets by their AV detection rates.

	Mean	Min	Max	Std.
Dataset 1	0.759	0.370	0.900	0.078
Dataset 2	0.706	0	0.9	0.185
Dataset 3	0.811	0.14	0.97	0.113

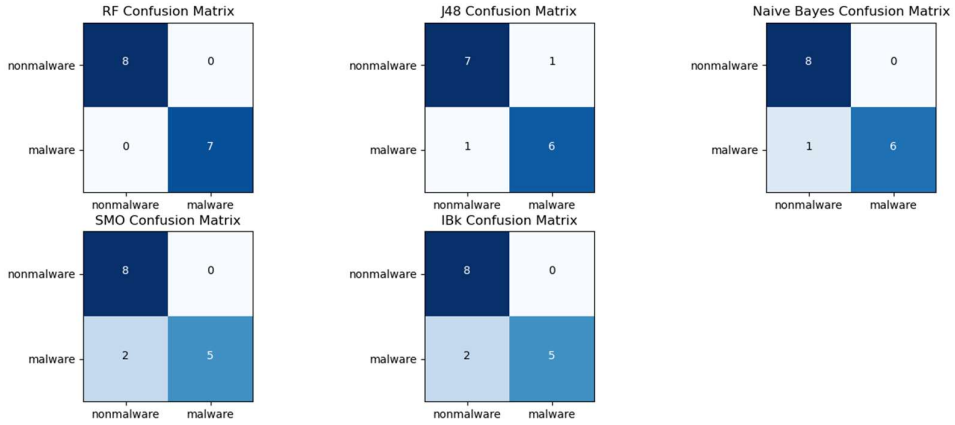

Figure 5 shows a Confusion matrix for experiment of selecting 30 features from Dataset 2, resulting RF 100% accuracy.

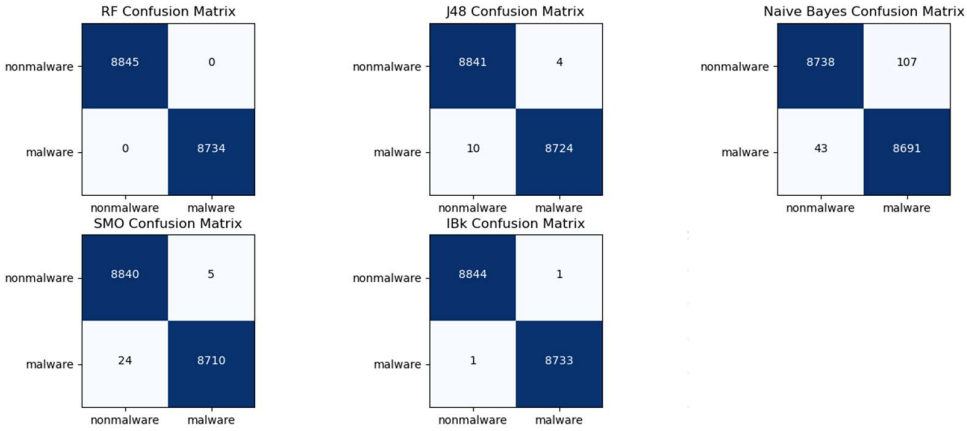

Figure 6 shows a Confusion matrix for experiment of selecting 55 features from Dataset 3, resulting RF 100% accuracy.

6.4 Discussion

Findings and Results:

- ***AV Detection Rates in Different Datasets:*** Across Dataset 1, 2, and 3, we observed varying AV detection rates for malware samples. Dataset 3 demonstrated the highest average detection rate (81.1%), followed by Dataset 1 (75.9%) and Dataset 2 (70.6%).
- ***ML Model Performance:*** In Dataset 2, machine learning models showed promising results with features. Notably, utilizing the original 55 features with behaviors features (section 6.3.6) in Dataset 3, where achieved remarkable accuracy score from 73.72% to 81.02%.

Implications and Significance:

- ***Effectiveness of AV Detection:*** The analysis showcased the capabilities and limitations of AV detection across different malware families. It highlights the necessity of robust and adaptive detection systems to handle diverse threats effectively.
- ***ML Model Potentials:*** The ML models, especially when equipped with a comprehensive set of features, exhibited high accuracy in distinguishing between benign and malicious software. This suggests the potential for leveraging ML techniques in strong malware detection systems.

Limitations and Challenges:

- Dataset Variability: While Dataset 3 demonstrated higher average detection rates, the variability in AV detection across different malware families suggests the need for more diverse and representative datasets to enhance model generalization.

- Feature Selection Impact: The experiments on reduced sets of features (6 critical features) in Dataset 2 and 3 showcased varying performance, indicating the significance of feature selection and its impact on model outcomes.

Moreover, the research question addressed how the combination and separate use of memory and behavior features enhance malware detection accuracy. Our study found that combining memory and behavior features resulted in improved detection accuracy compared to using either feature set alone. This suggests that hybrid detection models that leverage both types of features offer a promising approach to improving malware detection effectiveness.

7 Conclusion

This study investigated the efficacy of machine learning models for detecting malicious software using both memory and behavior features of malware. Across all three datasets, we observed varying detection rates and performance. Dataset 3, with the most comprehensive feature set, demonstrated the highest detection accuracy, ranging from 73.72% to 81.02%. These results highlight the potential of ML models to enhance malware detection effectiveness.

The combination of memory and behavior features proved to be more effective than using either feature set alone. This suggests that hybrid detection models offer a promising approach to countering the evolving landscape of malware.

Moreover, to address the limitations of existing datasets and optimize feature selection techniques, future research should focus on developing data augmentation techniques to expand the size and diversity of malware samples. Research should also explore methods for selecting relevant features that are indicative of malicious behavior, ensuring that the ML models are not biased towards specific patterns or artifacts.

In conclusion, this study provides valuable insights into the effectiveness of ML models for detecting malware using memory and behavior features. The findings suggest that hybrid detection models hold promise for improving malware detection accuracy and adaptability. Future work should focus on addressing data variability, optimizing feature selection, and integrating ML models into practical malware detection systems.

8 Future Study

Our next step is to investigate the stability of these methods when incorporating specific data features or employing optimization techniques. This will involve evaluating the performance of ML models across a broader range of datasets and using different feature selection and optimization techniques. Additionally, we will examine the trade-offs among detection accuracy, computational efficiency, and the inclusion of comprehensive memory features like network, file, registry, and process relationships. These features are currently absent in dataset 2 and dataset 3.

By addressing the limitations of existing datasets, optimizing feature selection techniques, and integrating ML models into practical systems, we can significantly enhance the ability of cybersecurity systems to detect and combat malicious software, safeguarding users against the evolving threat landscape.

References

Abeydeera, W. P. S. (2020). *Fileless Malware Detection in the Cloud Using Machine Learning Techniques*. https://digikogu.taltech.ee/en/Item/87cb2a3a-7ef5-43f0-89a5-ef4cb588b0d5

Al-Qudah, M., Ashi, Z., Alnabhan, M., & Abu Al-Haija, Q. (2023). Effective One-Class Classifier Model for Memory Dump Malware Detection. *Journal of Sensor and Actuator Networks*, *12*(1), 5.

Borana, P., Sihag, V., Choudhary, G., Vardhan, M., & Singh, P. (2021). An assistive tool for fileless malware detection. *2021 World Automation Congress (WAC)*, 21–25.

Botacin, M., Grégio, A., & Alves, M. A. Z. (2020). Near-memory & in-memory detection of fileless malware. *The International Symposium on Memory Systems*, 23–38.

Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, *103*, 102166.

Bucevschi, A. G., Balan, G., & Prelipcean, D. B. (2019). Preventing file-less attacks with machine learning techniques. *2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 248–252.

Carrier, T. (2021). *Detecting obfuscated malware using memory feature engineering*.

Dang, F., Li, Z., Liu, Y., Zhai, E., Chen, Q. A., Xu, T., Chen, Y., & Yang, J. (2019). Understanding fileless attacks on linux-based iot devices with honeycloud. *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 482–493.

Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. *Applied Sciences*, *12*(17), 8604.

Hendler, D., Kels, S., & Rubin, A. (2018). Detecting malicious powershell commands using deep neural networks. *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 187–197.

Kantola, T. (2022). *Exploring VirusTotal for security operations alert triage automation*.

Keyes, D. S., Li, B., Kaur, G., Lashkari, A. H., Gagnon, F., & Massicotte, F. (2021). EntropLyzer: Android malware classification and characterization using entropy analysis of dynamic characteristics. *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS)*, 1–12.

Khalid, O., Ullah, S., Ahmad, T., Saeed, S., Alabbad, D. A., Aslam, M., Buriro, A., & Ahmad, R. (2023). An insight into the machine-learning-based fileless malware detection. *Sensors*, *23*(2), 612.

Khushali, V. (2020). A Review on Fileless Malware Analysis Techniques. *International Journal of Engineering Research & Technology (IJERT)*, *9*(05).

Lee, G., Shim, S., Cho, B., Kim, T., & Kim, K. (2021). Fileless cyberattacks: Analysis and classification. *ETRI Journal*, *43*(2), 332–343.

Louk, M. H. L., & Tama, B. A. (2022). Tree-based classifier ensembles for PE malware analysis: A performance revisit. *Algorithms*, *15*(9), 332.

Mezina, A., & Burget, R. (2022). Obfuscated malware detection using dilated convolutional network. *2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 110–115.

Naeem, H., Dong, S., Falana, O. J., & Ullah, F. (2023). Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification. *Expert Systems with Applications*, *223*, 119952.

Nugraha, A., & Zeniarja, J. (2022). Malware Detection Using Decision Tree Algorithm Based on Memory Features Engineering. *Journal of Applied Intelligent System*, *7*(3), 206–210.

Roy, K. S., Ahmed, T., Udas, P. B., Karim, M. E., & Majumdar, S. (2023). MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. *Intelligent Systems with Applications*, *20*, 200283.

Sihwail, R., Omar, K., Zainol Ariffin, K. A., & Al Afghani, S. (2019). Malware detection approach based on artifacts in memory image and dynamic analysis. *Applied Sciences*, *9*(18), 3680.

Sudhakar, & Kumar, S. (2020). An emerging threat Fileless malware: a survey and research challenges. *Cybersecurity*, *3*(1), 1.

Talukder, M. A., Hasan, K. F., Islam, M. M., Uddin, M. A., Akhter, A., Yousuf, M. A., Alharbi, F., & Moni, M. A. (2023). A dependable hybrid machine learning model for network intrusion detection. *Journal of Information Security and Applications*, *72*, 103405.

Varlioglu, S., Elsayed, N., ElSayed, Z., & Ozer, M. (2022). The dangerous combo: Fileless malware and cryptojacking. *SoutheastCon 2022*, 125–132.

Zhang, S., Hu, C., Wang, L., Mihaljevic, M. J., Xu, S., & Lan, T. (2023). A Malware Detection Approach Based on Deep Learning and Memory Forensics. *Symmetry*, *15*(3), 758.