National College of Ireland

Enhanced Fileless Malware Detection in Memory Using Convolutional Neural Networks

MSc Research Project
Master of Science in Cybersecurity

Ademola Ajayi
Student ID: X23266015

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ademola Temitayo Ajayi
Student ID:	x23266015
Programme:	MSc in Cybersecurity **Year:** 2024-2025
Module:	Research Project
Supervisor:	Michael Pantridge
Submission Due Date:	11th – December – 2025
Project Title:	Enhanced Fileless Malware in Memory Using Convolutional Neural Network
Word Count:	7372 **Page Count** 23

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: ……Ademola Ajayi……

Date: ……10th – December – 2025 ……

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 computer.	□

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

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

Enhanced Fileless Malware Detection in Memory Using Convolutional Neural Networks

Ademola T. Ajayi
X23266015

Abstract

This thesis report looks at why spotting fileless malware's getting tougher as it runs only in computer memory, uses real Windows tools like PowerShell or Windows Management Instrumentation (WMI), plus it slips past regular antivirus by avoiding files. Traditional-based detection miss these hidden attacks because they rely on known patterns and so we need new ways to focus on what programs do and how memory gets used. To fix this gap, the thesis builds a CNN model trained on live-memory artefacts pulled from the CIC-MalMem-2022 data collection and it comprises of information like running processes, open handles, loaded libraries, modules, signs of code injections e.t.c all gathered via Volatility plugins such as pslist, malfind, ldrmodules e.t.c. The data gets cleaned up, scaled evenly, then packed into 8×8 matrices that help the neural network see attack habits based on spatial layout. The CNN's performance is tested thoroughly, compared side by side with four standard classifiers which are Random Forest, Naïve Bayes, RBF-SVM, and XGBoost by using measures like accuracy, precision, recall, F1-score, and ROC-AUC. Results show almost flawless detection for every model; XGBoost hits full marks across the board, while the suggested CNN scores 99.92% in accuracy and 99.98% in ROC-AUC, showing it handles unseen data well and rarely gives false positives. Though ensemble methods edge out CNNs slightly in numbers, CNNs win by automatically building their own features from raw input. That cuts down on manual feature engineering plus makes them more flexible when malware change tactics. The outcomes back up CNN-based memory artefact modelling as a robust foundation for next-generation, in-memory fileless malware detection, opening doors to explore time-aware neural networks and real-world endpoint telemetry.

1 Introduction

Malware is code or malicious software that harms the information systems it runs on. Its damage can have terrible effects such as erasing data on a personal computer or triggering a nuclear power plant to fail and even reputational damage. Information devices must therefore be protected from malware attacks (Tekerek & Yapici, 2022). The shortcomings of traditional threat protection systems, which often rely on static signatures and rule-based analysis, have been made clear by the growing of advanced cyberthreats. Malwares are intentionally created to undermine system availability, confidentiality, or integrity, is still the most common attack vector among these threats in both cloud and enterprise environments (Jabra et al., 2023). Antivirus (AV) software can identify traditional malware by comparing it to an already known binary patterns in executable files that are saved on disk. In other to avoid leaving forensic evidence, threat actors started using more sneaky tactics that take advantage of memory and legitimate system components as detection techniques advanced (Dewan & Sivakumar, 2024).

File-based malware exemplifies the traditional attack paradigm. This malware infiltrates systems by using portable executable files to install harmful payloads on computer disk. File-based threat detection methods sometimes encompass signature-based scanning (known signature as contained in a database), heuristic analysis, and sandbox execution (Eaton, 2025). Although, it is proficient in identifying an established malware families, these methods often struggles in detecting novel or obfuscated versions because the threat actors often utilize tactics such as encryption, packing, and polymorphism to avoid being detected.

Fileless malware on the other hand only operates in volatile memory (RAM), so there are no persistent file artefacts for traditional AV scanners to look at. It commonly uses a tactic called living-off-the-land (LOTL) to take advantage of legitimate and trusted windows system utilities like PowerShell, Microsoft Office macros and even WMI. Fileless attacks can easily integrate with normal system functions because these tools are legit system component (Sudhakar & Kumar, 2020). The outcome is a secret attack sequence that can survive and carry out commands without creating files on disk, significantly reducing the effectiveness of traditional antivirus and intrusion detection systems. Notable attacks like Poweliks have demonstrated the efficacy of fileless malware in evading endpoint security (LaPorte, 2024).

1.1 The challenge of In-Memory Detection

Identifying fileless malware also necessitates periodic monitoring of system memory, where the code is stored and executed. Inspecting volatile memory in real time is challenging due to its ephemeral characteristics and substantial data volume. Traditional forensic tools such as Volatility or Rekall can examine memory dumps offline; however, they are inadequate for continuous or real-time detection (Kara, 2022). This delay enables adversaries to finalize data exfiltration or privilege escalation prior to detection, highlighting the necessity for early-stage, memory-centric detection frameworks.

1.2 Deep Learning and Pattern Recognition

In recent times, researchers in the field of cybersecurity has increasingly turned to machine learning (ML) and deep learning (DL) to address the limitations of traditional malware detection tools whereas unlike signature-based approaches, these models learn patterns directly from the data, allowing them to recognise suspicious behaviour even when no predefined rules exist. Among the various DL approaches, CNNs have attracted a lot of interest due to its ability to pick up on complex patterns and spotting structures in data (Tekerek & Yapici, 2022). In everyday cybersecurity practice, this mean that CNN can support malware detection systems by helping them distinguish normal and real activity from potentially harmful behaviour. This is achieved by examining things like byte sequences, raw executable files, or even visual snapshots taken from different areas of system memory.

When it comes to spotting threats that live entirely in memory, CNNs can be surprisingly effective. They can treat memory dumps or entropy maps like regular 2-D images, where each “pixel” mirrors a byte value or a bit of entropy. By looking at them this way, the network starts to pick up on certain spatial habits linked to things like malicious code injections, process hollowing, or even obfuscation tricks. This pattern-spotting ability helps CNNs notice subtle anomalies that more traditional, hand-crafted feature techniques often miss.

Across the industry, the shift is slowly moving toward behaviour- and memory-focused detection. Industry vendors such as CrowdStrike Falcon and Microsoft Defender for Endpoint make heavy use of behavioural logs, AMSI inspection, and real-time memory scanning to handle fileless attacks (Microsoft, 2024) (CrowdStrike, 2022). The catch, though, is that these enterprise tools are mostly closed off and rely on rules or limited AI components, which leaves plenty of space for academic research to push more open, explainable, and adaptable CNN-based approaches.

1.3 Research Objectives

To design and implement a Convolutional Neural Network (CNN)-based detection framework that leverages memory artefacts from dataset to extract discriminative spatial behavioural patterns indicative of fileless malware execution in Windows environments.

To evaluate and benchmark the performance of the proposed CNN model against conventional machine-learning classifiers (Random Forest, Naïve Bayes, Support Vector Machine, and XGBoost) using key metrics such as accuracy, precision, recall, F1-score, false-positive rate, and ROC-AUC to assess its effectiveness and generalization capability

1.4 Research Questions

Research question 1: How can CNN be model in such a way to use memory dumps for identifying patterns to indicate the presence fileless malware activity in Windows environments?

Research question 2: How does the performance of the proposed CNN based detection framework be compared with traditional classifiers such as Random Forest, Naïve Bayes, Support Vector Machine, and XGBoost when evaluated using metrics such as accuracy, precision, recall, F1-score, false-positive rate, and ROC-AUC?

1.5 Research Structure

This document is organized into seven chapters, each addressing a core element of the research process and contributing to the development and evaluation of a CNN based framework for detecting in-memory fileless malware. The introduction section provides background information, problem statement and research objectives. The literature review section explores existing works related to memory forensic, deep learning algorithms and industry approaches to fileless malware detection. The chapter three emphasizes on the methodology for implementing the proposed framework which is followed by the design chapter which includes the training of the dataset, experimentations and evaluating the CNN based framework against the benchmarked machine learning algorithms. Ethical considerations was taken into account when sourcing for dataset and the acquired dataset was publicly available and accessible.

2 Related Works

This literature review looks at what researchers and industry have done so far to tackle fileless malware and where the gaps still are. It goes through key studies on in-memory malware analysis, paying attention to the datasets they used, the techniques they tried, and the results they reported. It also takes a closer look at how deep learning has been applied in this space, pointing out where current approaches fall short and where there's room to push the research further, especially when it comes to spotting fileless attacks that hide in system memory.

2.1 Detection Techniques and Emerging Challenges in Fileless Malware

(Achmad et al., 2025) recently explored both supervised and unsupervised machine learning models for detecting malware by analysing Sysmon event logs in real time. They used Principal

Component Analysis (PCA) to shrink the feature set without losing the behavioural signals needed for detection, which helped simplify the dataset while keeping the important parts intact. In their results, Naïve Bayes managed to catch most malicious events (high recall) but struggled with precision because the data was imbalanced. In contrast, supervised models like Random Forest and Decision Trees performed much more consistently, with Random Forest achieving the strongest F1-score overall. The unsupervised side of the study was just as interesting models such as Local Outlier Factor (LOF) and One-Class SVM were surprisingly good at spotting unusual patterns in Sysmon logs, with LOF coming very close to perfect performance. Altogether, the findings highlight how valuable low-level system telemetry is for uncovering subtle malicious behaviour. That said, the authors also noted a key limitation: Windows events don't always link parent and child processes cleanly. Because of that, they suggested that future work should look into combining multiple log sources or improving context-building techniques to close this gap.

(Sudhakar & Kumar, 2020) emphasizes the need for better fileless malware detection and prevention. Threat actors are increasingly using PowerShell and Windows Management Instrumentation (WMI) to run malicious actions, bypass signature-based detection systems, and persist or exfiltrate data. Their stealth nature renders traditional malware detection methods less effective. The rise in non-malware or living-off-the-land (LotL) attacks, such as those targeting SWIFT and the Ukrainian power grid, highlights the need for innovative, behaviour-based, and memory-centric detection frameworks to protect critical infrastructure and global financial systems.

(Borana et al., 2021) explain how fileless malware hides entirely in system memory, which makes detection and stopping it a lot tougher. In their work, they built a modular, Java-based system that watches system processes and network activity in real time, flagging anything that looks out of the ordinary. With fileless attacks becoming easier to create due to open-source tools, the authors stress that runtime behavioural analysis is now one of the most important lines of defence. They also point out that techniques like machine learning, data mining, and even AI-inspired immune system models can help detect and react to these stealthy, memory-resident threats far more effectively.

(Khushali, 2020) explained the existing spotting and mitigation techniques for fileless malware, clarifying common misconceptions regarding its operational mechanisms and emphasizing the need for behavioural and memory-based detection approaches to effectively identify and counter these stealthy threats.

2.2 Types of Fileless Malware

Fileless malware can be classified based on the method of delivery and execution of malicious code, which occurs fully within volatile memory, resulting in minimum artefacts on disk.

- Script-based fileless malware: This category utilizes native scripting engines like PowerShell, Windows Management Instrumentation (WMI), and VBScript to run payloads directly in memory. Threats like Poweliks and Kovter sustain persistence using encoded registry entries rather than file dumps (TrendMicro, 2020).
- Exploit-driven fileless malware: These variants target memory-corruption vulnerabilities, such as reflected DLL injection or buffer overflows, to inject malicious

code into legitimate processes such as Cobalt Strike, FIN7, and Meterpreter utilize in-memory shellcode for command-and-control activities (CrowdStrike, 2022).

- Fileless persistence Attack: Some families attain enduring persistence via registry run keys, scheduled tasks, or WMI event subscriptions, re-executing harmful payloads post-reboot while masquerading as legitimate administration operations (Cisco Talos, 2019).
- Registry-based: Persistence without files is commonly achieved by fileless malware using the Windows Registry, a configuration database. Registry keys with malicious code can be executed by legitimate Windows processes (Anyrun, 2025)

2.3 Industry Approach to Fileless Malware Detection

(Microsoft, 2025) explain that fileless malwares are highly polymorphic in nature, changing so quickly that traditional signature-based tools can't keep up. To counter this, endpoint protection like Microsoft Defender relies on behavioural blocking, AI, and machine learning rather than fixed rules. Also, its AMSI engine inspects scripts including PowerShell, VBA, and JavaScript, using functions like AmsiScanString and AmsiScanBuffer to catch obfuscated malicious actions. By combining behaviour monitoring, memory scanning, and cloud-based ML, Defender is able to identify and contain in-memory threats more effectively.

(CrowdStrike, 2022) collaborating with Intel introduced Intel Threat Detection Technology (TDT) to boost Falcon's ability to scan memory for malicious activity. So, instead of relying on file signatures alone, it looks for indicators of attack (IOAs) and behavioural patterns that suggest something suspicious is happening. One key advantage is that TDT provides strong visibility into in-memory attacks with very little performance impact, though it does require Intel-based hardware to get the full benefit. During a memory scan, Falcon identifies a process of interest and walks through its memory space, looking for artefacts such as shellcode patterns or unusual strings that signal an attack.

2.4 Deep learning approach to Fileless Malware Detection

(Demmese et al., 2023) introduced an image-based visualization method using CNNs which improved the field of fileless malware detection. This was achieved yy converting Cobalt Strike beacon payloads into grayscale images for CNN-based classification, this study targeted in-memory and evasive threats, in contrast to previous studies that mostly concentrated on file-based malware. The suggested model's excellent detection accuracy shows that fileless malware traces in network traffic can be successfully found using visual pattern recognition. Also, the study explains how feature extraction and classifier performance might be enhanced in subsequent research by using various conversion methods and picture enhancement, such as bitmap or RGB-based 3D representations.

(Kareegalan et al., 2025) addressed the growing challenge of fileless malware and the limitations of traditional detection techniques. Their study compared Bi-Directional Long Short-Term Memory (BLSTM) models with a proposed Convolutional Long Short-Term Memory (ConvLSTM) architecture for dynamic malware analysis. While BLSTM improved sequential learning, it was computationally intensive and lacked spatial–temporal feature integration. The ConvLSTM model incorporated convolutional layers to enhance feature

extraction and parameter sharing, achieving 98% accuracy compared to BLSTM's 90%, with reduced processing time and loss rates. The findings demonstrate that ConvLSTM offers a more efficient and accurate approach for detecting fileless malware, reinforcing its potential for real-time endpoint protection.

(Kushwaha & Bhati, 2025) presents an adaptable system for identifying fileless and polymorphic malware by combining memory forensics, computer vision, and machine learning techniques. The system attained a classification accuracy of 97.49% by transforming volatile memory dumps into RGB visual representations using a Linear SVM, enhancing baseline performance by 4.23%. A feature fusion strategy that integrates GIST and HOG descriptors effectively captured both global and local memory artifact patterns, resulting in a 3.16% improvement compared to single-feature methods. A significant contribution is the utilization of Uniform Manifold Approximation and Projection (UMAP) for zero-day detection, enhancing performance by 36.98% and increasing benign precision from 0.24 to 0.71. The framework exhibited robust generalization across intricate malware families like Kovter, Poweliks, FIN7, Emotet, and Trickbot. The method proficiently identifies evasive in-memory threats by examining runtime behaviour instead of static signatures, so laying the groundwork for real-time, memory-based malware detection and prospective adaptive learning improvements.

(Sherazi & Qureshi, 2025) presents a hybrid analytical methodology for identifying fileless malware by integrating static and dynamic methodologies to improve detection precision and behavioural understanding. The research demonstrates how fileless malware operates solely in memory, circumventing conventional file-based defences, through the analysis of both real-world and custom-developed samples. This methodology alleviates the limitations of single-method analyses, enhancing detection reliability and interpretability. Future endeavours will incorporate artificial intelligence and machine learning for automatic categorization, confirmed via cross-validation and benchmarked against tools like VirusTotal and Cuckoo Sandbox, with intentions to expand detection capabilities to Linux computers to address rising cross-platform threats.

2.5 Related Work at Glance

Authors	**Major Contribution**	**Key Findings**
Demmese et al. (2023b)	Image-based visualization method using Convolutional Neural Networks (CNNs)	ConvLSTM improves threat detection by capturing spatial-temporal features.
Crowdstrike Falcon (2022)	It uses indicators of attack (IOA) and behavioural analytics to detect suspicious activity	Memory scanning enables new level of visibility and protection.
Microsoft (2025)	It uses anti-malware scan interface (AMSI), behavioural monitoring, memory scanning etc to inspect scripts	N/A
Borana et al. (2021)	A modular, Java-based detection system developed in their study detects anomalous process behaviour	An assertive tool for forensic examiners which assists in

	and suspicious network activities in real time	identifying suspicious process, network and system activities.
Sudhakar & Kumar (2020)	It highlights the need for innovative, behaviour-based, and memory-centric detection frameworks to protect critical infrastructure	Identifying gaps and challenges in current detection and incident-response approach.
Achmad et al. (2025)	It used both supervised and unsupervised machine learning models to analyze malware dynamically utilizing Sysmon event data	Local Outlier Factor (LOF) outperformed supervised classifiers in detecting malicious behaviour
Khushali (2020)	The study carried out a survey on fileless malware lifecycle, detection and mitigation. Also clarifying common misconceptions regarding fileless malware, its operations and mechanism.	Multiple detection techniques are required to counter fileless malware due to its stealth and lack of file system artefacts
Kareegalan et al. (2025)	The paper compared Bi-Directional Long Short-Term Memory (BLSTM) models with a proposed Convolutional Long Short-Term Memory (ConvLSTM) architecture for dynamic malware analysis	The proposed ConvLSTM architecture outperforms BLSTM. It means ConvLSTM is more efficient model for real-time fileless malware detection
Sherazi and Qureshi (2025)	The paper present hybrid analysis framework for detecting fileless malware by combining static and dynamic techniques to enhance detection accuracy and behavioural insight.	The proposed hybrid approach reduces the limitations and blind spots of a single method analyses, resulting in more reliable and detection.

Table 1: Literature review summary

3 Research Methodology

This study employs an experimental methodology to create and assess a framework based on Convolutional Neural Networks (CNN) for the detection of fileless malware utilizing memory-resident artefacts. The research utilizes supervised machine learning methods by employing labelled data from the dataset to train, validate, and evaluate the model (Canadian Institute for Cybersecurity, 2022). So in this setup, the independent variables are the type of input representation used, the specific CNN architecture, and the technique chosen to handle class imbalance. The dependent variables are the usual detection metrics: accuracy, precision, recall, F1-score, and the area under the ROC curve (ROC-AUC).

Using an experimental methodology makes it possible to test, in a measurable way, how well the CNN can separate malicious memory patterns from benign ones while keeping false positives low. So, in other to understand how much value the CNN actually adds, the study also includes baseline models built with more traditional machine learning algorithms, such as Random Forest (RF) and Support Vector Machine (SVM), and compares their performance against the proposed CNN.

3.1 Data Selection

This study uses the CIC-MalMem-2022 dataset which is publicly available and accessible via Kaggle (Godoy, 2022), benchmark developed by the Canadian Institute for Cybersecurity (CIC). The dataset comprises labelled samples of benign malicious memory dumps obtained from Windows systems during the controlled execution of diverse malware families. The dataset's significance arises from its focus on volatile memory, where fileless malware operates without writing payloads to disk, thus offering a realistic basis for detection studies. Each record encapsulates dynamic system behaviour and memory artefacts, which are pre-processed into numerical representations appropriate for CNN-based learning.

Category	**Percentage (%)**	**Count**
Malware	50	29298
Benign	50	29298

Table 2: shows data distribution of CIC Malware data

Malware Category	**Malware Families**	**Percentage Representation (%)**	**Count**
Spyware	180Solutions	6.83	2000
	Coolwebsearch	6.83	2000
	Gator	7.51	2000
	Transponder	8.23	2410
	TIBS	4.81	1410
Ransomware	Conti	6.79	2000
	MAZE	6.68	1950
	Pysa	5.86	1710
	Ako	6.83	2000
	Shade	7.26	2200
Trojan Horse	Zeus	6.66	1950
	Emotet	6.71	1960
	Refroso	6.83	2000
	Scar	6.83	2000
	Reconyc	5.36	1570

Table 3: shows the malware categories and their families

3.2 Data Preprocessing

Before model training, comprehensive preprocessing is performed to guarantee data integrity, consistency, and appropriateness for deep learning.

Feature Category	**Count**	**Item Lists**
Process and Thread (pslist)	5	Pslist.nproc, pslist.nppid, pslist.avg_threads, pslist.nprocs64bit and pslist.avg_handlers
DLL-related (dll-list)	2	Dlllist.ndlls and dlllist.avg_dlls_per_proc

Handle-based (handles)	15	handles_nhandles,handles_avg_handles_per_proc,handles_nport, handles_nfile,handles_nevent,handles_ndesktop,handles_nkey, handles_nthread,handles_ndirectory,handles_nsemaphore,handles_ntimer,handles_nsection,handles_nmutant
Loaded Modules (ldrmodules)	6	ldrmodules_not_in_load,ldrmodules_not_in_init,ldrmodules_not_in_mem,ldrmodules_not_in_load_avg,ldrmodules_not_in_init_avg, ldrmodules_not_in_mem_avg
Memory Injection (malfind)	4	malfind_ninjections,malfind_commitCharge,malfind_protection,malfind_uniqueInjections
Process cross-view validation (psxview)	14	psxview_not_in_pslist,psxview_not_in_eprocess_pool,psxview_not_in_ethread_pool,psxview_not_in_pspcid_list, psxview_not_in_csrss_handles,psxview_not_in_session,psxview_not_in_deskthrd,psxview_not_in_pslist_false_avg, psxview_not_in_eprocess_pool_false_avg,psxview_not_in_ethread_pool_false_avg,psxview_not_in_pspcid_list_false_avg, psxview_not_in_csrss_handles_false_avg,psxview_not_in_session_false_avg,psxview_not_in_deskthrd_false_avg
Module summary (modules)	1	modules_nmodules
Service Enumeration (svcscan)	8	svcscan_nservices,svcscan_kernel_drivers,svcscan_fs_drivers,svcscan_process_services,svcscan_shared_process_services,svcscan_interactive_process_services,svcscan_nactive
Callback Analysis (Callbacks)	3	callbacks_ncallbacks,callbacks_nanonymous,callbacks_ngeneric

Table 4: shows memory-based features extracted from the CIC-MalMem-2022 dataset

- Process and Thread Features: Examples such as pslist is a volatily plugin which lists processes by traversing over the active process list linked within each proc structure (Ligh et al., 2014). However, advanced malware can hide from pslist, so cross view validation is often needed.
- DLL Features: The DLLList is a volatility plugin which was designed to enumerate DLLs by walking the load order list. It describes the number of dynamic-link libraries loaded by processes [pg 233]. This is very common in reflective DLL injection where a malicious process writes a DLL into the memory space
- Handle Features: Handle features provide deep insight into the types of operating system objects accessed by processes, such as files, ports, threads, keys etc.
- Loaded Modules Features (Ldrmodules): This plugin automatically cross-references libraries identified within the kernel's list of per-process memory mappings and the doubly linked list of libraries maintained by the dynamic linker. It can be used to detect libraries injected solely into memory [pg 708].
- Memory Injection (Malfind): This plugin can detect memory injection by looking for process memory mappings that are not associated with files and have specific attributes. They are designed to hunt down remote code injection [pg 253-258]
- Process Cross-View Validation (Psxview): This plugin identifies processes by utilizing an alternative process listing that cross-references several information sources to uncover malicious discrepancies.
- Service Enumeration (Svcscan): This plugin utilizes a linked list while also executing a brute-force scan of all memory allocated to services.exe [pg 352-353]

3.2.1 Data Preprocessing

The raw data collection acquired from CIC-MalMem-2022 necessitated structured preprocessing to guarantee consistency, compatibility, and appropriateness for model training. Data preparation is an essential phase in the machine-learning pipeline, as it mitigates noise, standardizes data scales, and readies features for optimal learning efficacy (Han et al., 2012).

3.2.1.1 Label Encoding

The original dataset included a categorical attribute, Label, denoting the classification of each memory sample as either Benign or Malware. Due to the necessity for numerical input in ML and DL algorithms, categorical values were encoded as binary integers, with 0 assigned to Benign samples and 1 to Malware samples. The encoding procedure enhances computational efficiency during training, as binary labels correspond with the binary cross-entropy loss function employed by the CNN.

$$Label_{encoded} = \begin{cases} 0, & \text{if Label} = \text{Benign} \\ 1, & \text{if Label} = \text{Malware} \end{cases}$$

Figure 1: Label encoding representation

3.2.1.2 Data Normalization

The CIC-MalMem-2022 dataset has 55 numerical forensic attributes, including process counts, handling actions, and module injections, which exhibit considerable variation in magnitude. For instance, handles_nhandles may reach into the thousands, whereas malfind_ninjections could be limited to single-digit figures. Disparities in scale can skew model training, since characteristics with greater magnitudes overshadow the learning process (Goodfellow, Bengio, and Courville, 2016).

To address this, Min–Max Normalization was employed to convert all characteristics to a consistent range between 0 and 1, as shown in figure 2:

$$x^1 = \frac{\mathrm{x} - x_{min}}{x_{\max} - x_{min}}$$

This method maintains the relative relationships among features while enhancing the convergence rate and numerical stability of the CNN. Normalization guarantees that all input variables contribute equally to the classification decision, which is crucial when integrating multi-scale forensic indications.

3.2.2 Feature Representation

This research converts volatile memory artefacts into organized inputs appropriate for CNNs. In contrast to usual byte-plot or entropy-based visualization techniques that transform raw binaries into grayscale or entropy images, the proposed method uses a ready-made forensic characteristics derived from the CIC-MalMem-2022 dataset.

Each record encapsulates process level statistics such as injected threads, handle counts, DLL loads, and code injection metrics that collectively reflect the in-memory behaviour of both

benign and malicious processes. All these features are normalized with “Min–Max” scaling to ensure uniform numeric ranges and subsequently restructured into an 8 × 8 matrix to retain inter-feature interactions. This spatial mapping allows the CNN to recognize distinctive behavioural patterns instead of depending on low-level byte textures.

3.2.3 Data Splitting

The dataset is divided into training (70%), validation (15%), and testing (15%) subsets by stratified sampling to maintain class distribution. Organizing by process or capture ID guarantees that samples from the identical execution trace are confined to a single subset, thus averting data leakage.

3.3 Data Mining and Machine Learning Classifiers

This study's data mining section explains the utilization of ML and DL techniques to identify fileless malware from memory-based artefacts. Data mining, in this sense, means the automatic extraction of concealed patterns and relationships from pre-processed forensic data that differentiate benign from malicious system behaviour (Han et al., 2012). A collection of supervised classifiers is employed to extract distinguishing behavioural traits from the CIC-MalMem-2022 dataset. The postulated Convolutional Neural Network (CNN) model is evaluated against four conventional machine-learning algorithms: Random Forest (RF), Naïve Bayes (NB), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGBoost) to provide a comparative performance baseline.

3.3.1 Random Forest

Random Forest (RF) has been extensively utilized in malware and intrusion detection studies owing to its resilience and interpretability (Souri & Hosseini, 2018). In the field of fileless malware detection, RF can detect threshold-based correlations among forensic attributes such as the quantity of injected modules or process handles yet it is constrained in its ability to capture sequential or geographical connections among related data.

3.3.2 Naïve Bayes (NB)

This study uses the NB method as a minimal benchmark for probabilistic reasoning. Nevertheless, the interdependence of features in memory forensics such as thread counts, DLL loads, and handle counts that co-vary during malicious execution may compromise the veracity of the independence assumption. Empirical research on cyber-threat categorization indicates that Naive Bayes generally exhibits inferior performance compared to ensemble and deep learning models when feature correlations are elevated (Alzubaidi et al., 2021). Nonetheless, its incorporation provides significant comparative insight into the way probabilistic models manage organized, high-dimensional forensic data.

3.3.3 Support Vector Machine (SVM)

This research employs SVM on the normalized CIC-MalMem-2022 feature vectors to model non-linear correlations between process-level and kernel-level properties. The model's dependence on global margins instead of local spatial patterns limits its capacity to understand intricate feature interactions within the 8×8 heatmap representations. Consequently, although SVM functions as a robust classical benchmark, CNNs are anticipated to excel by independently acquiring hierarchical spatial filters that generalize more effectively across hitherto unencountered malware varieties.

3.3.4 Extreme Gradient Boosting (XGBoost)

XGBoost is a scalable version of gradient-boosted decision trees that incrementally integrates weak learners to reduce classification loss (Chen & Guestrin, 2016). The approach incorporates regularization terms to manage model complexity and utilizes second-order gradient information for expedited convergence. XGBoost has exhibited competitive accuracy and robustness to feature noise in malware analysis. Its capacity to capture non-linear connections via additive tree ensembles renders it appropriate for diverse feature sets, such as those obtained from memory forensics.

3.3.5 Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) serves as the principal model established for this research. Convolutional Neural Networks (CNNs) extract spatial features using convolutional filters that detect localized feature patterns within input matrices (Ersavas et al., 2024). Memory artefacts in the CIC-MalMem-2022 dataset are transformed into two-dimensional feature maps (e.g., 8×8) and input into the CNN, allowing the network to discern high-level spatial connections among process-level behaviours, manage anomalies, and identify injection signs.

3.3.6 Model Comparison and Evaluation

All classifiers (RF, NB, SVM, XGBoost, and CNN) are trained and validated on the identical pre-processed dataset under uniform experimental setup. The comparison uses quantitative criteria such as Accuracy, Precision, Recall, F1-score, and ROC-AUC (Area under curve), in conjunction with qualitative interpretability through Grad-CAM for CNN-based visualizations. This multi-model methodology guarantees a thorough evaluation of each model's ability to identify fileless malware from volatile memory artefacts. The mathematical formulas for these quantitative criteria are as follows

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

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\mathrm{F1\ Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$AUC = \sum_{i=1}^{n-1} (\mathrm{FP}_{i+1} - FP_i) \times \frac{TP_{i+1} + TP_i}{2}$$

Where:
TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

3.4 Ethical Consideration

The entire testing process utilizes publicly accessible, sanitized datasets devoid of active harmful binaries. No actual malware execution occurs. The dataset is securely maintained on disks, with access limited to authorized researchers alone. This research complies with ethical standards for the secure management of cybersecurity datasets, data protection, and the responsible dissemination of results. Moreover, findings are disclosed with complete transparency, encompassing limits and other sources of bias.

4 Design Specification

This chapter outlines the design specifications of the proposed fileless malware detection architecture

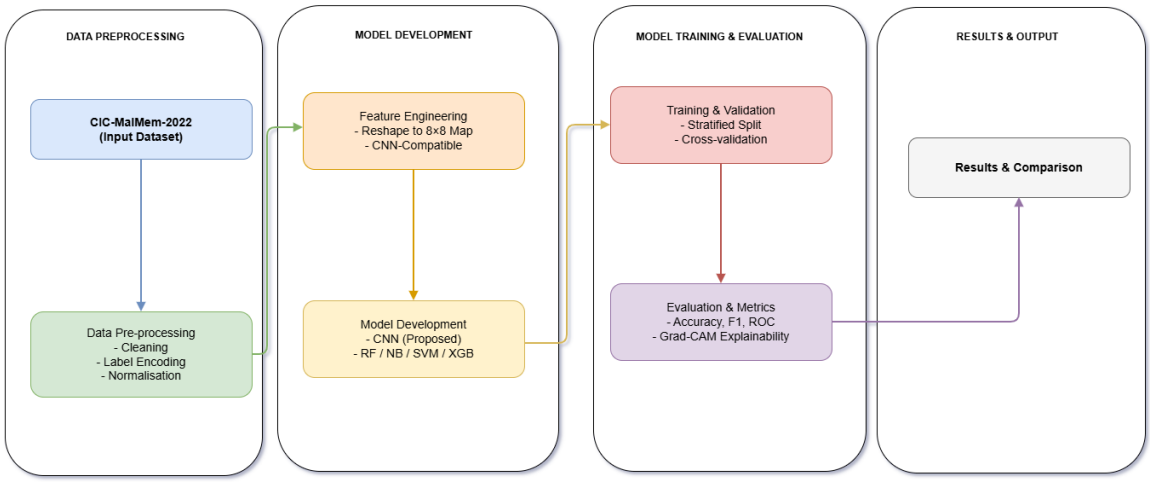

Figure 2: Fileless malware detection Framework using CNN

Module	Function	Input	Output
Data Loader	Imports CIC-MalMem-2022 dataset and extracts relevant attributes.	CSV files	Pandas DataFrame
Pre-Processor	Handles missing data, normalises numerical values, encodes labels.	Raw data	Cleaned dataset
Feature Transformer	Converts each feature vector into an 8×8 matrix (64-dim heatmap).	Normalised data	CNN input tensor
Model Trainer	Trains CNN and baseline ML classifiers using cross-validation.	Feature matrices	Trained models
Evaluator	Computes performance metrics (ACC, Precision, Recall, F1, AUC).	Predictions	Metric tables & plots
Comparator	Compares all models statistically and summarises findings.	All metrics	Comparative analysis

Table 5: Functional design specifications

5 Implementation

This chapter explains the implementation of the proposed CNN framework for fileless malware detection alongside a comparison of ML models baselines such as Random Forest (RF), Naïve Bayes (NB), Support Vector Machine (SVM), and XGBoost. The implementation phase adapts the conceptual design outlined in Chapter 4 into a working, reproducible detection pipeline that can identify malicious memory patterns while reducing false positives.

5.1 Testing Environment

The experimental environment comprised of both hardware and software resources which are optimized for computational efficiency and reproducibility. Experiments were performed on a GPU-enabled workstation featuring an Intel Core i7 processor, 16 GB of RAM, and an Intel ARC GPU operating on Windows 11 Home (64-bit). All models were executed in Python 3.12 utilizing Visual Studio Code (VS Code) as the Integrated Development Environment (IDE).

5.2 Data Preprocessing and Feature Preparation

The implementation used the CIC-MalMem-2022 dataset, which has labelled traces of how benign and fileless malware processes behave. The dataset was loaded from a CSV file and went through a number of preprocessing to get it ready for the model to use.

1. Label Encoding: Using the label encoder from Scikit-Learn, the categorical "Label" attribute (Benign/Malware) was converted to numeric values of 0 and 1.
2. Feature Selection: In order to guarantee compatibility with both deep-learning and machine-learning models, only numerical columns were maintained.
3. Normalization: Continuous features were scaled to the range $[0,1]$using a MinMaxScaler to prevent bias toward attributes with large magnitudes.
4. Data Splitting: In order to maintain class balance, stratified sampling was implemented to partition the dataset. The final proportions were as follows: 70% training, 15% validation, and 15% assessment.

5.3 Training Approach and Hyperparameters

The CNN was trained for 30 epochs with a batch size of 64 using Adam optimizer at a learning rate of 1×10^{-3}. To make sure the model learnt broad patterns rather than memorising the data, the dataset was divided into training (70%), validation (15%), and test (15%) sets.

6 Evaluation

The results were achieved by conducting testing on the CIC-MalMem-2022 dataset in accordance with the data separation strategy delineated in Chapter 5 (70% training, 15% validation, and 15% testing). The goal of this analysis is to assess the efficiency of the CNN in comparison to conventional classifiers in terms of their overall robustness, false positive reduction, and detection accuracy.

6.1 Summary of Performance

The table below shows that all models obtained an exceptional performance, with F1-scores and accuracy exceeding 99%. The XGBoost model obtained perfect classification across all metrics, while the proposed CNN closely followed with an accuracy of 99.92% and a ROC-AUC of 99.98, demonstrating exceptional discriminative ability between benign and malicious samples.

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**	**ROC-AUC (%)**
Random Forest	99.99	99.98	100.00	99.99	100.00
Naïve Bayes	99.22	98.78	99.66	99.22	99.57
SVM (RBF Kernel)	99.74	99.68	99.80	99.74	100.00
XGBoost	100.00	100.00	100.00	100.00	100.00
Proposed CNN	99.92	99.86	99.98	99.92	99.98

Table 6. Comparison of CNN and other baseline models

6.2 Convolutional Neural Network (CNN)

The CNN got 99.92% accuracy, 99.86% precision, and 99.97% recall, which means it learnt strong spatial and statistical feature patterns from the 8×8 feature maps. The CNN's high recall means that it found almost all of the malware, and its high precision means that it didn't make many mistakes (benign processes that were incorrectly tagged as malware). The ROC-AUC of 99.98 shows that the CNN can almost perfectly tell the difference between classes at varied threshold levels.

This finding shows that CNNs work well for recognizing malware behaviour patterns and extracting features from memory. Minor deviations from perfection as compared to XGBoost may be due to a random initialization in deep learning and slight numerical discrepancies in feature scaling. Still, the CNN's capacity to generalize is still very good, and the fact that it learns from end to end means that it doesn't need to do manual feature engineering, which is a big plus in contexts where threats might change quickly.

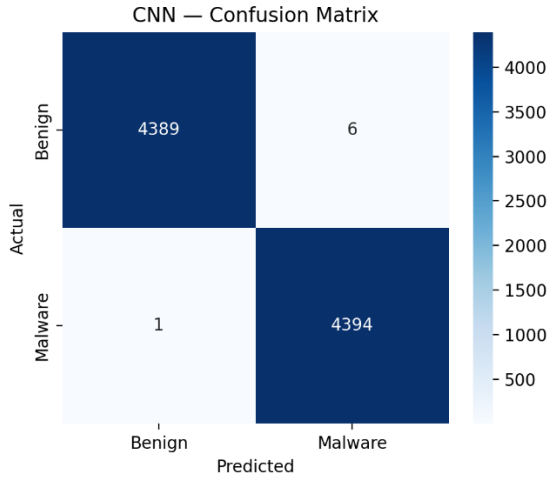

Figure 3: Visualization of CNN confusion matrix

6.3 Random Forest (RF)

Random Forest did almost perfectly, with an accuracy of 99.99% and an AUC of 100.00%. Its ensemble-based design was able to handle high-dimensional data well by averaging several decision trees to lower variance. The model only incorrectly labelled one harmless case as malware, which meant that the false positive rate was practically zero.
This shows that tree-based ensemble approaches are still good starting points for classifying fileless malware, especially when working with structured tabular information.

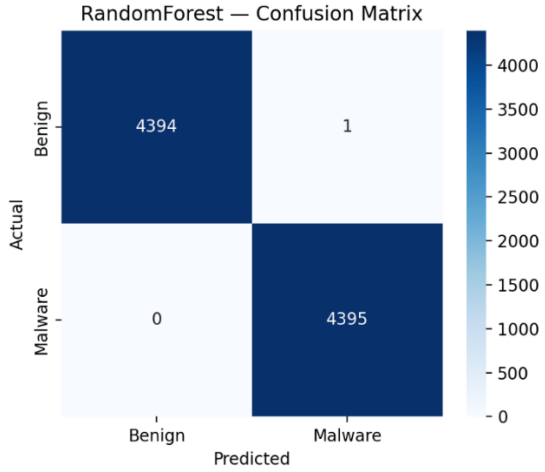

Figure 4: Visualization of Random Forest Confusion Matrix

6.4 Naïve Bayes (NB)

The Naïve Bayes classifier got an accuracy of 99.21% and an AUC of 99.57%. This was still excellent, but lower than other models. This performance difference exists because it assumes that features are independent, which is not the case with malware data when feature correlations are substantial and because of this, NB had a few more false positives and false negatives than the CNN, SVM, and ensemble models. Still, it is light on resources and works well, which makes it good for systems that are limited in space or are built into other devices.

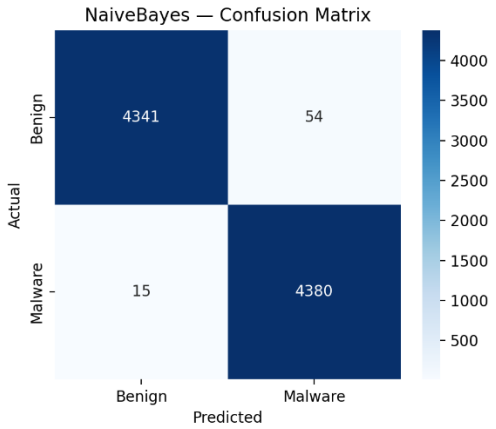

Figure 5: Visualization of Naïve Bayes Confusion Matrix

6.5 Support Vector Machine (SVM -RBF Kernel)

The SVM classifier got an accuracy of 99.74%, a precision of 99.68%, and an AUC of 99.99%, which shows that it can generalize and separate quite well. The SVM was able to simulate non-linear boundaries well since it used the RBF kernel. This made it able to handle complicated decision surfaces in the malware feature space. The AUC value is very close to ideal, which means that SVM could almost perfectly rank benign and malicious data. However, it took longer to train than tree-based and neural methods.

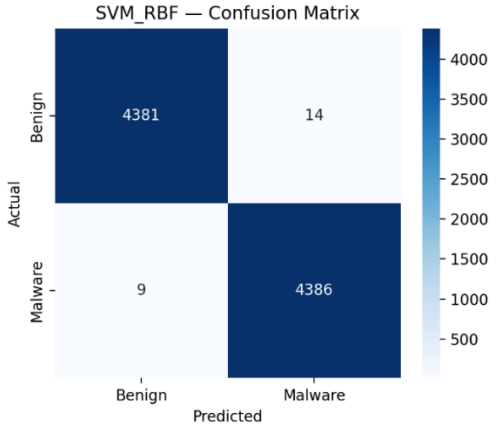

Figure 6: Visualization of Support Vector Machine Confusion Matrix

6.6 Extreme Gradient Boosting (XGBoost)

XGBoost got a perfect score of 100.00 on all performance criteria, which means it made faultless predictions on the test data. This shows that the algorithm is better at selecting features and boosting them. However, getting perfect scores on all metrics can also be a sign of overfitting, especially when the training and testing sets are quite comparable in terms of statistics. Even yet, the model's interpretability and its speed make it one of the best ways to find malware in tabular datasets.

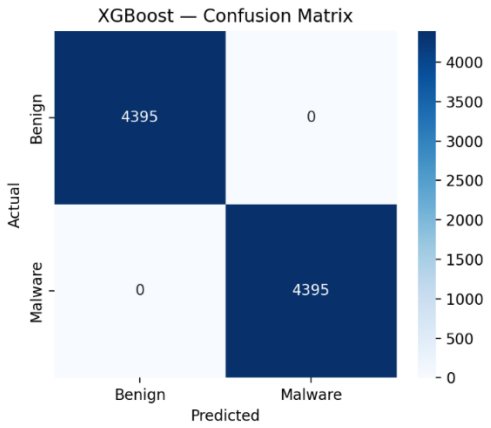

Figure 7: Visualization of XGBoost Confusion Matrix

6.7 Discussion

The findings demonstrate that although traditional ensemble approaches like XGBoost and Random Forest attain the best numerical scores on this dataset, the CNN exhibits more adaptability and sustained relevance for identifying emergent malware behaviours. Tree-based models depend significantly on explicit feature hierarchies, while CNNs can immediately learn new discriminative representations from raw or little pre-processed memory data. As malware developers progressively utilize polymorphism, obfuscation, and fileless execution methods, the capacity of neural networks to generalize from intricate patterns emerges as a critical benefit.

7 Conclusion

This study developed and assessed a CNN-based framework for identifying in-memory and fileless malware using the CIC-MalMem-2022 dataset and contrasted its performance with established machine learning models (Random Forest, Naïve Bayes, SVM-RBF, and XGBoost). The evaluated results indicate nearly flawless detection performance across all models, with the CNN attaining 99.92% accuracy and an AUC of 0.99977, thereby confirming robust generalisation and a very low false alarm rate. Tree-based ensemble methods (Random Forest and XGBoost) achieved the highest numerical scores, with XGBoost attaining ideal metrics, while Support Vector Machines closely trailed; Naïve Bayes, although somewhat less performant, continued to serve as a reliable and efficient baseline. While ensemble models like Random Forest and XGBoost exhibited slightly better numerical performance compared to the CNN, the proposed deep-learning methodology showcased competitive efficacy, providing the substantial benefit of end-to-end feature learning that diminishes dependence on manual feature engineering and enhances adaptability to changing malware behaviours. XGBoost attained flawless classification across all measures, indicating remarkable modelling capability, although it may also imply dataset separability or potential overfitting resulting from statistical similarity between the training and test sets.

Critical forensic plugins like malfind and ldrmodules contribute little features compared to other categories, according to dataset analysis. These features encode high-signal harmful activity signs like memory injection and reflected DLL loading despite their lower dimensionality. This shows that feature quality, not quantity, determines malware detection performance and that locally tailored memory forensics measurements work.

7.1 Future of Work

Future work in this field will concentrate on improving deep learning-based fileless malware detection's realism and usefulness. Even while the suggested CNN performed exceptionally well on structured memory features, real-world attacks show strong temporal behaviour and are constantly evolving. To capture the evolution of malware operation over time, integrating temporal modelling approaches like Long Short-Term Memory (LSTM) networks or hybrid CNN–LSTM architectures is a logical next step. This would allow the model to learn behavioural sequences that distinguish benign activities from covert threats in addition to identifying static memory anomalies. An additional significant focus is on expanding evaluation beyond curated benchmark datasets to encompass real-world memory acquisitions and endpoint telemetry. Utilising the framework on unprocessed RAM dumps, system-level monitoring logs, and endpoint detection and response (EDR) data would enhance external validity and illustrate real-time detection capabilities. This testing would also subject the model to environmental noise, operational intricacies, and various malware variants that are inadequately represented in public datasets.

References

Achmad, R.M., Nariswari, D.P., Pratomo, B.A. & Studiawan, H., 2025. Sysmon event logs for machine learning-based malware detection. *Cyber Security and Applications*, 3, [Accessed 03 November 2025].

Alzubaidi, L. et al., 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(53), [Accessed 2025 November 12].

Anyrun, 2025. *Which type of malware resides only in RAM? Explaining fileless malware*. [Online] Available at: https://any.run/cybersecurity-blog/threat-coverage-digest-november-2025/ [Accessed 01 November 2025].

Borana, P. et al., 2021. An Assistive Tool For Fileless Malware Detection. In *World Automation Congress (WAC)*., 2021. IEEE.

Canadian Institute for Cybersecurity, 2022. *Malware memory analysis (CIC-MalMem-2022)*. [Online] Available at: https://www.unb.ca/cic/datasets/malmem-2022.html [Accessed 11 October 2025].

Chen, T. & Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*., 2016. Association of Computing Machinery.

Cisco Talos, 2019. *Divergent: "Fileless" NodeJS Malware Burrows Deep Within the Host*. [Online] Available at: https://blog.talosintelligence.com/divergent-analysis/ [Accessed 30 September 2025].

CrowdStrike, 2022. *CrowdStrike Falcon Enhances Fileless Attack Detection with Intel Accelerated Memory Scanning Feature*. [Online] Available at: https://www.crowdstrike.com/en-us/blog/falcon-enhances-fileless-attack-detection-with-accelerated-memory-scanning/ [Accessed 27 September 2025].

Demmese, F.A. et al., 2023. Machine learning based fileless malware traffic classification using image visualization. *Cybersecurity*, 6(32), [Accessed 3 November 2025].

Dewan, R. & Sivakumar, V., 2024. A Deep Dive into Detecting and Investigating Fileless Malware. *SSRN Electronic Journal*.

Eaton, J.M., 2025. *NetworkThreatDetection.com*. [Online] Available at: https://networkthreatdetection.com/fieless-malware-detection-challenges/ [Accessed 15 October 2025].

Ersavas, T., Smith, M.A. & Mattick, J.S., 2024. Novel applications of Convolutional Neural Networks in the age of Transformers. *Scientific Reports*, 14(1), [Accessed 24 October 2025].

Godoy, L., 2022. *Malware Memory Analysis / CIC-MalMem-2022*. [Online] Available at: https://www.kaggle.com/datasets/luccagodoy/obfuscated-malware-memory-2022-cic/data [Accessed 12 October 2025].

Han, J., Kamber, M. & Pei, J., 2012. Data Mining Trends and Research Frontiers. In *Data Mining: Concepts and Techniques*. 3rd ed. Elsevier eBooks. pp.585-631.

Jabra, M.B. et al., 2023. Malware Detection Using Deep Learning and CNN Models. In *International Conference on Cyberworlds (CW)*., 2023.

Kara, I., 2022. Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges. *ScienceDirect*, 214, Available at: https://www.sciencedirect.com/science/article/pii/S0957417422021510 [Accessed 20 October 2025].

Kareegalan, K. et al., 2025. Convolutional Long Short-Term Memory for Fileless Malware Detection. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 64(4), [Accessed 14 November 2025].

Khushali, V., 2020. A Review on Fileless Malware Analysis Techniques. *International Journal of Engineering Research & Technology*, 09(05), [Accessed 19 October 2025].

Kushwaha, K. & Bhati, N.S., 2025. Adaptive Detection of Fileless & Polymorphic Malware Using Memory Forensics, Computer Vision, and Machine Learning. In *2025 Global Conference in Emerging Technology (GINOTECH)*. India, 2025. IEEE.

LaPorte, B., 2024. *Morphisec*. [Online] Available at: https://www.morphisec.com/blog/fileless-malware-attacks/ [Accessed 7 September 2025].

Ligh, M.H., Case, A., Levy, J. & Walters, A., 2014. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. 1st ed. John Wiley & Sons.

Microsoft, 2024. *Fileless Threats*. [Online] Available at: https://learn.microsoft.com/en-us/defender-endpoint/malware/fileless-threats [Accessed 28 October 2025].

Microsoft, 2025. *Antimalware Scan Interface (AMSI)*. [Online] Available at: https://learn.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal [Accessed 13 October 2025].

Sherazi, S.N. & Qureshi, A., 2025. Hybrid Analysis Model for Detecting Fileless Malware. *MDPI*, 14(15), [Accessed 22 October 2025].

Souri, A. & Hosseini, R., 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. *Springer*, 8(3), [Accessed 2 November 2025].

Sudhakar, N. & Kumar, S., 2020. An emerging threat Fileless malware: a survey and research challenges. *Springer*, 3(1), [Accessed 15 October 2025].

Tekerek, A. & Yapici, M.M., 2022. A novel malware classification and augmentation model based on convolutional neural network. *Computers & Security*, 112, [Accessed 2 October 2025].

TrendMicro, 2020. *Tracking, Detecting, and Thwarting PowerShell-based Malware and Attacks*. [Online] Available at: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/tracking-detecting-and-thwarting-powershell-based-malware-and-attacks [Accessed 24 October 2025].