Volume 4, Issue III, March, 2025, No. 60, pp. 768-781
Submitted 1/3/2025; Final peer review 22/3/2025
Online Publication 26/3/2025
Available Online at http://www.ijortacs.com

DEVELOPMENT OF AN INTELLIGENT FILELESS MALWARE CLASSIFICATION SYSTEM USING OPTIMIZED DEEP LEARNING TECHNIQUE

Nwafor Anthony C[1*], Mgbeafulike I.J.[2], Okeke O.C.[3]

[1,2,3]Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University, Anambara State.
Email: anthnonynwafor981@gmail.com[1]; ike.mgbeafulike@gmail.com[2]; co.okeke@coou.edu.ng[3]
Corresponding Author's Email and Tel:[1*] anthnonynwafor981@gmail.com; +2348065375935

Abstract

Fileless malware is a significant cybersecurity threat because of its ability to operate without traditional file-based signatures which makes it challenging for conventional security techniques to detect. Hence, this study presents the development of an intelligent fileless malware classification system with the use of deep learning and optimization techniques. The system employs the Behaviour-Driven Development (BDD) methodology which enables precise definition and validation of detection scenarios. Data was collected from primary sources like Cyber-Dome testbed infected with fileless malware across multiple operating systems (Windows, Linux, and Mac), and secondary sources such as Kaggle repositories. Feature engineering was performed using the African Vulture Optimization Algorithm (AVOA) to select the most relevant attributes, enhancing model accuracy while reducing computational complexity. A Deep Neural Network (DNN) classifier was trained on the optimized dataset to detect malicious activity. The system was implemented in Python using TensorFlow and tested as a web-based platform. The software tested indicates that the proposed model significantly improves the detection of complex malware behaviours, providing a robust cybersecurity solution.

Keywords: ***Fileless Malware (FM); Deep Neural Network (DNN); Behaviour-Driven Development (BDD); African Vulture Optimization Algorithm (AVOA); Cybersecurity***

1. INTRODUCTION

The taxonomy of malware refers to the classification and categorization of various types of malicious software based on their characteristics, methods of attack, and impact on target systems. This classification allows cybersecurity professionals to better understand, identify, and mitigate the risks posed by different malware types. One of the most widely recognized categories is ransomware, which encrypts the victim's data or locks them out of their system, demanding a ransom payment for restoring access. This form of malware is often delivered through phishing emails, malicious links, or software vulnerabilities. Another prevalent malware type is the botnet, which involves a network of compromised machines, or "bots," that are controlled remotely by attackers to carry out tasks like distributed denial-of-service (DDoS) attacks, spamming, or data theft (Dang et al., 2019).

Trojans are another common type of malware, often masquerading as legitimate software to deceive users into executing malicious payloads. Once activated, Trojans can open backdoors or steal sensitive information. Spyware, on the other hand, is designed to silently monitor user activity, capturing personal data such as login credentials, browsing habits, or financial information without the user's knowledge. This type of malware is often used for identity theft or espionage (Tekiner et al., 2021).

The infection chain of file-based malware highlights the process through which malicious software is transmitted and executed on a user's system via files (Snow, 2021). These malware types typically rely on files such as executables, documents, or scripts to infiltrate a system, making them a common vector for cyber attacks. Understanding the infection chain is crucial for identifying how file-based malware spreads and how it can be prevented or mitigated. The section explains the typical process by which file-based malware enters a system, executes its payload, and potentially compromises the device, along with the role of antivirus software in detection and protection.

FM operates on the principle of utilizing legitimate system tools and resources to carry out malicious activities, without leaving traditional traces on the disk. Unlike traditional malware, which relies on files to execute its payload, FM runs entirely in the system's memory. By doing so, it evades detection by many conventional security mechanisms that rely on file scanning. This type of malware exploits the inherent trust that operating systems and security software place in legitimate processes such as PowerShell, Windows Management Instrumentation (WMI), or other administrative tools to perform malicious actions without triggering alarms (Atapattu and Jayawardena, 2021).

The core theory behind FM lies in its stealth and persistence. It does not rely on traditional attack vectors, such as downloading or executing files from external sources. Instead, it often exploits vulnerabilities in system software, such as unpatched operating systems or applications, to inject malicious code directly into memory (Mohanta and Saldanha, 2020).Detecting FM presents unique challenges due to its ability to operate entirely in system memory and evade traditional detection methods that rely on file-based analysis. Unlike traditional malware that often leaves traces on the disk, FM resides in volatile memory, leveraging legitimate system tools and processes to execute malicious actions without leaving permanent files behind. As a result, detecting FM requires a combination of advanced techniques and tools tailored to identify unusual patterns of behaviour, memory usage, and system activity. Machine Learning (ML) has become a crucial approach for detecting fileless malware, which poses significant challenges for traditional security tools due to its ability to operate entirely in system memory without leaving traces on disk. Unlike file-based malware that relies on signatures and static characteristics, FM leverages legitimate system processes and tools, making it difficult to detect using conventional methods (Singh et al., 2020).Deep learning, particularly through models like Deep Neural Networks (DNN), has emerged as a powerful technique for detecting FM due to its ability to analyze and learn complex patterns in system behaviour.

Traditional malware detection methods often rely on predefined signatures or static features, making it challenging to detect sophisticated threats like fileless malware, which operates in memory and uses legitimate system processes to execute its payload (Obini et al., 2024). Deep learning models can address this limitation by learning dynamic and high-dimensional features from large datasets, enabling them to recognize subtle anomalies in system activity, such as unusual process behaviour or unauthorized use of system tools like PowerShell or WMI.

Among the notable and recent studies are Obini et al. (2024) who applied deep neural network to develop a machine learning based Fileless malware filter, while Siddiqui et al. (2024) trained several machine learning algorithms to generate model for the detection of malware. However, despite the success of these studies, adaptive model capable of detecting Fileless malware on different operating systems remained a gap. Therefore, this study proposed the design and implementation of dynamic AI-driven network traffic analysis framework for malware detection and prevention, using real-time approach. The contributions of the paper are as follows, the development of new data model for FM considering different computer operating system, then feature engineering approaches will be applied to process the data. DNN carefully designed will e proposed and then trained to generate reliable model for real-time FM detection. Several experiments will be carried out to test the model considering variant operating systems to demonstrate the application diversity of the solution proposed in this work.

2. SYSTEM DEVELOPMENT METHODOLOGY

The methodology used for this work is the Behaviour-Driven Development (BDD) approach. The approach was adopted because it focuses on defining and validating system behaviours in collaboration with stakeholders. Fileless malware is sophisticated and relies on leveraging legitimate system tools, making its detection highly scenario-specific. BDD allows the creation of clear, natural-language scenarios that describe the expected detection outcomes, such as identifying malicious PowerShell activity or unauthorized registry modifications. This ensures alignment between cybersecurity experts, developers, and end-users, while enabling iterative refinement. Furthermore, BDD facilitates robust testing through predefined behaviours, improving the accuracy and reliability of the classifier in detecting complex, real-world fileless malware patterns.

2.1 Data Acquisition

The Data Acquisition Module is responsible for collecting raw malware data from various system logs, memory dumps, registry modifications, and network activity traces. Fileless malware is particularly challenging to detect because it does not leave behind traditional file-based signatures. Therefore, this module focuses on extracting behavioural patterns, process execution logs, and anomalous registry changes that indicate potential malware activity. It gathers both normal and adversarial malware datasets, ensuring that the classification model is trained on a diverse range of samples. The pre-processed structured dataset generated by

this module is then forwarded to the feature engineering stage for further processing.

Data of FM used in this work will be collected from cyber-dome as the primary data source. The testbed FM infected computer will consider operating systems such as windows (version 2007), Linux (Ubuntu 20.04) and Mac (version, 10.7, 2011). The sample size of the data collected is 5026 features of fileless malware. The secondary source of data collection is Kaggle repository, considering open-source FM dataset. The sample size of data collected is 13,743 features of fileless malware.

2.2 Feature Engineering

The Feature Engineering Module plays a critical role in refining the raw data obtained from the acquisition module. It applies three main techniques: feature selection, feature transformation, and feature extraction. Feature selection is particularly important in this context, as it involves filtering out redundant or irrelevant features using the African Vulture Optimization Algorithm. This optimization algorithm ensures that only the most significant features are retained, thereby improving the model's accuracy and reducing computational complexity. Feature transformation normalizes and encodes data into a format suitable for machine learning, while feature extraction identifies key behavioural attributes of fileless malware. The final output of this module is an optimized feature set, ready for classification.

2.3 The System Submenus

The section discussed the sub-system of the fileless malware detection system, showcasing the key sub-systems that contribute to the detection and classification of fileless malware as n Figure 1. This system is designed to handle various stages of data processing, feature engineering, deep learning model training, and evaluation to ensure accurate detection. The workflow begins from a Main Menu, which provides access to the different sub-systems: Dataset, Feature Engineering, Deep Neural Network, and the Detection Model. Each of these sub-systems plays a critical role in improving the reliability of the malware detection process.

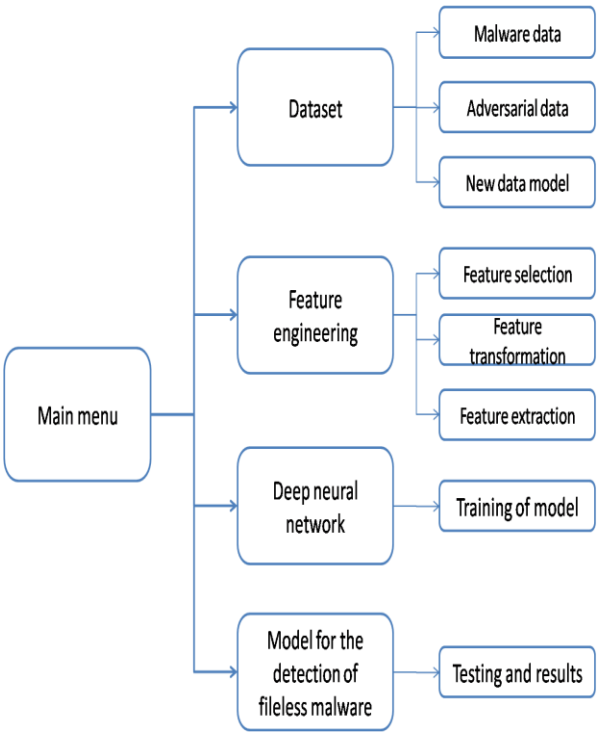

Figure 1: System Sub-menu

The dataset sub-system is responsible for gathering and managing the data required for training the malware detection model. This includes three key components: Malware Data, Adversarial Data, and New Data Model. Malware data consists of real-world examples of fileless malware collected from various sources, including infected systems and cybersecurity research databases. Adversarial data is designed to test the robustness of the detection model against sophisticated malware

that attempts to evade detection. Finally, the new data model represents an evolving dataset that is continuously updated with new malware signatures and behaviours to keep the detection system adaptive and effective against emerging threats.

Feature engineering is a crucial step in preparing the dataset for deep learning analysis. This sub-system comprises three major processes: Feature Selection, Feature Transformation, and Feature Extraction. Feature selection involves identifying the most relevant attributes from the dataset that contribute significantly to malware classification, ensuring that the model focuses on meaningful patterns. Feature transformation converts raw data into a format suitable for deep learning, applying techniques like normalization and encoding to enhance model accuracy. Feature extraction further refines the data by deriving new features that improve the model's ability to distinguish between benign and malicious activities.

The DNN sub-system is the core of the malware detection system, where the actual learning and classification take place. The primary function in this sub-system is training the model, which involves feeding the processed dataset into a deep learning framework that learns patterns associated with malware behaviour. The DNN consists of multiple layers, including input, hidden, and output layers, that work together to improve detection accuracy. Advanced techniques such as dropout regularization, activation functions, and optimization algorithms are employed to enhance the model's generalization and prevent overfitting.

Once the deep neural network is trained, the final step is evaluating its performance in detecting fileless malware. This sub-system includes Testing and Results, where the trained model is validated using test data to assess its effectiveness. Performance metrics such as accuracy, precision, recall, and F1-score are used to measure how well the model distinguishes between benign and malicious activities. The results from this stage determine whether further optimization is needed before deploying the model in real-world cybersecurity environments.

3. THE PROPOSED DEEP NEURAL NETWORK MODULE

The Deep Neural Network (DNN) Module is the core component responsible for training a classification model based on the optimized feature set. This module consists of multiple layers, including an input layer, several hidden layers with activation functions, and an output layer that determines whether a sample is benign or malicious. During training, the model learns to recognize malware patterns by adjusting weights through back-propagation and an adaptive optimization algorithm (e.g., Adam optimizer). The model undergoes extensive training and fine-tuning using labelled datasets, ensuring that it generalizes well to unseen malware samples. The trained DNN model serves as the decision-making engine for classifying new malware instances.

3.1 System Algorithms

This section presents the algorithm of the various modules applied for the fileless malware classification system.

The AVOA for feature selection (Abdollahzadeh et al., 2021)

1. *Inputs: The population size N and maximum number of iterations T*

2. *Outputs: The location of Vulture and its fitness value 3: Initialize the random population Pi(i = 1,2, ...,N)*
3. *while (stopping condition is not met) do*
4. *Calculate the fitness values of Vulture*
5. *Set PBestVulture1 as the location of Vulture (First best location Best Vulture Category 1)*
6. *Set PBestVulture2 as the location of Vulture (Second best location Best Vulture*
7. *for (each Vulture (Pi)) do*
8. *Select R(i) using*
9. *Update the F*
10. *if (|F| ≥ 1) then*
11. *if (P1 ≥ randP1) then*
12. *Update the location Vulture*
13. *else*
14. *Update the location Vulture*
15. *if (|F| < 1) then*
16. *if (|F| ≥ 0.5) then*
17. *if (P2 ≥ rand P2) then*
18. *Update the location Vulture*
19. *else*
20. *Update the location Vulture*
21. *else*
22. *if (P3 ≥ randP3) then*
23. *Update the location Vulture*
24. *else*
25. *Update*
26. *End*

The PCA Algorithm (Pechenizkiy, et al., 2004)

Input: Feature matrix X

Output: Transformed dataset with reduced features

1. *Standardize the dataset X by normalizing feature values.*
2. *Compute the covariance matrix of X.*
3. *Perform Eigen decomposition on the covariance matrix.*
4. *Select top principal components based on explained variance.*
5. *Transform the dataset using the selected components.*
6. *Return the reduced dataset.*

Back-Propagation algorithm (Ogbeta and Nwobodo, 2022);

1. *Start*
2. *Parameters initialization*
3. *Set gradient loss tolerance* = $1e-10^{6}$
4. *Forward propagation:*
5. *Compute output of training samples*
6. *For Y output unit K*
7. $\delta_k \leftarrow o_k\,(1-o_k)(t_k-o_k)$
8. $for\ each\ hidden\ layers\ unit\ h$
9. $\delta_k \leftarrow o_h(1-o_k)\sum_{k\in outputs} w_{h,k}\delta_k$
10. *Apply regularization algorithm to loss function*
11. $Update\ each\ neuron\ weights\ w_{ij}$
12. $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$
13. $where\ \Delta w_{ij} = \mathfrak{y}\delta_j x_{ij}$
14. End

Dropout Regularization algorithm (Salehin and Kang, 2021)

1. *Start*
2. *Input activation values*
3. *Parameter initialization (weight, bias, drop rate p)*
4. *Generate probability vector (p) for neurons layers*
5. *Choose probability vector as (0 and 1)*
6. *Apply dropout of neurons using probability vector of neurons set to 0*
7. *Forward propagation with dropout*
8. *Compute the next layer activated value*
9. *Compute weight and bias sum of dropout*
10. *Apply nonlinearity with* $tanh$ *activation function*
11. *Repeat for each layer*
12. *Update neural network training*
13. *End*

The DNN Algorithm for Malware Classification

1. *Initialize network with input, hidden, and output layers.*
2. *For each training iteration*
3. *Perform Forward Propagation to compute predictions.*
4. *Compute the loss using a chosen loss function.*
5. *Apply Back-propagation to update weights.*
6. *Repeat until the model converges.*
7. *Return the trained model.*
8. *End*

4. SYSTEM IMPLEMENTATION

The programming environment for developing the fileless malware classification system was chosen based on its ability to support machine learning, deep learning, and optimization techniques efficiently. Python was selected as the primary programming language due to its extensive ecosystem of libraries and frameworks for data science, cybersecurity, and artificial intelligence.

The development was carried out using Jupyter Notebook and Google Colab, providing an interactive coding environment with cloud-based GPU acceleration for deep learning model training. Essential libraries such as TensorFlow and Keras were used for deep learning, while Scikit-learn supported traditional machine learning tasks like feature selection and classification. Additionally, NumPy and Pandas were utilized for efficient data manipulation, and Matplotlib and Seaborn helped visualize results. The integration of these tools ensured a seamless development process, from data pre-processing and feature selection using the AVOA to DNN training and evaluation.

4.1 System Testing

System testing is a crucial phase to evaluate the functionality, reliability, and performance of the fileless malware classification system. This stage ensures that the implemented system meets the defined requirements and performs as expected under various conditions. The testing process includes the Test Plan, Test Data, and a comparison of Actual vs. Expected Results.

4.2 Test Plan

The test plan outlines the approach for validating the system's effectiveness in classifying fileless malware. The primary objectives of testing include:

a. Verifying the accuracy and efficiency of the AVOA-based feature selection process.
b. Evaluating the deep learning classification model to ensure high detection rates.
c. Assessing system response time and computational efficiency.
d. Ensuring the system correctly identifies and quarantines detected threats.

Different testing methodologies were applied, including unit testing for individual modules, integration testing to ensure seamless interaction between components, and performance testing to validate system efficiency under various workloads.

4.3 Test Data

The test dataset was composed of 1000 samples, including a mix of normal system processes, benign software, and fileless malware. These data points were pre-processed and subjected to feature selection using AVOA, followed by classification using

a Deep Neural Network (DNN). The test data included:

i. Feature vectors of normal files (e.g., system processes, legitimate software).
ii. Feature vectors of malware samples (fileless attacks, memory-based threats).
iii. Augmented data generated using synthetic techniques to improve model robustness.

5. RESULT AND DISCUSSION

The system integration as software involves the seamless incorporation of the fileless malware detection model into a fully functional software solution. This integration ensures that the detection algorithm operates efficiently within a real-world environment, facilitating continuous monitoring and analysis of malicious activities across different operating systems. The software implementation of the detection system was tested on Windows, Linux, and macOS, where it successfully scanned files, identified fileless malware, and generated real-time alerts. The graphical user interface provided an intuitive platform for users to interact with the detection model, allowing them to monitor detection rates, analyze malware behaviours, and manage security responses effectively. Figure 2 presents the GUI interface of the software.

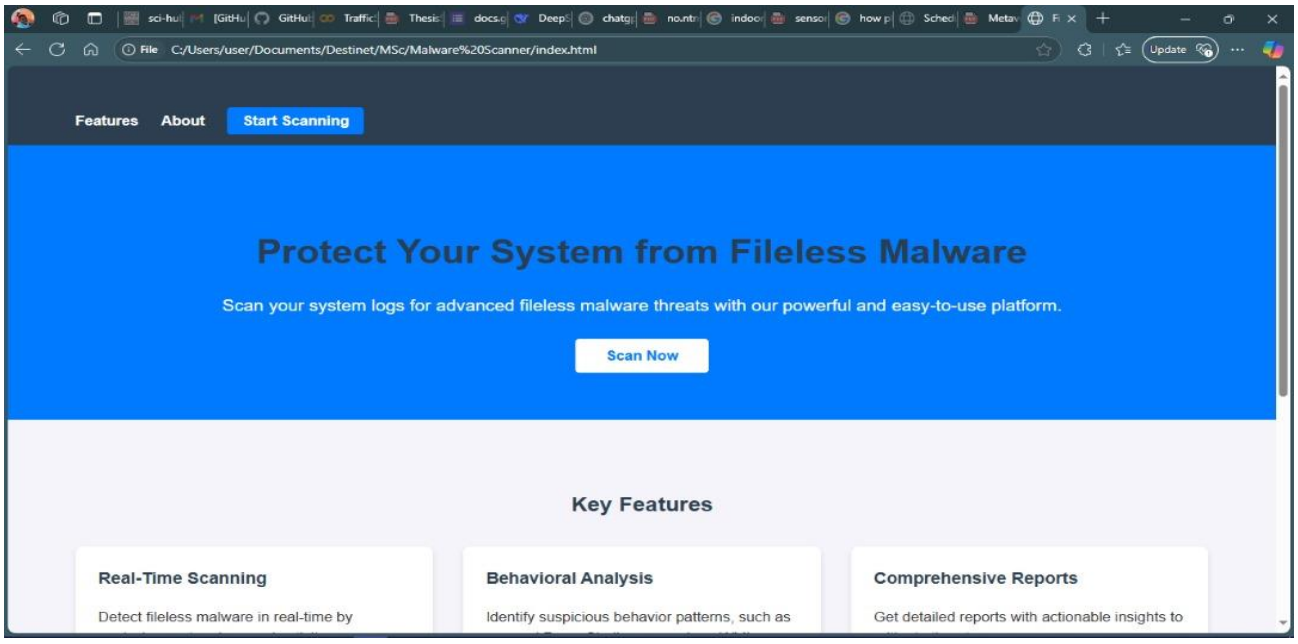

Figure 2: The GUI interface of the software

Figure 2 presents the GUI interface of the software for the classification of files malware. This interface allows real-time system scanning to detect malicious activities like the fileless malware which has continued to facilitate cyber-attack. To test the software files from various operating system was collected from windows, Linux and Mac IOS, then imported them separately on the software as shown in Figure 3.

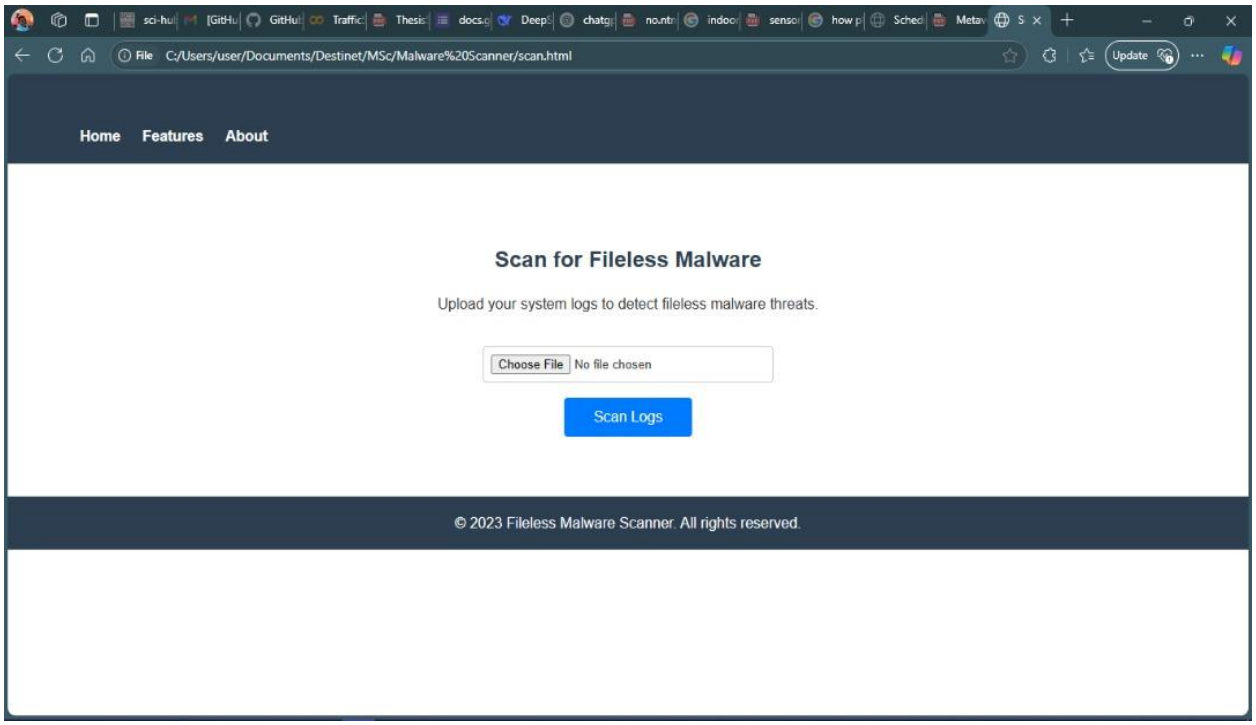

Figure 3: Interface to load the test data samples

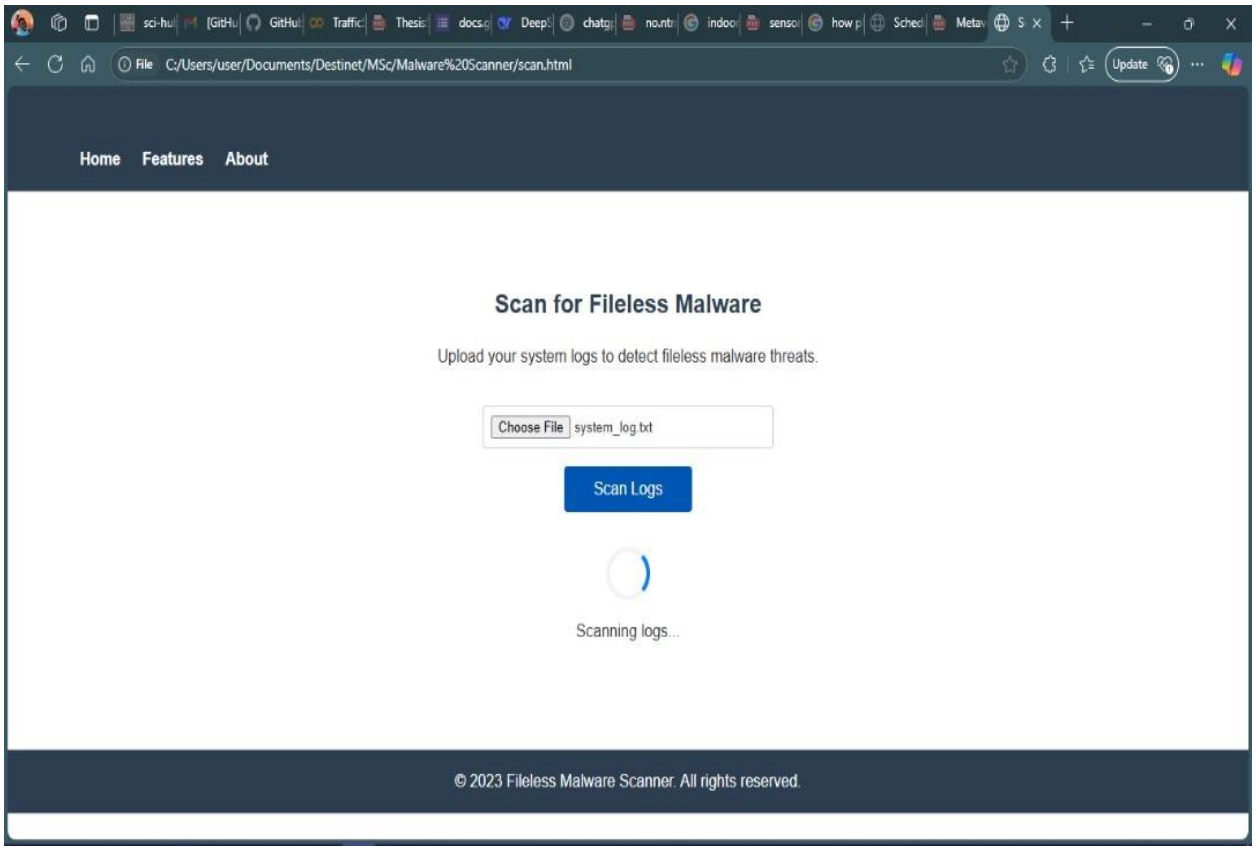

Figure 4: Interface showing the scanning process when tested on Windows IOS

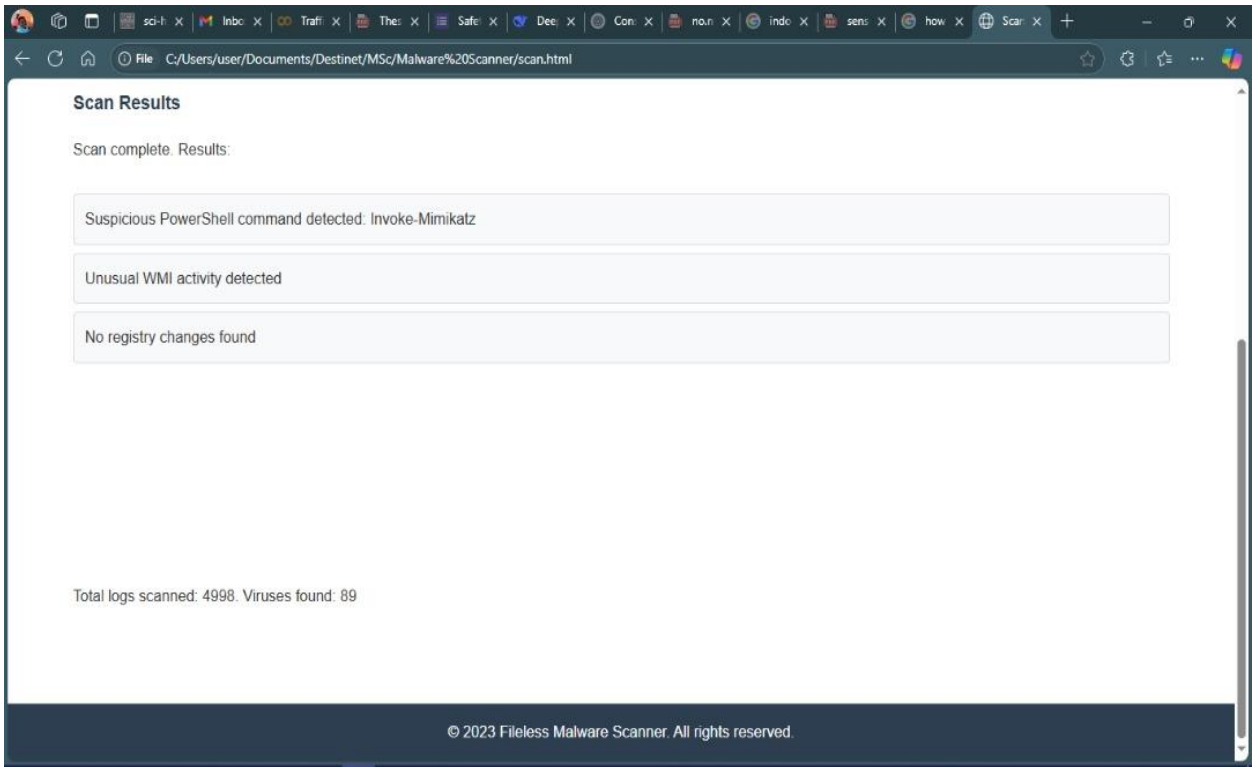

Figure 5: Results of the scanning outcome on Windows IoS.

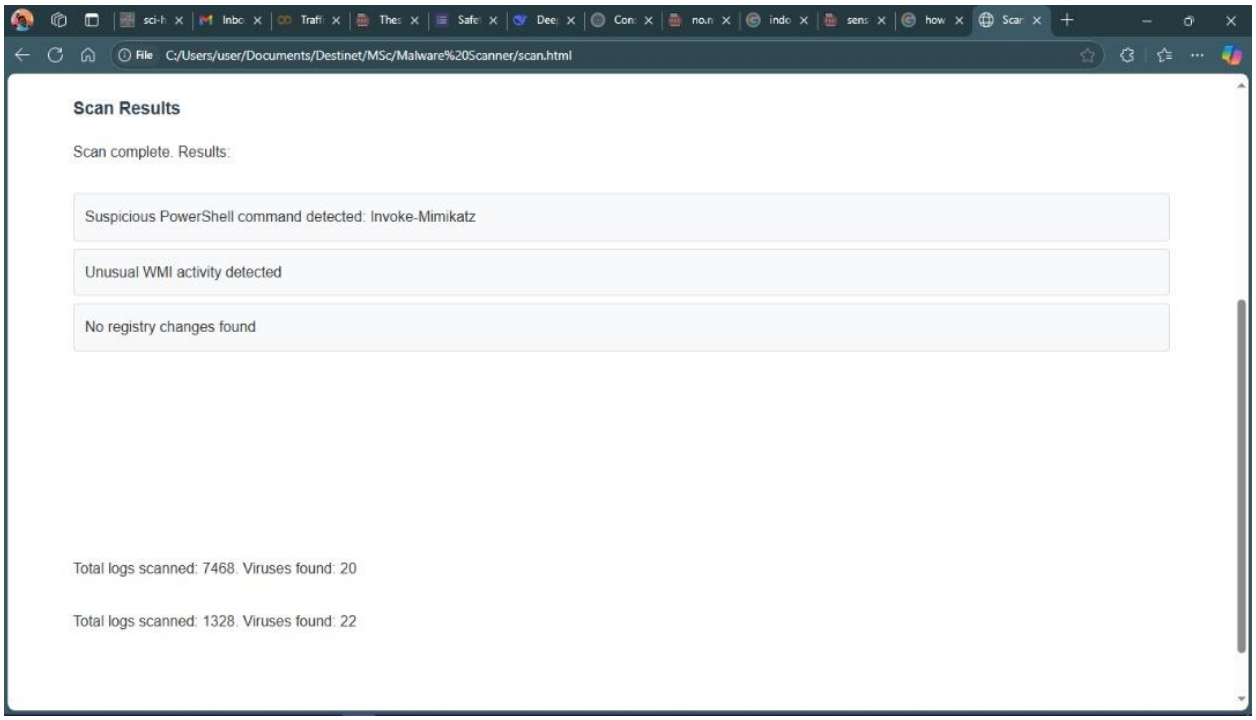

Figure 6: Result of scanning on Linux IOS

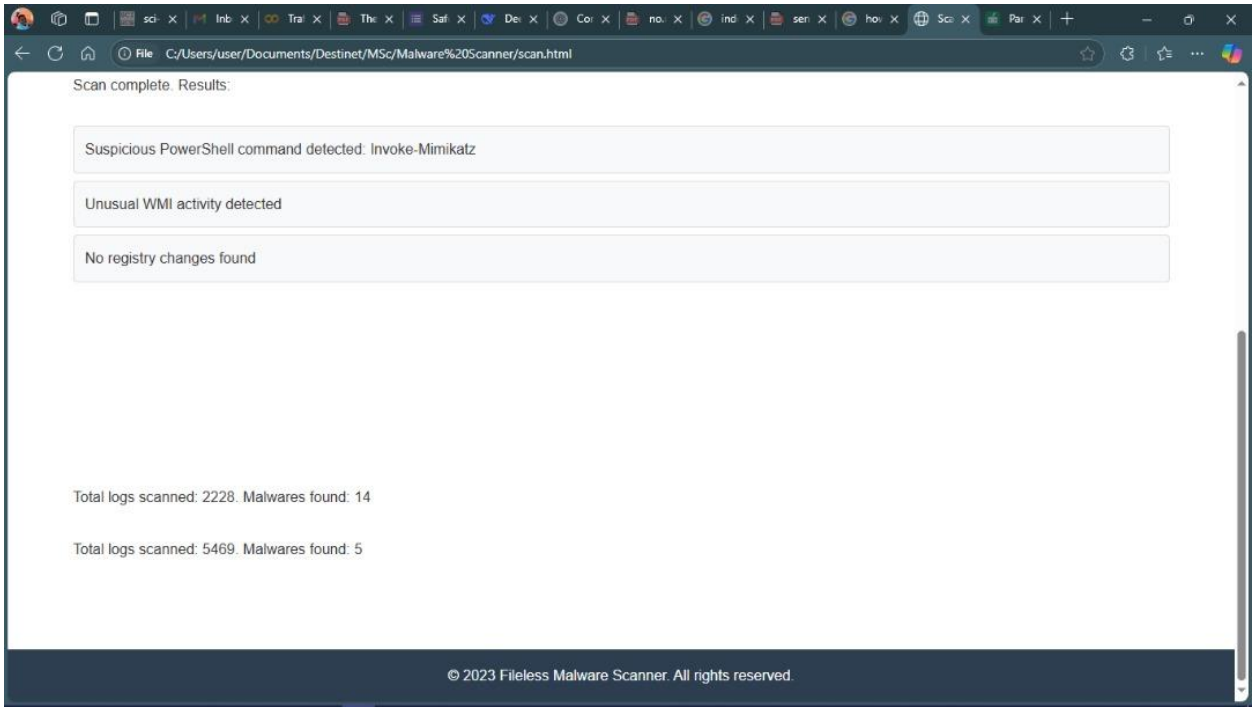

Figure 7: Result of scanning on Mac IOS.

In Figure 3, when the test data was imported to the system, automatically the scanning process began as shown in Figure 4, allowing the DNN based fileless malware classifier to perform scanning of features and classifying fileless malware on the OS. In the Figure 5, the result of the scanning process upon completion was reported for Windows IOS, Figure 6 reported result of scanning for Linux IOS and Figure 7 was reported for Mac IOS testing.

5.1 System Security

System security is a critical aspect of the fileless malware classification system, ensuring that the system remains protected against cyber threats, unauthorized access, and data breaches. Given that fileless malware operates without traditional file-based signatures, the security mechanisms implemented in this system are designed to detect, isolate, and mitigate such attacks while maintaining system integrity.

The security implementation includes multiple layers of protection, such as access control, data encryption, and secure model execution. Role-based authentication ensures that only authorized users can access system functionalities, preventing malicious actors from tampering with the malware detection process.

Additionally, encrypted communication channels safeguard sensitive data during model training and classification. To prevent adversarial attacks, the system employs secure model execution environments, where the deep learning model is protected against manipulation. The AVOA enhances security by dynamically adapting feature selection to counter evasion techniques used by malware developers. Furthermore, real-time monitoring detects anomalies in system behaviour, automatically triggering an alert or quarantine

action when suspicious activity is identified. Overall, the system security measures ensure that the malware classification model remains robust, resistant to attacks, and capable of maintaining high detection accuracy in real-world cybersecurity environments.

5.2 Training

The system training phase involves teaching the malware classification model using a carefully curate dataset of both benign and malicious fileless malware samples. The training process includes data pre-processing, feature extraction using the AVOA, dimensionality reduction with PCA, and classification using a DNN. The system undergoes multiple iterations to optimize its accuracy, reducing false positives and negatives. The model is evaluated using performance metrics such as precision, recall, F1-score, and accuracy to ensure robustness before deployment.

5.3 Documentation

Comprehensive documentation is essential for ensuring ease of use, maintenance, and future upgrades of the system. The documentation consists of:

1. User Manual: Provides step-by-step guidance on how to operate the malware detection system, including input data formats, expected outputs, and troubleshooting procedures.
2. Developer Documentation: Contains detailed descriptions of the AVOA-based feature selection model, deep learning architecture, hyperparameter tuning, and system APIs for future enhancements.
3. System Architecture: Includes diagrams illustrating the data flow, feature extraction, classification pipeline, and security protocols implemented.
4. Testing Reports: Provides results of system testing, validation datasets, and performance benchmarks, ensuring the system meets security and efficiency requirements.

6. CONCLUSION

This paper developed a robust approach for detecting fileless malware using deep learning techniques. The study leveraged data from two primary sources such as the Cyber Dome, which provided real-time attack data, and Kaggle, which served as a supplementary dataset for enhancing model robustness. The collected data underwent a comprehensive pre-processing phase, including integration and cleaning, to ensure quality and consistency. Feature selection was carried out using the African Vulture Optimization Algorithm (AVOA), which helped in identifying the most relevant attributes while reducing computational overhead. Further dimensionality reduction was applied using PCA to improve the efficiency of the detection model. The refined dataset was then used to train a DNN, which was designed to classify and detect fileless malware accurately.

The software integrated with the developed model was extensively tested on different operating systems, such as Windows, Linux, and mac-OS, to evaluate its generalization ability across diverse environments. The results demonstrated high detection accuracy across all platforms, with notable success rates of 99% for Windows, 96% for Linux, and 92% for mac-OS. Additionally, the classifier maintained a low false alarm rate, ensuring minimal disruption to legitimate processes.

Corresponding Author Tel: +2348065375935

Following model training and validation, the deep learning-based detection system was integrated into a software framework. This integration facilitated real-time detection and monitoring of fileless malware attacks across different platforms. The system's effectiveness was validated through rigorous testing on various operating systems, confirming its capability to detect and mitigate fileless malware threats efficiently.

This study provides a significant advancement in cybersecurity, particularly in combating fileless malware threats that do not rely on traditional file-based execution. The findings underscore the potential of deep learning in strengthening endpoint security and offer a scalable approach to protecting computing environments against sophisticated cyber threats.

7. REFERENCES

Abdollahzadeh, B., Gharehchopogh, F. S., &Mirjalili, S. (2021). African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. Computers and Industrial Engineering, 158, 107408. https://doi.org/10.1016/j.cie.2021.107408

Atapattu, M., & Jayawardena, B. (2021). An approach to detect fileless malware that maintains persistence in Windows environment. Proceedings of the International Conference on Advances in Computing and Technology (ICACT), Kelaniya, Sri Lanka.

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, Seoul, South Korea. ACM, 482–493. https://doi.org/10.1145/3307334.3326083

Kaspersky Security Bulletin 2022. (2022). Statistics. Kaspersky Security Bulletin. Available at: https://securelist.com/ksb-2022-statistics/108129/ (Accessed on 29 November 2023).

Mohanta, A., & Saldanha, A. (2020). Malware analysis and detection engineering: A comprehensive approach to detect and analyze modern malware. Springer. https://doi.org/10.1007/978-1-4842-6193-4

Obini, C., Jeremiah, C., & Igwe, S. A. (2024). Development of a machine learning-based FM filter system for cybersecurity. J 2192. Nig. Soc. Physiological Sciences, 6.

Ogbeta, L. K., & Lois, N. (2023). Neuro-based strategy for real-time protection of wireless network ecosystem against DDoS attack. I1SRED, ISSN 2581-7175, 79–98.

Pechenizkiy, M., Tsymbal, A., &Puuronen, S. (2004). PCA-based feature transformation for classification: Issues in medical diagnostics. 17th IEEE Symposium on Computer-Based Medical Systems, Bethesda, USA, 535–540. https://doi.org/10.1109/CBMS.2004.1311770

Salehin, I., & Kang, D. K. (2023). A review on dropout regularization approaches for deep neural networks within the scholarly domain. Electronics, 12, 3106. https://doi.org/10.3390/electronics12143106

Siddiqui, A. A., Ali, I., Arbab, S., & Kumari, S. (2024). Efficient malware investigation and recognition using machine learning algorithms. Asian Bulletin of Big Data Management, 4.

https://doi.org/10.62019/abbdm.v4i3.209

Singh, J., Thakur, D., Ali, F., Gera, T., & Kwak, K. S. (2020). Deep feature extraction and classification of Android malware images. Sensors, 20, 7013. https://doi.org/10.3390/s20247013

Snow, D. (2021). Investigating fileless malware [PhD thesis]. Utica College.

Tekiner, E., Acar, A., Uluagac, A. S., Kirda, E., & Selcuk, A. A. (2021). SoK: Cryptojacking malware. Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria. IEEE, 120–139. https://doi.org/10.1109/EuroSP51992.2021.00019