Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

BEHAVIORAL ANALYSIS OF BACK-DOOR MALWARE EXPLOITING HEAP OVERFLOW VULNERABILITIES USING DATA MINING AND MACHINE LEARNING

Ali Raza Khaliq[1], Subhan Ullah[1], Tahir Ahmad[2], Ashish Yadav[3], Imran Majid[4]
[1]Faculty of Computer Science, National University of Computer and Emerging Sciences (NUCES-FAST), Islamabad 44000, Pakistan
[2]Center for Cybersecurity, Bruno Kessler Foundation, 38123, Trento, Italy
[3]New York University Brooklyn, NY, USA, [4]University of Warwick, UK

[*]Corresponding author: Tahir Ahmad (email; ahmad@fbk.eu)

Abstract: **Backdoor malware remains a persistent and elusive threat that successfully evades conventional detection methods through intricate techniques, such as registry key concealment and API call manipulation. In this study, we introduce an approach to detect backdoor malware, drawing upon the diverse domains of cybersecurity. Our method combines static and dynamic analysis techniques with machine learning methodologies, particularly emphasizing classification, and feature engineering. Through static analysis, we extract valuable raw features from malware binaries. Discerning the most significant attributes, we delve into the calling frequencies embedded within these raw features. Subsequently, these selected attributes undergo a meticulous refinement process facilitated by feature engineering techniques, culminating in a streamlined set of distinctive features. To accurately detect malware exploiting heap-based overflow vulnerabilities, we employ three distinct yet potent classifiers: J48, Naïve Bayes, and Simple Logistic. These classifiers are trained and tested using carefully curated feature sets. Our approach combines machine learning and data mining principles to develop a comprehensive malware detection methodology. We demonstrate the efficacy of our approach through rigorous validation using two distinct settings: a dedicated training/testing set and a comprehensive 10-fold validation. Our approach simultaneously achieves 90.29% and 84.46% accuracy in train/ test split and cross-validation strategies.**
Keywords: Backdoor malware, Malware detection, Heap-based overflow vulnerability

I. INTRODUCTION

Heap overflow is a type of buffer overflow that occurs in a heap data area and is a memory segment used for storing program data dynamically allocated by the application at runtime [1]. The heap also stores the global variables. Heap overflow can be exploited by corrupting program data in a heap and often by manipulating pointers or indices to overwrite memory locations before or after the buffer. Each portion of memory in a heap contains boundary tags that contain information related to memory management. When a heap buffer overruns, the control statistics in these tags can be overwritten, resulting in access violations and memory address overwrites. If the overflow is executed in an organized manner, it can allow an attacker to overwrite a memory location with crafted input. This vulnerability can directly affect the CIA triad, which includes confidentiality, integrity, and system availability [2]. The consequences of heap overflow vulnerability include unauthorized reading of memory, execution of unauthorized programs, evasion of protection mechanisms, modification of memory by running arbitrary or unauthorized programs, crashing of systems, generation of DDoS attacks, resource consumption, and potential infinite loops in the program [3,4].

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

Attackers can leverage heap overflow vulnerabilities to inject malicious code or data into a program's memory, potentially leading to unauthorized access and control over a system. Backdoor malware can be designed to exploit heap overflow vulnerabilities to infiltrate and compromise a system. For example, the malware may use a heap overflow to inject its code into a running application, thereby establishing a backdoor for remote access. According to Malwarebytes, backdoor malware increased by almost 73% in 2018 and was listed among the top 10 malware commonly detected in organizations [5]. Hackers use backdoor malware to bypass standard authentication and gain unauthorized access to a system. Hackers use backdoors to install malicious files or programs, modify codes, and gain unauthorized access to a system. Vulnerabilities, such as buffer overflow, heap overflow, cross-site scripting, and remote administration, can introduce backdoors into a system, which can then be used to steal information and personal data from user computers. Therefore, we focused on detecting backdoor malware to protect the user and organizational data. Moreover, the absence of behavioral information in existing malware datasets used in machine learning could facilitate a stronger impact by ML in malware analysis [6].

Malware has different variants and impacts depending on exploiting vulnerabilities and the tricks used by attackers in social engineering to compromise a system. End users and companies widely use antivirus software but are not always efficient in detecting new and unknown malware. There is a need for improvement in detecting and eliminating new threats. Different malware detection schemes, such as anomaly detection and signature- based malware detection, have been proposed, but they have limitations in detecting sophisticated malware that updates themselves to avoid detection [7,8]. Various techniques have been recommended to identify heap overflow vulnerabilities; however, most require deep code analysis and runtime execution, which can be cumbersome. Data mining and machine-learning techniques have introduced new dimensions in malware analysis. This paper proposes a hybrid detection technique that combines data mining and machine learning to identify heap overflow vulnerabilities and predict their outcomes. This study focused on the dynamic analysis of files in a sandbox environment. We used machine-learning-based malware detectors that rely on datasets (extracted from on- line repositories such as Virus total (https://www.virustotal.com/) and Vxheaven (https://www.vxheaven.org)) and extracted features from malicious and benign files to detect previously unseen malware. We also explored different feature extraction methods, including static, dynamic, and hybrid approaches, where static analysis extracts feature from malware without executing it, dynamic analysis runs malware in a safe environment using a sandbox, and hybrid analysis combines static and dynamic analyses for feature extraction. The proposed approach provides better results than static analysis, which is time-consuming and relies on human analysts.

The contributions of this paper are as follows,

- This research unveils the connection between backdoor malware and heap overflow vulnerabilities, shedding light on their exploitation tactics and serving as a foundation for the proposed detection method.
- A novel approach emerges, merging data mining and reverse engineering, to classify malware by behavior, particularly identifying heap overflow exploitation as a hallmark of malicious activity.
- The approach's effectiveness is validated through machine learning classifiers, highlighting J48's superior accuracy in identifying backdoor malware leveraging heap overflow vulnerabilities, confirming its real-world viability.

The rest of the paper is organized as follows. Section 2 comprehensively reviews related work in malware detection, heap overflow vulnerabilities, and behavioral analysis. In Section 3, we delve into the methodology of our proposed approach, detailing the data mining techniques, reverse engineering procedures, and the behavior-based detection framework. Section 4 presents the experimental setup, including the dataset, evaluation metrics, and classification algorithms. The results of our experiments and their analysis are discussed in Section 5. Finally, Section 6 offers a concise conclusion, reflecting on our work's contributions, limitations, and prospects in detecting backdoor malware through heap overflow exploitation.

PJETS
E-ISSN 2224-2333
P-ISSN 2222-9930

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984

II. LITERATURE REVIEW

A. Malware Detection

Zolotukhin and Hamalainen [11] proposed an anomaly detection approach. They first analyzed the operation of the code sequence of malicious files. Then, they used the n-gram model to extract other features for achieving a higher accuracy of detection of malicious files. Their algorithm consists of two stages. In the first stage, they analyzed the feature matrix obtained from the training set to identify benign files using support vector machine clustering. In the second stage, they detect malicious files that entered the system using an opcode sequence. However, this approach may not be practical for all types of malwares with different methods and API calls. Markel and Bilzor [12] proposed an approach that learns from metadata in the header of executable files, PE32, to identify malware and benign files. They efficiently detect malware at an early stage of execution, and their main goal was to design a classifier that can accurately and efficiently detect malicious and benign files. However, this approach also may not be sufficient for all malware types. Z. Xu et al. [13] proposed a hardware-level approach for detecting malware, focusing on registry modification and changes in system-level data structures, kernel-level API calls, and user-level heap modifications. This approach aims to overcome the limitations of software-based approaches, which can be easily exploited and rendered useless. Chen et al. [14] proposed a new Swarm Learning (SL) approach for decentralized training on a temporary no central server. In this approach, a participant node selects a temporary server for each round of the training and does not share their private dataset for aggregation in a central server fairly and securely. They investigate backdoor attacks on swarm learning to explain the high-security risk. The approach provides comparatively accurate defensive methods for backdoor attack detection and prevention.

B. Static and Dynamic Analysis

Firdausi et al. [15] proposed a model for detecting the malicious intent of malware through dynamic and static analysis, using three different phases of the 20s, 60s, and 300s. They combined the results obtained from these methods and trained them on various ML approaches, such as neural networks, SVM, and k-nearest neighbors, to classify malware families. They achieved 92% accuracy and an efficient categorization of malware and their families, with most malware exhibiting malicious intent within 20 seconds. This approach reduces the time required for malware detection and provides time slots for malware analysis in sandbox environments. Ranveer et al. [16] proposed a feature-extraction method based on static and dynamic analyses to detect malware. They discussed the advantages of both approaches and proposed a hybrid approach. Kilgallon et al. [17] developed a model for the detection of the malicious intent of malware through dynamic and static analyses. They combined the analysis results of the three different time intervals (e.g., 20, 60, and 300 seconds). They trained them using machine learning approaches such as neural networks, SVM, and k-nearest neighbors. Their method achieved an accuracy of 92% in classifying malware families and found that most malware showed malicious intent within 20 seconds. Santos et al. [18] proposed a hybrid approach that combines static and dynamic analysis for improved accuracy in malware detection. They utilized machine-learning approaches and selected features from opcode sequences, PE, API calls, registry modification, network connection, and process behavior. Their results showed that the hybrid approach outperformed static and dynamic analyses alone.

C. Machine Learning and Malware Analysis

Chowdhury et al. [19] proposed a malware detection system with significant com- ponents, including malicious files, pre-processing, feature extraction, feature reduction, feature classification using an artificial neural network (ANN), and detection. They trained their model on a dataset of features extracted from malware and cleanware. They compared the accuracy of their approach with that of other mechanisms, such as SVM, J48, Naive Bayes, and Random Forest, using similar features. Their proposed scheme using an ANN with n-gram features showed higher accuracy rates. However, like other approaches, this method may also not be effective for all malware because they do not use the same techniques or API calls. Chowdhury et al. [20] proposed a two-stage classification process involving training and testing stages. They provided a system with different malicious and benign files for training purposes. The system

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

learns from labelled data to detect malware using the feature reduction technique for the input of multilayer perception called the BAM network layer. The experimental results showed higher accuracy than other techniques, with the hybrid BAM and MLP approach achieving almost 94% accuracy compared to SVM, k-nearest neighbors, and other techniques. Joshi et al. [21] presented an ML-based malware detection technique, specifically a random forest classifier, in a Linux virtual machine environment. Their framework consists of three major parts: virtualization for monitoring malware attacks, a web-based interface for administration access to the system for data extraction and capturing of behavioral information, and data analytics for data analysis of the stored data in a database. However, they only used memory analysis in the Linux environment, which may not be effective in a Windows-based environment commonly used in organizations. Willems et al. [22] proposed dynamic analysis using Cwsandbox to monitor system calls, DLL, and API hooking, emphasizing the importance of dynamic analysis and API hooking in understanding malware behavior. They generated reports in a human-readable language, extracted the features to update signatures of the signature-based antiviruses, and used them in ML-based approaches to detect malicious activities in networks. They focus solely on API hooking. Loi et al. [23] discussed backdoor malware and its impact on computer systems, identifying network weaknesses that allow malware to exploit and access malicious activity. They proposed a low-cost scheme for users to detect backdoor malware in networks without expensive security solutions, achieving a detection rate of up to 90%. However, their scheme was designed specifically for Windows platforms, not for IOS, Android, or Linux [24]. K. A. Asmitha and P. Vinod [25] proposed a novel methodology utilizing machine learning for identifying malicious executable linkable files. They used a system call tracer to separate system calls and efficiently identify the best feature set for detecting benign and malware files. Yang et al. [26] proposed a new attack called Jigsaw Puzzle (JP), which learns a trigger that complements the latent patterns of the malware samples and activates the backdoor. They also focus on the possible triggers in software code using bytecode gadgets harvested from the benign software. They claim the stealthiness of the Jigsaw puzzle as a backdoor against the current defenses and consider it a potential threat in a realistic environment. Further, they present an extensive evaluation and show the possibility of the Jigsaw Puzzle detection using the currently available method of Severi et al. [27]. They further discussed that MNTD [28] is a classifier-based detection method that can successfully identify backdoor malware.

D. Summary

The reviewed literature mainly focuses on API calls and does not address vulnerabilities in the system. In addition, many approaches rely on selecting only one feature to detect malware, which may not be effective for all types of malwares that use different activities and techniques for infection and data theft. Backdoor malware is often not focused on in the research, as their infection and data theft approaches differ from other malware, making them difficult to detect without deep analysis. Detecting malware that exploits software vulnerabilities is a difficult task. Traditional antivirus programs cannot detect malware owing to their working principles [29,30]. Simultaneously, machine learning classifiers often produce false positives and fail to detect malware based on exploiting vulnerabilities [31]. A new malware detection scheme is required to detect malware that exploits software vulnerabilities, which will help program and security analysts build se- cure applications and protect organizations. Although researchers have proposed different techniques for detecting malware that exploit overflow vulnerabilities [32], they have failed to inspect the runtime performance of malware, and their methods are ineffective against encrypted features. We extracted and used over 20 features in our proposed framework to efficiently detect malware that exploits heap-based overflow [33] vulnerabilities. Our proposed methodology is automatic and flexible, making it suitable for deployment in any operational environment.

III. PROPOSED APPROACH

The proposed method leverages data mining and machine learning strategies to effectively identify backdoor malware that exploits vulnerabilities in malware binaries. Figure 1 illustrates the architectural blueprint depicting the proposed approach's essence.

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

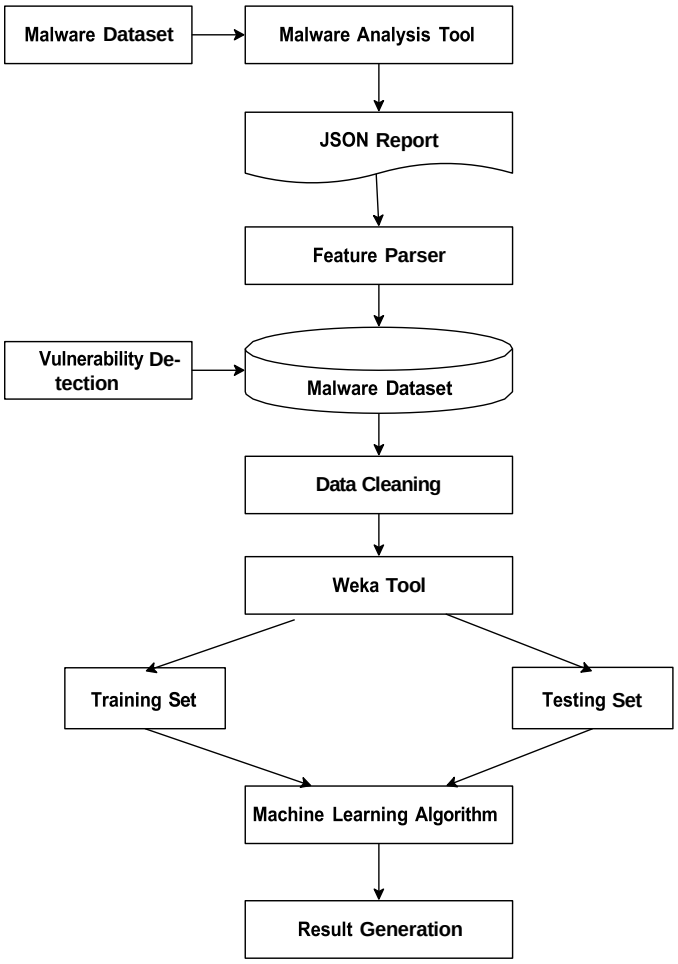

Figure 1. Architectural Diagram of the proposed approach.

The main components are as follows,

- Malware Dataset: We collected our dataset from various sources, including antivirus companies and online platforms. The dataset consists of malware samples selected based on specific criteria to focus on heap-based overflow vulnerabilities. The dataset was carefully curated and analyzed to ensure its relevance to our research. In particular, the initial dataset consists of malware samples sourced from online repositories such as Virus total (https://www.virustotal.com/) and Vxheaven (https://www.vxheaven. org). These samples serve as the foundation for training and testing the detection model. The main challenge in malware detection study is highly imbalanced, where the number of data having malware is highly less (minority) than the normal one (majority). However, in our specific case, our dataset is not imbalanced. Both malware and benign samples consisted of 103 instances each. Therefore, we decided not to apply data augmentation techniques like SMOTE or GAN in this study, as they are typically used to address imbalanced datasets.
- Malware Analysis Tools: To analyze the malware, the Cuckoo sandbox [34] is employed. This sandbox facilitates the dynamic analysis of malware by executing them in a controlled environment to capture their behavior. By running malware within this environment, the sandbox isolates potential threats, thus enabling the observation of their behavior and interactions.
- Feature Parser: Extracting relevant features from the dynamic analysis is essential for classification. The feature parser reads the generated JSON reports and selects pertinent attributes, such as file size, file type, API calls, network communication, and system calls. These features are then stored in a CSV format for

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

subsequent processing. The parser automatically extracts attributes from the JSON reports generated by the malware analysis tool, streamlining the feature extraction process.

- Vulnerability Detection: Reverse engineering plays a crucial role in understanding the intricacies of malware behavior. Our technique employs both static and dynamic analyses to detect backdoor malware. Features derived from the static analysis are combined with insights from dynamic analysis to enhance detection accuracy. Olly Dbg, a dynamic analysis and debugging tool, scrutinizes the malware's execution flow, memory utilization, and interaction with the system, unveiling potential vulnerabilities and malicious intents.
- Data Cleaning: The feature extraction process can introduce noise or inconsistencies, such as missing values, duplicates, or irregular formatting. The data cleaning component addresses these issues using data imputation, duplicate removal, normalization, and encoding categorical attributes. This step ensures the extracted features are reliable and suitable for subsequent machine-learning stages.
- Weka Tool: The Weka tool, a comprehensive collection of machine learning algorithms designed for data mining tasks (https://www.cs.waikato.ac.nz/ml/weka/), is used for the subsequent analysis stages.
- Machine Learning Algorithms: Machine learning algorithms are applied to the pre-processed and transformed data to derive classification results. The dataset, once prepared, is used to train a machine learning or statistical model. The selection of algorithms depends on the dataset's characteristics and specific requirements. The trained model is then utilized to predict class labels for new malware samples, enabling the evaluation of the approach's performance through metrics like accuracy, precision, and recall. We have chosen J48, Naive Bayes, and Simple Logistic classifiers for this study. We were motivated by their proven performance across diverse datasets, widespread use in similar studies, interpretability, ease of implementation, and our goal to provide a comprehensive assessment of the proposed methodology's applicability across different scenarios.

IV. EXPERIMENTAL SETUP

In this section, we explain the building blocks of our experimental evaluation.

A.Setup

Malware executables are classified into two categories: win-32 and win-64 executable files. Sandboxes are commonly employed for the static or dynamic analysis of malware. Our analysis utilizes sandbox tools such as Cuckoo, and OllyDbg. Code analysis is conducted to identify the type of software vulnerability by analyzing malware with debugging tools like OllyDbg. Subsequently, we use a PHP parser to extract features from the data generated during our analysis, including logs and reports. It is followed by a data cleaning process to select relevant features and create a dataset. We utilize Weka to work with machine learning models and data analysis. Weka doesn't directly detect exploits but assists in generating results based on the analysis of extracted features and a pre-trained dataset.

B.Prototype Implementation

To conduct accurate and efficient malware analysis, it is crucial to use appropriate software and configurations.

- Test Environment: We set up an Ubuntu environment and installed the Cuckoo sand- box for our analysis. Subsequently, we created virtual machines running Windows 7 within Ubuntu, equipped with adequate resources, including 2GB of RAM, a 256GB hard drive, and a 2 GHz processor.
- Resource Allocation: We intentionally allocated resources to our virtual machines, providing them with 2 GB of RAM and a 256 GB disk. This resource allocation strategy aims to counteract potential evasion techniques that attackers might employ in low-resource analysis environments [35].
- Virtual Machine Setup: We utilized VirtualBox to create our virtual machines on the Ubuntu host. Each virtual machine was configured with 4GB of RAM, a 4 GHz processor, and a 500GB hard disk. Enabling virtualization was essential to facilitate the creation of virtual machines in Ubuntu or other operating systems. We assigned higher resources to our virtual machines to ensure smooth and efficient operation.

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

- Software Installation: We installed the Cuckoo sandbox after installing Ubuntu. This required downloading various packages, including Python and other dependencies. Additionally, we installed MongoDB to support the web interface, aiding in the analysis process.
- Database Configuration: By default, Cuckoo uses SQLite as its database, but we had the flexibility to choose an alternative database, such as MongoDB, if desired. This provided us with more options for database management.
- Virtual Machine Management: We installed VirtualBox as it is necessary for creating virtual machines for analysis. To collect network traffic and detect malicious activities, we installed TCPdump. Furthermore, we configured a network adapter for the virtual machines to facilitate communication. We employed a virtual machine snapshot mechanism to maintain the integrity and consistency of the analysis environment. After each analysis, this approach allowed us to revert virtual machines to a clean state.
- Enhanced Realism: To enhance the realism of our virtual environment and create additional challenges for potential malware, we introduced supplementary software such as web browsers and Adobe Reader.
- Snapshot Generation: We generated multiple snapshots to enable the simultaneous analysis of four malware samples.
- Configuration: A critical setup aspect involved configuring Cuckoo. Any incorrect configuration could lead to errors during analysis. We made necessary modifications, including values and machine IP additions, in configuration files such as Virtual- Box.conf, routing.conf, reporting.conf, and cuckoo.conf. We updated the network interface settings in the configuration and activated the MongoDB web interface. Global routing among all virtual machines was also enabled.
- Execution Environment: Finally, we initiated the Cuckoo rooter, which created a UNIX socket, and started the Cuckoo web interface from the specified IP address using a browser command.

C. Feature Engineering

The objective of feature engineering is the careful curation of features derived from raw data. This process, encompassing both feature selection and extraction, constitutes feature engineering. In this study, feature engineering is employed to meticulously curate pertinent features from raw data, molding them into coherent models that, in turn, bolster the accuracy of the testing data's predictive models. The features chosen for inclusion within the dataset directly impact the performance of machine learning algorithms deployed for result prediction. Consequently, allocating ample time to selecting features is paramount to achieving optimal outcomes.

D. Feature Parser

The feature extraction phase transforms raw data into a format suitable for machine learning algorithms. Raw data often contains much information which may be irrelevant or redundant for the intended analysis. Feature extraction is a crucial preprocessing step to condense and refine the dataset, reducing its dimensionality while retaining essential information. We employ a specialized tool known as a feature parser to automate the feature extraction process. This custom-designed program, implemented in PHP, facilitates the systematic extraction of relevant attributes from the raw data. The extraction of features from malware executables necessitates a comprehensive analysis of the malware within a secure sandbox environment. In this study, we utilized the Cuckoo sandbox, a widely recognized platform for executing and analyzing potentially malicious software.

Following the execution of malware samples in the sandbox, the analysis generates specific output files that encapsulate critical behavioral data and system interactions. These generated files, representing the behavior of the malware during execution, serve as the primary input to the feature parser. The feature parser then extracts, compiles, and organizes pertinent attributes from these files, producing a refined dataset optimized for subsequent machine learning analysis.

The features used in our analysis were extracted using a custom feature extraction process. This process involved analyzing the behavior of each malware sample in both static and dynamic environments. Key features extracted from

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

the analysis include file name, risk parameter, network connections, number of mutexes, number of loaded libraries, number of process interactions, DNS queries, frequent API calls, downloaded files, process interactions, registry writes, registry reads, mutex count, mutex name, file queries, type of strings, the total count of strings, strings, total count of loaded libraries in .dll format, loaded libraries in .dll format, and the type of malware.

The risk parameter, for instance, represents a calculated risk score based on the mal- ware's behavior and characteristics. Mutexes, loaded libraries, and process interactions are indicators of the malware's activity and resource utilization. DNS queries and network connections help assess network-related behavior, while registry writes and reads provide insights into interactions with the Windows registry. The feature extraction process in- volved careful consideration of these indicators to identify relevant patterns and behaviors associated with heap-based overflow vulnerabilities.

E. Feature Selection

Feature selection is a pivotal cornerstone in malware analysis, significantly influencing the subsequent stages of result generation. It entails meticulously discerning and curating pertinent attributes extracted from malware executables while effectively sieving out extraneous and non-essential information. The goal is to sculpt the dataset to emphasize attributes that bear substantial relevance for the detection process, ultimately refining it to focus on key elements instrumental in achieving accurate results.

Our feature selection methodology in this study reflects a judicious and discerning approach. We harness the power of a controlled sandbox environment, where malware behavior is systematically analyzed. This environment allows us to observe the intricate workings of malware in a secure and isolated setting. Additionally, we employ reverse engineering techniques to delve deeper into the malware's operations, unveiling valuable insights into its functionality and potential vulnerabilities.

During the feature selection phase, we pay particular attention to two fundamental aspects: Indicators of Compromise (IOCs) and discernible patterns of suspicious activity exhibited by the malware. IOCs are crucial clues or markers that indicate the presence of malicious activity, and they play a pivotal role in identifying and characterizing malware.

Table 1. Feature selected from malware executables based on malware working.

Sr. No	Extracted/ Selected Features	Total Counts
1.	Malware Signature	103
2.	Risk Parameter	12
3.	Network Connections	268
4.	Number of Mutex	566
5.	Number of loaded libraries	18834
6.	Number of Process Interactions	655
7.	DNS Queries	33
8.	Process Interactions	103
9.	Registry Writes	103
10.	Registry Reads	103
11.	Mutex Count	131
12.	Mutex Name	6
13.	File Queries	103
14.	Type of Strings	3

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

15.	Count of Strings	11051
16.	Strings	42
17.	Total Count of Loaded Libraries (.dll format)	2639
18.	Loaded Libraries (.dll format)	100

By incorporating IOCs and scrutinizing suspicious behavior patterns, we ensure that our selected features align seamlessly with the intricate characteristics of malware that exploit heap-based overflow vulnerabilities. From the extensive pool of extracted features, we meticulously curate a set of 18 essential attributes that encapsulate the essence of heap-based overflow vulnerability exploitation. These attributes are the bedrock for training and evaluating our machine-learning model. In Table 1, we provide a comprehensive list of these selected features, along with their corresponding counts, offering transparency and insight into the elements driving our detection methodology.

V. EXPERIMENTAL EVALUATION

In this research, 18 features were carefully computed for final evaluation (see Table 1). The process of result generation is described along with comprehensive validation techniques. Detecting malware that exploits heap-based overflow vulnerabilities can be achieved through a straightforward process using classifiers. This process can vary from a few minutes to several months, depending on factors such as the clarity of objectives and scope, availability of the dataset, and pre-processing efforts related to the data. The analysis consists of two main parts: data collection and tool acquisition. The collected data undergoes a pre-processing stage to transform into the required format for classifier implementation and heap overflow detection. The execution of results and analysis of data are crucial steps in understanding the subsequent model and its rule sets. Our dataset consists of 103 backdoor malwares. We simultaneously upload files in Cuckoo for investigation. After accessing the web interface by enabling mango DB, we also enabled remote control, internet connectivity, injection, process memory dump, and simulated hum interface. Each of our analyses ran for 300 Seconds, and a timeout was not enforced, so if the analysis is not completed, it will continue after 300 seconds. After the analysis, reset the machine and give us the analysis. Most analysis results are generated between 6 and 7; the highest score is 13.4. After analysis, we observed different types of activity from malware: queries about computer names, command execution, check memory, crashed process, file creation, and create process. The process created in the hidden window delay malicious URL, try to detect the VM machine and collects information about the installed application etc. For the results, the open-source Weka tool was exploited. Weka is a best-known data mining tool with various machine learning algorithms [36]. The created dataset of 103 malware files comprising 18 features is converted to ARFF format. ARFF files are used to work with Weka machine-learning software.

A. Classification Algorithms

The choice of classification algorithms in this research is underpinned by a strategic selection that capitalizes on each algorithm's unique strengths. The J48 classifier, grounded in the C4.5 decision tree algorithm, stands out for its transparency and interpretability, which align seamlessly with the cybersecurity domain's need for understanding the decision- making process [37]. This classifier's ability to handle various features and its propensity to identify crucial patterns holds promise for capturing the intricate behaviors associated with heap-based overflow vulnerabilities.

In contrast, the Naive Bayes classifier, while based on a simplifying assumption of feature independence, showcases exceptional performance in handling high-dimensional data – a key characteristic of malware detection scenarios [38]. Its probabilistic approach and ability to estimate the likelihood of a sample being malware based on observed features make it a robust tool for assessing classification confidence. The Simple Logistic classifier, derived from logistic regression, bridges the gap between simplicity and performance [39]. With a linear modelling approach, it adeptly captures linear feature relationships, which are relevant in heap-based overflow vulnerability identification. Together,

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

this ensemble of classification algorithms – J48, Naive Bayes, and Simple Logistic – fortifies the research's methodology, leveraging each algorithm's strengths to enhance the comprehensive detection of malware exploiting heap-based overflow vulnerabilities.

- J48 Classifier: By implementing the J48 classification algorithm with 10-Fold cross-validation, we get an accuracy rate of 84.466%; with the train/test validation technique, we get an accuracy of 90.29% with the J48 classifier.
- Naive Bayes: By implementing the Naive Bayes classification algorithm, we get an accuracy rate of 83.4951% with 10-Fold validation and 84.466% with the train/test set technique.
- Simple Logistic: By implementing a Simple Logistic classification algorithm. We get an accuracy rate of 84.466% with the 10-fold validation technique and 85.436% with the train/test split technique.

B. Evaluation Metrics

To rigorously assess the effectiveness of our proposed methodology, we employ a set of key metrics that elucidate the classification performance, particularly concerning heap-based overflow vulnerability exploitation. These metrics provide insights into the accuracy and reliability of our system in distinguishing between malicious executables that exploit heap-based overflow vulnerabilities and benign programs. The following evaluation metrics are considered:

- True Positives (TP): This metric represents the count of malware samples that exploit heap-based overflow vulnerabilities and are correctly classified as malicious executables.
- False Positives (FP): It accounts for the number of benign programs misclassified as heap-based overflow malware. These instances are incorrectly labelled as malicious when they are not.
- Accuracy: The accuracy metric quantifies the ratio of correctly classified results to total results, providing an overall measure of the system's classification correctness.

These metrics are fundamental indicators of the system's performance and ability to accurately identify and differentiate between malware that exploits heap-based overflow vulnerabilities and benign software. A comprehensive analysis of these metrics is presented in the subsequent sections to substantiate the efficacy of our approach.

C. Experimental Protocol and Results:

To validate the performance of our proposed system, two validation techniques are implemented: a 10-fold cross-validation and a train/test split. Cross-validation computes the accuracy of the implemented model by dividing the dataset into multiple training and testing sets. J48, Naïve Bayes and simple logistic classification models are trained on the training set. Their accuracy is calculated based on their performance on the testing set.

- 10-Fold Cross Validation: Three classifiers, J48, Naïve Bayes, and Simple Logistic, are trained and tested with 10-fold cross-validation, i.e., the created dataset is divided arbitrarily into ten subsets, where one subset is used for testing and nine for training. For every combination, the process is repeated ten times. This procedure aids in assessing the strength of a given approach to detect malware that exploits heap-based overflow vulnerability without any previous information.
- Train/Test Validation: Three classifiers, J48, Naïve Bayes, and Simple Logistic, are trained and tested using the training set validation, i.e., the complete dataset we used for learning is also used for testing purposes. This approach gives better results than other technique that exploits heap-based overflow vulnerability.

Table 2. Train/tests split validation results.

	J48	**Naive Bayes**	**Simple Logistic**
Accuracy Rate	90.29%	84.466%	85.436%
True Positive (TP)	0.903	0.845	0.854
False Positive (FP)	0.198	0.345	0.323

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

Table 2 shows the training set validation results and Table 3 shows the 10-Fold cross-validation results. It can be seen how different machine learning algorithms are used for the performance evaluation of the proposed approach in terms of accuracy rate, true positive and false positive.

Table 3. 10-Fold cross-validation results.

	J48	Naive Bayes	Simple Logistic
Accuracy Rate	84.466%	83.495%	84.466%
True Positive (TP)	0.845	0.835	0.845
False Positive (FP)	0.310	0.349	0.345

Table 4 shows the comparison between 10- fold cross validation and percentage split based on accuracy.

Table 4. Summary of obtained results

	J48	Naive Bayes	Simple Logistic
Accuracy Rate (10-Fold cross-validation)	84.466%	83.4951%	84.466%
Accuracy rate (percentage split)	76.191%	76.191%	85.714%
Use training set	90.290%	84.466%	85.436%

VI. DISCUSSION

Data mining classification algorithms are employed to categorize newly acquired data into pre-defined groups. These algorithms utilize a previously classified dataset to classify new data based on current trends and patterns. Once the rules are generated from the implemented algorithm, the logic can be incorporated into various intrusion detection technologies, such as firewalls and IDS signatures. The results are not dependent on any single feature, as the algorithm validates multiple features even if a particular feature is absent in real-world malware scenarios. The dataset is designed with this issue in mind and is addressed during result generation.

The classification algorithms employed in this study assume a pivotal role in categorizing newly acquired data into predefined clusters. These algorithms are contingent upon a pre-existing dataset that has undergone prior classification, which is the foundation for identifying and categorizing novel data based on prevailing trends and patterns. This iterative process entails deriving decision rules by the implemented algorithm, with the resultant logic amenable to integration into a spectrum of intrusion detection technologies, encompassing firewalls and Intrusion Detection System (IDS) signatures. Notably, our approach's efficacy transcends sole reliance on any individual feature. The algorithm rigorously evaluates multiple features, even when features are absent within real-world malware scenarios. This aspect has been meticulously considered in the design of our dataset and is explicitly addressed during the result-generation phase.

Among the three implemented classifiers, the J48 algorithm exhibited the highest accuracy rate of 90.29% when using the training set technique, in which the entire dataset is used for training and testing. The false positive rate was also low at 84.466% in the 10-fold cross-validation technique, where both the J48 and simple logistic classifiers performed well in terms of accuracy. In the 10-fold cross-validation technique, the dataset is divided into ten equal parts, and each part is utilized for training and testing in various combinations. This process is repeated ten times, and a weighted average is computed at the end. This procedure helps evaluate the approach's effectiveness in detecting malware that exploits heap-based overflow vulnerability without relying on previous information.

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

VII. LIMITATIONS

Our proposed approach has some limitations. Firstly, the exploitation of relatively simple classifiers for the malware detection task. While our chosen classifiers, namely J48, Naïve Bayes, and simple logistics, demonstrated commendable accuracy in our evaluation, their effectiveness might be constrained when confronted with more complex and sophisticated malware variants. Modern malware has exhibited increasing obfuscation and polymorphism, often designed to evade traditional detection methods. By employing straightforward classifiers, our methodology may struggle to classify such intricate and advanced malware strains accurately. To address this limitation, in future research, we aim to explore integrating more advanced machine learning techniques, possibly incorporating deep learning approaches, to enhance the model's ability to identify evasive and intricate malware threats—and secondly, the size of the dataset used for evaluation. While our proposed detection technique exhibited promising results in terms of accuracy, it was assessed on a comparatively smaller dataset. This reduced dataset size might not fully capture the diversity, variability, and behavior of real-world malware samples. Consequently, the generalizability of our approach to handling a broader range of malware instances could be limited. Future endeavors should gather larger and more diverse datasets encompassing a wider spectrum of malware families, strains, and attack scenarios. This expansion in dataset scope could provide a more accurate representation of the challenges faced in real-world malware detection scenarios and allow for a more comprehensive evaluation of our methodology's performance across many cases.

VIII. CONCLUSION

The increasing prevalence of polymorphic variants and new malware families has driven the Anti-Malware industry to develop automated tools for classifying malware based on their potential to exploit vulnerabilities. Our research introduces a behavioral detection technique to identify malware that exploits heap overflow vulnerabilities. By implementing our proposed system, we have established a comprehensive detection method for classifying malware that leverages heap-based overflow vulnerabilities. The logical framework derived from our algorithm enhances understanding of Advanced Persistent Threat (APT) strategies and bolsters overall security for organizations. Our work combines data mining and reverse engineering techniques to construct a malware detection system. We have devised a system that extracts features from binary files to detect malware exploiting vulnerabilities. We evaluated the performance of our proposed scheme using machine learning algorithms for result generation and achieved a high accuracy rate during validation against a training set and 10-fold cross-validation. Among the three classifiers employed (J48, Naïve Bayes, and simple logistics), J48 achieved the highest accuracy with 90.29% in training set validation and 84.466% in 10-fold cross- validation. The suggested methodology is easily implementable in cybersecurity operations, offering insights into the behavior of malware targeting an organization.

REFERENCES

[1] Katzenbeisser, S.; Kinder, J.; Veith, H. Malware Detection., 2011.Ståhlberg, M. Malware detection, 2011. US Patent App. 12/462,913.
[2] Ullah, S.; Mahmood, Z.; Ali, N.; Ahmad, T.; Buriro, A. Machine Learning-Based Dynamic Attribute Selection Technique for DDoS Attack Classification in IoT Networks. Computers 2023, 12. https://doi.org/10.3390/computers12060115.
[3] Khalid, O.; Ullah, S.; Ahmad, T.; Saeed, S.; Alabbad, D.A.; Aslam, M.; Buriro, A.; Ahmad, R. An Insight into the Machine- Learning-Based Fileless Malware Detection. Sensors 2023, 23. https://doi.org/10.3390/s23020612.
[4] Joshi, S.; Upadhyay, H.; Lagos, L.; Akkipeddi, N.S.; Guerra, V. Machine learning approach for malware detection using random forest classifier on process list data structure. In Proceedings of the Proceedings of the 2nd International Conference on Information System and Data Mining, 2018, pp. 98–102.
[5] Smith, M.R.; Johnson, N.T.; Ingram, J.B.; Carbajal, A.J.; Haus, B.I.; Domschot, E.; Ramyaa, R.; Lamb, C.C.; Verzi, S.J.; Kegelmeyer, W.P. Mind the gap: On bridging the semantic gap between machine learning and malware analysis. In Proceedings of the Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, 2020, pp. 49–60.
[6] Gashi, I.; Sobesto, B.; Stankovic, V.; Cukier, M. Does malware detection improve with diverse antivirus products? an empirical study. In Proceedings of the International Conference on Computer Safety, Reliability, and Security. Springer, 2013, pp. 94–105.
[7] Idika, N.; Mathur, A.P. A survey of malware detection techniques. Purdue University 2007, 48, 2007–2.
[8] Buriro, A.; Buriro, A.B.; Ahmad, T.; Buriro, S.; Ullah, S. MalwD&C: A Quick and Accurate Machine Learning-Based Approach for Malware Detection and Categorization. Applied Sciences 2023, 13. https://doi.org/10.3390/app13042508.
[9] Gorment, N.Z.; Selamat, A.; Cheng, L.K.; Krejcar, O. Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges and Future Directions. IEEE Access 2023, pp. 1–1. https://doi.org/10.1109/ACCESS.2023.3256979.

Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 11. Issue. 1, PP. 1-13, November 2023
https://doi.org/10.22555/pjets.v11i1.984
E-ISSN 2224-2333
P-ISSN 2222-9930

[10] Zolotukhin, M.; Hämäläinen, T. Detection of zero-day malware based on the analysis of opcode sequences. In Proceedings of the IEEE 11th Consumer Communications and Networking Conference (CCNC). IEEE, 2014, pp. 386–391.
[11] Markel, Z.; Bilzor, M. Building a machine learning classifier for malware detection. In Proceedings of the 2014 second workshop on anti-malware testing research (WATeR). IEEE, 2014, pp. 1–4.
[12] Xu, Z.; Ray, S.; Subramanyan, P.; Malik, S. Malware detection using machine learning based analysis of virtual memory access patterns. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2017, pp. 169–174.
[13] Chen, K.; Zhang, H.; Feng, X.; Zhang, X.; Mi, B.; Jin, Z. Backdoor attacks against distributed swarm learning. ISA Transactions 2023. https://doi.org/https://doi.org/10.1016/j.isatra.2023.03.034.
[14] Firdausi, I.; Erwin, A.; Nugroho, A.S.; et al. Analysis of machine learning techniques used in behavior-based malware detection. In Proceedings of the 2010 second international conference on advances in computing, control, and telecommunication technologies. IEEE, 2010, pp. 201–203.
[15] Ranveer, S.; Hiray, S. Comparative analysis of feature extraction methods of malware detection. International Journal of Computer Applications 2015, 120.
[16] Kilgallon, S.; De La Rosa, L.; Cavazos, J. Improving the effectiveness and efficiency of dynamic malware analysis with machine learning. In Proceedings of the Resilience Week (RWS). IEEE, 2017, pp. 30–36.
[17] Santos, I.; Devesa, J.; Brezo, F.; Nieves, J.; Bringas, P.G. Opem: A static-dynamic approach for machine-learning-based malware detection. In Proceedings of the International joint conference CISIS'12-ICEUTE´ 12-SOCO´ 12 special sessions. Springer, 2013, pp. 271–280.
[18] [Chowdhury, R.a. Protecting data from malware threats using machine learning technique. In Proceedings of the 12th IEEE Conference on Industrial Electronics and Applications. IEEE, 2017, pp. 1691–1694.
[19] Chowdhury, M.; Rahman, A.; Islam, R. Malware analysis and detection using data mining and machine learning classification. In Proceedings of the International Conference on Applications and Techniques in Cyber Security and Intelligence. Springer, 2018, pp. 266–274.
[20] Joshi, S.; Upadhyay, H.; Lagos, L.; Akkipeddi, N.S.; Guerra, V. Machine learning approach for malware detection using random forest classifier on process list data structure. In Proceedings of the Proceedings of the 2nd International Conference on Information System and Data Mining, 2018, pp. 98–102.
[21] Willems, C.; Holz, T.; Freiling, F. Toward automated dynamic malware analysis using cwsandbox. IEEE Security & Privacy 2007,5, 32–39.
[22] Loi, H.; Olmsted, A. Low-cost detection of backdoor malware. In Proceedings of the 12th International Conference for Internet Technology and Secured Transactions (ICITST). IEEE, 2017, pp. 197–198.
[23] Qamar, A.; Karim, A.; Chang, V. Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems 2019, 97, 887–909. https://doi.org/https://doi.org/10.1016/j.future.2019.03.007.
[24] Asmitha, K.; Vinod, P. A machine learning approach for linux malware detection. In Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE, 2014, pp. 825–830.
[25] Yang, L.; Chen, Z.; Cortellazzi, J.; Pendlebury, F.; Tu, K.; Pierazzi, F.; Cavallaro, L.; Wang, G. Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023, pp. 719–736.
[26] Severi, G.; Meyer, J.; Coull, S.; Oprea, A. Explanation-Guided backdoor poisoning attacks against malware classifiers. In Proceedings of the 30th USENIX security symposium (USENIX security 21), 2021, pp. 1487–1504.
[27] Xu, X.; Wang, Q.; Li, H.; Borisov, N.; Gunter, C.A.; Li, B. Detecting ai trojans using meta neural analysis. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 2021, pp. 103–120.
[28] O'Kane, P.; Sezer, S.; McLaughlin, K.; Im, E.G. SVM training phase reduction using dataset feature filtering for malware detection. IEEE transactions on information forensics and security 2013, 8, 500–509.
[29] Vemparala, S.; Di Troia, F.; Corrado, V.A.; Austin, T.H.; Stamo, M. Malware detection using dynamic birthmarks. In Proceedings of the Proceedings of the 2016 ACM on international workshop on security and privacy analytics, 2016, pp. 41–46.
[30] Dhalaria, M.; Gandotra, E. MalDetect: A classifier fusion approach for detection of android malware. Expert Systems with Applications 2024, 235, 121155. https://doi.org/https://doi.org/10.1016/j.eswa.2023.121155.
[31] Youssef, A.; Abdelrazek, M.; Karmakar, C. Use of Ensemble Learning to Detect Buffer Overflow Exploitation. IEEE Access 2023,11, 52009–52025. https://doi.org/10.1109/ACCESS.2023.3279280.
[32] Tu, H. Boosting Symbolic Execution for Heap-based Vulnerability Detection and Exploit Generation. In Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2023, pp. 218–220. https://doi.org/10.1109/ICSE-Companion58688.2023.00059.
[33] Oktavianto, D.; Muhardianto, I. Cuckoo malware analysis; Packt Publishing Ltd, 2013.
[34] GÜVEN, E.Y.; et al. Mirai Botnet Attack Detection in Low-Scale Network Traffic. Intelligent Automation & Soft Computing 2023, 37.
[35] Kulkarni, E.G.; Kulkarni, R.B. Weka powerful tool in data mining. International Journal of Computer Applications 2016, 975, 8887.
[36] Quinlan, J.R. C4.5: Programs for machine learning; Morgan Kaufmann, 1993.
[37] Domingos, P.; Pazzani, M. On the optimality of the simple Bayesian classifier under zero-one loss. Machine learning 1997, 29, 103–130.
[38] Nelder, J.A.; Wedderburn, R.W. Generalized linear models. Journal of the Royal Statistical Society: Series A (General) 1972, 135, 370–384.