Enterprise Malware Detection using Digital Forensic Artifacts and Machine Learning

MATHIEU DROLET, VINCENT ROBERGE
Electrical and Computer Engineering,
Royal Military College of Canada,
13 General Crerar Crescent, Kingston, ON,
CANADA

Abstract: - Malware detection is a complex task. Numerous log aggregation solutions and intrusion detection systems can help find anomalies within a host or a network and detect intrusions, but they require precise calibration, skilled analysts, and cutting-edge technology. In addition, processing host-based data is challenging, as every log, event, and configuration can be analyzed. In order to obtain trusted information about a host state, the analysis of a computer's memory can be performed, but obtaining the data from acquisition and performing the analysis can be challenging. To address this limitation, this paper proposes to collect artifacts within a network environment. This approach involves remotely gathering memory-based and disk-based artifacts from a simulated enterprise network using Velociraptor. The data was then processed using three machine learning algorithms to detect the malware samples against regular user activity generated with a user simulation tool for added realism. With this method, Random Forest and Support Vector Machine achieved a perfect classification of 41 malware samples.

Key-Words: - Digital forensics, Host-based monitoring, Machine learning, Malware, Memory forensics, User simulation, Volatility, Velociraptor.

Received: January 6, 2024. Revised: April 11, 2024. Accepted: June 13, 2024. Published: July 9, 2024.

1 Introduction

Detecting malware presents a complex challenge due to its varied forms and ability to target diverse processes, protocols, and devices. Traditionally, three main strategies are utilized to defend against intrusions. The first approach involves identifying malware as it enters the network or computer, with significant research focusing on detecting files downloaded from suspicious URLs or when users browse malicious websites. However, it is possible for URLs to be altered to appear less suspicious or to avoid blacklisted domains. A second strategy involves monitoring activities within workstations, such as tracking system calls in the operating system and identifying launched processes and their parent processes. This method often incorporates antivirus software, which compares specific strings and binary patterns against a database of known malicious software. However, some types of malware, such as rootkits, can affect the normal operation of the OS and hide from antivirus software, which makes detection difficult. The last strategy consists of using digital forensics, which focuses on compromised computer hard drive and memory, post-exploitation, to better understand how a specific malware works. This allows for signatures to be developed so subsequent occurrences of the intrusion can be detected. Digital forensics is advantageous because it can provide more context on the state of the machine when the disk or the memory capture was collected. Files that have been deleted or hidden can often be retrieved, thus leading to malware analysis and a deeper understanding of the adversary and its intent. However, this technique is limited by the time required by a skilled analyst to perform it.

Currently, host-based monitoring solutions, such as Endpoint Detection and Response (EDR) tools and antivirus software, use agents, that are running on different endpoints, and all report back to a centralized server. Collating this data with network-based tools remains a challenging task and it can be difficult for network defenders to maintain situational awareness. Additionally, EDR tools and antivirus programs can generate numerous alerts due to their ability to report on a wide range of events and logs. Even though EDRs may lack the capacity for the extensive behavioral analysis needed to identify new threats, they are adept at detecting known threats, which constitute the majority faced by enterprise networks, [1]. Security Operation Centers (SOC) often rely on supplementary tools like threat-hunting feeds and indicators of compromises.

Establishing an accurate behavior-based detection system requires a robust baseline built on sufficient data samples that capture typical network and computer activities. However, the evolving network demands and software usage by individuals present a challenge in maintaining the baseline's effectiveness. The primary focus of current digital forensic approaches is on pinpointing how malware infiltrated a system to aid in implementing mitigating strategies. Nonetheless, the valuable insights obtained from this method are frequently delayed. The analysis of collected artifacts poses a considerable challenge due to the vastness of the dataset. One potential solution to this issue is the application of machine learning to automate digital forensic investigations. While existing research mainly concentrates on extracting forensic artifacts from individual computer memory captures, Virtual Machine (VM) memory, or sandboxes, the applicability of these findings to operational networks can be complex, [2], [3]. To effectively transfer the principles and conclusions of these studies to live networks, the collection and analysis of artifacts need to occur in near real-time. This study seeks to overcome this obstacle by proposing a solution suitable for live enterprise networks.

This research, which is based on a thesis, [4], introduces four key contributions: identifying Velociraptor as a valuable tool for generating features to train a machine learning model for effective malware detection, developing a methodology to produce data using a user simulation tool, pinpointing features capable of identifying malware presence on an active computer and comparing three machine learning algorithms in the context of malware detection.

2 Previous Works

Machine learning has proven to be a valuable tool in various aspects of computer security. In [5], research is conducted in network security, focusing on the identification of anomalous network traffic through the use of the Isolation Forest algorithm. Their goal was to detect covert channels, malware usage, and other anomalies within the network. Host-based analysis researchers such as in [6] have used machine learning to analyze features collected from hosts, such as system traces. In [7], the authors proposed an anomaly detection approach that used Isolation Forest and K-means for real-time anomaly detection using the network traffic logs. In [8], a Deep Convolutional Neural Network is used to detect intrusion. Their proposed algorithm achieved better results compared to other current Intrusion Detection System (IDS) implementations, such as Deep Belief Network, while reducing the processing time. They achieved an F1-Score of 0.97 to 0.98, depending on the type of tested network attacks.

Machine learning has also been used for digital forensic research. In their disk-based forensic research, the authors of [9] looked at machine-learning techniques for file system forensic analysis. They aimed at detecting modified files to assist in timeline reconstruction. Multiple memory forensic researchers such as in [10], [11] and [12] have used machine learning to automate and assist in the detection of malware. In [10], the authors used machine learning and artifacts found in memory using Volatility to detect ransomware and Remote Access Trojan (RAT) in a cloud computing server, hosting hundreds of virtual machines (VM). Using Volatility plugins, information found in memory, such as running processes, the services and DLLs could be retrieved, and features generated to feed into a machine learning algorithm. Their approach had the advantage of enabling detection of fileless malware, which does not have a presence on the disk. They used VMware's vSphere infrastructure to collect snapshots of the VMs, then extracted the memory capture from the snapshot files so it could be analyzed using Volatility. They used a baseline of 100 snapshots taken at ten-minute intervals, with 100 more snapshots taken for each of the nine ran programs, benign and ransomware. They used nine machine learning algorithms for their datasets: J48, Random Forest (RF), Naïve Bayes (NB), Bayesian Network (BN), Logistic Regression (LR), LogitBoost (LB), Sequential Minimal Optimization (SMO), Bagging, and AdaBoost (AB). Out of their multiple test cases, RF achieves the best overall results; this has also been observed by the authors of [11] who researched kernel-level rootkit using memory forensic and machine learning with a similar methodology, but different features. The authors of [12] performed similar research to detect unknown malware in Linux cloud environments. More recently, the authors of [13] used a similar methodology to detect different types of malware using a custom Volatility plugin to gather specific process data. Their test environment was using a virtual machine from which the memory was collected. In addition, recent work including [14] has been focusing on analyzing the memory capture of potential malware samples using computer vision techniques by converting the executables into RGB images and processing them using machine learning.

In their assessments of various EDR solutions, the authors of [15] tested multiple EDRs, including Carbon Black, CrowdStrike Falcon, F-Secure Elements EDR, McAfee Endpoint Protection, and Symantec Endpoint Protection, in order to assess their effectiveness against Tactics, Techniques, and Procedures (TTPs) employed by Advance Persistent Threat (APT) actors. Their research revealed that none of the EDR solutions were capable of identifying all threats. A significant number of the EDRs examined performed poorly in identifying DLL Side-loading [15]. EDR solutions are now incorporating more machine learning algorithms and processing capabilities in an effort to enhance detection rates while also keeping false positives at a minimum and identifying malware at an earlier stage in the cyber kill chain [16] before significant damage occurs. The primary focus for EDR vendors is to identify effective features for evaluation while ensuring a minimal impact on the host system and requiring limited bandwidth, in addition to effectively managing and processing vast amounts of data [15]. Conversely, in another study, the authors of [17] concentrate on the utilization of custom tools, like their Python-based Based Tool, for gathering artifacts from Windows hosts.

Current research has certain significant limitations that could be enhanced. The effectiveness of signature-based monitoring is constrained by the information stored in its database, making it more effective in identifying older threats that have already been scrutinized. Furthermore, a few current studies, such as those conducted in [10] and [18], may not be directly applicable in a real-time setting as the collection of memory images necessitates the use of snapshots for analysis. This methodology does not scale effectively with modern systems that have a large memory capacity. Outside of the cloud-computing environment, disk acquisition and memory capture need to be done computer by computer, which creates delays and can require a lot of bandwidth if done remotely. In addition, if sandboxes are used, it is possible to fool them. Some malware can be context-aware and change their execution based on whether they are on a virtual machine or if they suspect they are executing in a sandbox environment using sandbox fingerprinting techniques, [19]. This paper addresses this limitation by using Velociraptor for data acquisition, which is more lightweight than performing a full memory capture, and is applicable to non-virtualized environments.

Limitations presented in this section are addressed by collecting digital forensic evidence using Velociraptor at regular intervals. The Velociraptor offline collector, running on each workstation, can gather information about the system state. Features can then be generated on a dedicated server. This enables the collection of the artifacts from all computers in the network simultaneously and then processing them using a machine learning algorithm. This process is transparent to the user and could be applied to an enterprise network. Using this technique, it is possible to widen the range of artifacts available to host-based detection tools by looking at both volatile and non-volatile digital forensic artifacts. In addition, to provide a more realistic test environment, this paper uses an advanced simulation environment; not simply a virtual machine, but a complete enterprise network with simulated user activity.

3 Background Theory

Evaluation of a machine learning algorithm performance can be achieved through various metrics, including accuracy, precision, recall, and F1-Score. These metrics are based on confusion matrices, which show the frequency of correct and incorrect predictions made by the classifier regarding the null hypotheses, [20]. For instance, when determining whether a data point is malicious, the confusion matrix consists of True Positive (TP) for accurate identification as malicious, True Negative (TN) for accurate identification as non-malicious, False Positive (FP) or Type I error for inaccurate malicious classification, and False Negative (FN) or Type II error for inaccurate benign classification, [20].

Equation 1 illustrates the concept of accuracy, which offers insight into a model's performance, as referenced in [11]. However, relying solely on accuracy as a metric may be limited in its usefulness, as it only signifies the percentage of correctly classified samples. Precision, as depicted in Equation 2 and as referenced in [10] , serves as a measure of how many samples identified as malicious are truly malicious, without considering missed malware samples. A high precision score suggests that minimal normal benign data has been mistakenly classified as anomalous. On the other hand, recall, outlined in Equation 3 and as shown in [11], further refines this assessment by indicating the number of correctly identified malware samples. An ideal classifier, accurately detecting all malware samples, would achieve a recall value of 1, without reflecting the occurrence of FP in the process. The F1-Score, also known as F-measure, and presented in Equation 4 and as seen in [11], merges precision and recall to deliver a more holistic evaluation of the algorithm's overall performance. Computed through the harmonic mean of precision and recall, a high F1-Score signifies elevated levels of precision and recall, [20].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$
$$Precision = \frac{TP}{TP + FP} \quad (2)$$
$$Recall = \frac{TP}{TP + FN} \quad (3)$$
$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \quad (4)$$

The comprehensive machine learning pipeline encompasses all stages from data acquisition to result generation. The process involves gathering raw data, such as logs, and transforming them into a format suitable for processing by machine learning algorithms, typically numerical data. Not all features generated are equally valuable, prompting the need for careful selection of the most relevant ones. Each chosen machine learning algorithm undergoes model training with the data to evaluate its performance. Optimization strategies involve adjusting parameters within the selected algorithm to enhance results. Ultimately, the model is applied to new data for classification or anomaly detection, [21]. The steps of the machine learning pipeline are illustrated in Figure 1.

The following paragraphs will be discussing the three machine learning algorithms used in this paper, Isolation Forest, Random Forest and Support Vector Machines (SVM).

Isolation Forest is a tree-based algorithm that separates all the data points into different nodes of a tree, or splits. This process effectively isolates all data points into different branches of the tree. The more splits required to reach the data point, the more normal the point is determined to be; anomalous data points tend to be easier to isolate and therefore require fewer splits. This results in fewer branches and makes the process of walking back to the top of the tree shorter. The algorithm repeats this process for multiple trees, creating a forest, [22]. The anomaly score, which is a value between 0 and 1, is calculated using Equation 5, as seen in [20], and express the anomaly score of a given data point. In Equation 5, *E(h(x))* is the average path length of a point x in the forest, *c(n)* is the average path length of any given data point in the dataset and *s(x,n)* is the anomaly score of a given data point x. The closer to 1 *s(x,n)* is, the more likely it is to be an anomalous data point, [22].

$$s(x, n) = 2^{-\frac{E(h(x))}{C(n)}} \quad (5)$$

where *E(h(x))* is the average path length of a point *x* in the forest, *c(n)* is the average path length of any given data point in the dataset and *s(x,n)* is the anomaly score of a given data point *x*.

Random Forest is a collaborative group of decision trees in which multiple trees are constructed in a random manner. This enhances the diversity of Random Forest compared to traditional decision tree models. In a typical decision tree, a node is divided based on the optimal feature for each split, whereas Random Forest makes the split based on the best feature from a randomly selected set of features to introduce variety, utilizing the *feature_importances* parameter in SKLearn. The *feature_importances* value, also known as Mean Decrease of Impurity (MDI) importance, is computed for an individual tree using Equation 6, as referenced in [23] . Each feature (X_m) in a given tree *T* in the forest is assigned a score based on the MDI of that particular feature. To obtain a more precise estimation of the impact a specific feature can have on the machine learning model, the average MDI value of each tree is determined using Equation 7, as illustrated in [23] . The prediction made by Random Forest is essentially the prediction that is most prevalent among all the trees in the ensemble. This characteristic enables this algorithm to outperform other tree-based techniques by minimizing the error that could be generated by a single tree, [24].

$$Imp(X_m, T) = \sum_{t \in T : v(s_t) = X_m} p(t) \Delta\, i(s_t, t) \quad (6)$$
$$Imp(X_m) = \frac{1}{N_T} \sum_{T} Imp(X_m, T) \quad (7)$$

SVMs function by attempting to construct a hyperplane that divides the distinct classes of data into distinct regions. Two parallel auxiliary hyperplanes, which intersect the data points nearest to the initial hyperplane, are identified as the support vectors. SVMs have the capability to utilize various kernels in order to establish these hyperplanes. The kernel serves as the mathematical representation of the hyperplane. The formula for the linear SVM kernel, displayed as Equation 8 in Table 1 of [25], is one of the options. Nevertheless, this approach may not always be the most effective. Alternative non-linear approaches, such as the Gaussian Radial Basis function (RBF) depicted in Equation 9 of the same reference, can be employed to better suit the dataset and achieve a more effective classification.

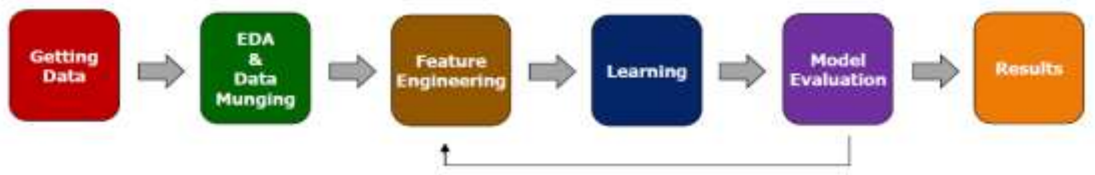

Fig. 1: Machine Learning Pipeline

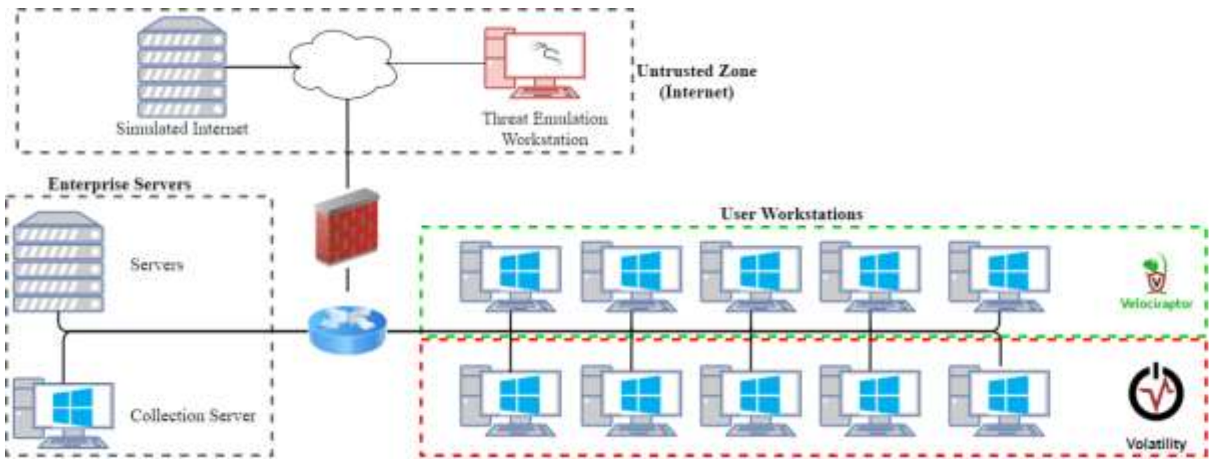

Fig. 2: Test Environment

The RBF kernel stands out as the most widely used SVM kernel according to reference, [25]. It has applications in intrusion detection as well as in linear and malware detection models as stated in the same source.

$$K(x_i, x_j) = x_i^T x_j + 1 \quad (8)$$

$$K(x_i, x_j) = exp^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}} \quad (9)$$

During the process of model training, various methods can be employed to select features. One such technique is Principal Component Analysis (PCA), which is a feature reduction approach grounded in linear algebra. It involves a sequence of orthogonal transformations designed to retain the majority of dataset variance while decreasing its dimensionality, [26].

Feature importance can be assessed using different tree-based algorithms, such as Random Forest using the MDI, or feature_importances. Once a model is fitted using SKLearn functions, the variable feature_importances can be accessed and used to select the best features. The higher the MDI score, the more useful the feature is at predicting the model, [20].

The Predictive Power Score (PPS) is an algorithm introduced in 2020, [27], to facilitate the exploration of a dataset and help find relationships between the different features and data points. It looks at the probability that any given column, which is one of the features of the dataset, can predict the next column, [28]. Using the PPS algorithm, features that bring the most predictability to the model can be selected. The PPS algorithm can be used to perform the feature selection step by only retaining the features of a dataset where the PPS score for the row containing the label is above zero.

A complex problem with machine learning algorithms is the tuning of an algorithm's various hyperparameters. These algorithms have two types of parameters: the parameters, which are determined automatically during the training of an algorithm, and the hyperparameters, which need to be provided to the training method during the training of the algorithms, [29]. A popular method of tuning the algorithms is Grid Search. Its goal is to identify, out of the different possible hyperparameters of an algorithm, the values that will lead to the best prediction while minimizing overfitting, which is when the algorithm is too perfectly tailored to the training set and has difficulties adapting to the test set. If both are very similar, the algorithm may find it difficult to adapt to new data in the future, [30]. A range of values is specified to the algorithm, for each hyperparameter, which makes a grid, and each value is tested exhaustively, [31].

The industry standard tool for memory forensics is Volatility. The book "The Art of Memory Forensics", [32], provides a lot of details about how to use this tool and how it manages to get its data. To conduct an investigation, memory forensics is typically performed by capturing the RAM of a computer before it is shut down. Due to the nature of computer memory, once the computer is powered off, the data in memory is lost and an analyst is no longer able to gather information from it, which prevents the creation of indicators of compromise generated from the analysis of the malware behavior in memory. The analysis of a live system by an analyst gives an opportunity to monitor the system behavior over a longer period of time and take more memory captures if required. However, this can add risk to the network and to the host, if an adversary is exfiltrating data from the computer, shutting it down quickly to avoid further damage may be preferred. Methods to capture the memory include the use of virtual machine snapshots Windows hibernation files, or live-collection software tools such as Dumpit, [32]. Using VM snapshots is trusted as it does not depend on the Windows API to gather its information.

Velociraptor, [33], is an endpoint monitoring and Digital Forensics, Incident Response (DFIR) tool that was designed to collect forensic evidence, monitor events, facilitate enterprise threat-hunting efforts and help security practitioners respond to an incident in an enterprise network. Velociraptor has hundreds of plugins and can be easily extended by writing new custom plugins that are specifically tailored to an analyst's needs, [33]. Velociraptor uses query language similar to the Structured Query Language (SQL), the Velociraptor Query Language (VQL), to analyze the data directly. VQL also allows parsing of the network computers as if they were all part of a database.

4 Methodology

4.1 Data Acquisition

To develop the implemented method, a test environment was created using VMware vSphere. The environment simulates a small company, which has ten employees. The company network used for this research contains the basic infrastructure that would be required for such a company to operate. The company users are being simulated using the Human Actor Like Orchestration (HALO) software, developed by Field Effect Software, [34]. This tool was used in the test environment to add realism and generate background activity within the environment. The network configuration is depicted in Figure 2, illustrating the layout of the environment. Ten Windows 10 workstations, operating on Version 1909 (build 18363.778), were configured for the experiment. To confirm the effectiveness of the implemented method, the workstations were divided into two groups of five each, with one group utilizing Velociraptor for artifact collection and the other employing Volatility for validation purposes.

HALO replicates user behavior to enhance the authenticity of the corporate network, encompassing tasks such as running programs, managing emails, and browsing the web. This platform facilitates the establishment of a timetable for automating user actions. Artifacts from desktops are gathered every half-hour through a scheduled task. The task initiates a series of actions, starting with a batch script that executes the Velociraptor offline collector on the designated five workstations tracked by Volatility. Subsequently, a PowerShell script is activated to transfer all gathered artifacts to the Windows collection server, serving as a central hub for data aggregation and subsequent analysis. Concurrently, memory images are captured at regular intervals from the five monitored workstations, with the plugin results being exported to the Windows server for further processing.

Multiple servers are present in the environment to ensure the proper functioning of the emulated users by HALO. These servers include a domain controller housing the company's DNS server, a file share, a mail server, and a web server. To replicate real-world scenarios, a grey infrastructure was established, which consists of a Grey DNS server serving as the authoritative DNS server for the environment and a simulated Internet. The simulated Internet within the environment comprises numerous scraped websites to enhance the authenticity of the simulation. HALO agents can navigate the simulated web, retrieve actual web pages, and simulate user activity realistically. For the execution of malware, a red infrastructure was necessary, with Kali Linux serving as the attack platform. All malware samples were prepared from this workstation, with the C2 connectivity directed towards it. The initial phase involved collecting a network baseline by observing user activities conducted by the HALO users according to their routines, such as document creation, email correspondence, etc. A total of 1340 samples were collected as a baseline over an 11-day period using the Velociraptor offline collector executable from all five workstations. Subsequently, each malware was executed to generate diverse malware datasets.

After the data collection process, machine learning algorithms were utilized to analyze the data outside of the original environment. This analysis was conducted through the implementation of SKLearn libraries within a Jupyter notebook. The collected data was divided into separate training and testing sets for further evaluation. Subsequently, the classification results of the test set were examined and interpreted. The Velociraptor feature selection approach drew inspiration from various scholarly works, including, [9], [10] and [12], which focused on disk-based and memory forensic investigations. A total of 76 plugins were identified for potential use based on insights gleaned from prior research utilizing tools like Volatility in disk forensic studies. Furthermore, experimental assessments were carried out using the Velociraptor graphical user interface to assess the efficacy of individual plugins in enhancing detection capabilities concerning specific types of malware.

Table 1 displays the number of attributes within each specific domain. In accordance with the methodology outlined by the authors of [3] for classifying attributes, Velociraptor plugins were chosen to achieve a comparable level of host visibility through examination of Registries, DLLs, APIs, and Network-related artifacts. In addition to the four domains identified in their study, two additional domains were incorporated: a file system category encompassing all files-based features generated in a similar fashion [9] and a Windows event category comprising PowerShell events, and remote login events, among others. All chosen Velociraptor attributes align with one of the six domains listed in Table 1.

In the experimental phase, 14 types of malware and frameworks were utilized to produce 41 distinct malware samples. Table 2 outlines the list of malware and tools employed. The subsequent section delves into the functionalities of these programs and their intended impact on the compromised host.

4.2 Exploratory Data Analysis and Data Munging

The information was initially standardized through the application of a min-max scaler. Subsequently, utilizing the *feature_importances* function from SKLearn's Python class for tree-based classifiers, the most impactful features were determined by creating a graphical representation of feature importance to enhance clarity. A selection criterion was applied to the top 20 features as a method for reducing features. Another approach to feature reduction involved utilizing the PPS matrix to identify features that exhibited a correlation with the label. This led to a reduced feature set of 15 features that showed a correlation in predicting the label and would help classify the data. A third feature reduction method was to useuse PCA to retain 95% and 99% of the variance.

4.3 Feature Engineering

During the feature engineering phase, five different versions of the dataset were used to test the performance of the algorithm. Those five testing sets used different features: all the collected features, using the PPS matrix score to keep only the features having a positive correlation with the label, using PCA, to keep 95% and 99% of the variance, and selecting the top 20 features using the feature_importances class. Each machine learning algorithms were tested against each of those five different datasets to assess performance and the impact of the feature reduction process.

Table 1. Type of features generated for Velociraptor

Domain	Features
Registry	24
DLL	7
API	1
Network	13
File System	18
Events	13

Table 2. Malware executables and tools used

Malware Name	Malware Type	Samples
CatfishHTTPSExfiltrator	Data Exfiltration	1
Lyonfish	Ransomware	1
CatfishFileShredder	RAT	1
CatfishSocket1	RAT	1
CatfishExplorer	RAT	1
CatfishPowerShell1	RAT	1
Metasploit	RAT	3
PowerShell Empire	RAT	1
Cobalt Strike	RAT/Persistence/ Credentials Stealing	19
Living Off The Land **[35]**	Persistence	1
CatfishPersister	Credentials Stealing/ Persistence	2
OffensivePH **[36]**	Post-exploitation tool	3
77rootkit **[37]**	Rootkit	1
Hidden **[38]**	Rootkit	5

To conduct cross-validation, the test-train split method from SKLearn was employed ten times to create 10 distinct test sets, enhancing the accuracy of model evaluations. The datasets were divided equally, with half allocated to the training set and the other half to the test set, given the varied types of malware samples and resulting artifacts. This partitioning was necessitated by the limited availability of only 41 malware collections, resulting in approximately 20 to 21 samples per set.

4.4 Model Learning and Evaluation

After data collection, the machine learning models underwent training with Isolation Forest, Random Forest, and SVM algorithms. Each algorithm was utilized to create 11 different sets of trained models through three tuning methods: default parameters of the algorithm, Grid Search, and a manual exhaustive implementation based on Grid Search with certain manually chosen values. Furthermore, all algorithms were trained with each feature selection method.

The specific combinations of feature selection and tuning methods employed in this study can be found in Table 3. The default parameters were only utilized once to compare results with all features included.

4.5 Validation

In order to validate both the outcomes and the methodology, the approach of utilizing Velociraptor and machine learning was compared to the validation technique involving Volatility and machine learning. The research utilizing Volatility closely resembles the implemented method. The validation technique was devised by replicating an experiment carried out in previous studies; characteristics from [10], [11] and [12] were utilized to create the validation dataset. For the implemented method, which utilizes Velociraptor, to be deemed successful, it needed to surpass the performance of the Volatility method. The same malware samples were employed, and the gathering of artifacts with Volatility was conducted using Dumpit and a Volatility executable, at the same time interval as the implemented method. A key advantage of the implemented method is that Velociraptor is significantly more efficient and does not necessitate a memory capture.

Table 3. Trained Machine Learning model combination

Features Selection	Tuning Method
All features	Default parameters Grid Search Exhaustive
Positive PPS correlation	Grid Search Exhaustive
PCA with 95% variance retained	Grid Search Exhaustive
PCA with 99% variance retained	Grid Search Exhaustive
Top 20 features with feature_importances	Grid Search Exhaustive

5 Results

The findings indicate that the implemented approach utilizing Velociraptor and machine learning is more efficient in malware detection compared to the use of Volatility and machine learning. In this section, the two methodologies are defined as Velociraptor and Volatility, respectively.

Each model underwent testing using the test dataset. The model evaluation was repeated ten times, with different random splits between training and testing sets, in order to achieve a more precise outcome; the performance could vary based on the selection of malware for the training set. The results from the model with the highest F1-score for each methodology and algorithm are presented in

Table 4.

The standard deviation of the F1-Scores between each of these ten runs of the algorithm is also displayed in

Table 4. SVM emerged as the most effective and consistent model, with a standard deviation of 0.008%; there was only one false positive in two test cases.

Isolation Forest did not prove to be the most successful algorithm for both methods. The Velociraptor methodology yielded a lower F1-Score compared to Volatility but achieved a higher recall. In the Velociraptor method, the model that demonstrated the best performance utilized all features with manual exhaustive tuning. With Random Forest, the optimum models for both methods successfully detected all malware instances. For Velociraptor, all models, with the exception of those using PCA, achieved a flawless or nearly flawless classification, while the four PCA-based models detected none of the malware. Utilizing SVM, the optimal models for the Velociraptor method attained a perfect classification, whereas the optimal models for the Volatility method were able to identify the majority of the malware samples.

Both the Random Forest and SVM algorithms utilizing Velociraptor characteristics exhibited instances of flawless classification or very poor classification. Possible reasons for this variability include a class imbalance in the dataset, where there were significantly fewer malicious samples compared to benign samples for both Velociraptor and Volatility. This imbalance, with malicious to benign sample ratios of 0.031 and 0.015 respectively, could have influenced the classification results. To address this issue, two common techniques are suggested: under-sampling the majority class and oversampling the minority class, [39]. Additionally, mixing baseline samples with malware samples during training may have impacted the outcomes. Future studies should explore training methods with different datasets, potentially focusing solely on baseline data. Overfitting is also a concern with perfect classification results, and alternative approaches such as employing bagging or boosting techniques through algorithms like bagging trees or AdaBoost can help mitigate this issue.

When it comes to validation, the method incorporating Velociraptor and machine learning surpassed the validation method using Volatility for both Random Forest and SVM. This approach is noted for being more efficient and less resource-intensive, as Velociraptor eliminates the need for memory capture and reduces the need for memory structure scanning.

The detection rates for each algorithm utilizing Velociraptor in the proposed method are detailed in Table 5. The optimal models for Random Forest and SVM resulted in a 100 percent detection rate, however, Isolation Forest exhibited shortcomings in detecting certain types of malware, such as CatfishPersister operating at the user-level privilege, data exfiltration attempts, and some RATs samples like CatfishPowershell1, CatfishExplorer, Cobalt Strike process injection, and screenshot capture.

Table 6 presents the validation method results which closely align with findings from similar studies. The outcomes obtained from the validation method, particularly in experiments with RATs and Ransomware conducted by the authors of [10] as well as [11], are compared. This study demonstrated a higher recall value compared to previous research [10] and [11] and a slightly lower F1-Score. These results validate the effectiveness of this study's validation method utilizing Volatility and its ability to evaluate the performance of the implemented method with Velociraptor.

6 Conclusion

This paper introduced an innovative approach for identifying malware present in a corporate network by utilizing digital forensics artifacts gathered through Velociraptor and analyzed through machine learning. This paper presented a novel methodology to detect malware within an enterprise network using digital forensics artifacts collected using Velociraptor and analyzed using machine learning. A total of 41 malicious samples and 1340 benign samples were tested against three machine learning algorithms. It was determined that Random Forest and SVM were the most effective classifiers for the used dataset, detecting all malicious samples on all occurrences with no or minimal false positives, with an F1-Score of 1.0 for both algorithms and with a minimal standard deviation between the test occurrences. In order to validate this work, this method was compared to a validation methodology based on [10], [11], and [12] using artifacts collected with the memory forensic tool Volatility and machine learning at detecting malware in an enterprise network. The method proposed in this paper achieved the best results, which validated both work and methodology and showed that Velociraptor is an effective tool for this domain of research.

This paper solves the limitation of live forensics data collection. Previous methods relying on memory forensics required VMs to be suspended to collect the memory image. The proposed method addresses this by performing the data collection using the Velociraptor Offline collector, which can also collect some volatile data typically recovered using memory forensics. This method is more lightweight and does not require the suspension of the user workstation. In addition, it enables the collection of live data from an enterprise network and can be applicable to non-virtualized environments. It also enables a faster incident to investigation time delay as data can be processed rapidly once the model has been trained.

Table 4. Comparison of Volatility and Velociraptor methodologies mean results, for each algorithm, after ten occurrences

Method	ML	Acc.	Recall	F1-Score	Std. Dev.
Volatility	IF	0.978	0.390	0.525	0.122
Velociraptor	IF	0.973	0.637	0.418	0.065
Volatility	RF	0.998	1.000	0.919	0.056
Velociraptor	RF	1.000	1.000	1.000	0.017
Volatility	SVM	0.997	0.955	0.879	0.082
Velociraptor	SVM	1.000	1.000	1.000	0.008

Table 5. Velociraptor malware type detection, by algorithms

		Sample Detected		
Malware Type	Number of Malware	IF	RF	SVM
Credentials Stealing	2	1	2	2
Data Exfiltration	1	0	1	1
Persistence	7	7	7	7
Post-exploitation tool	3	3	3	3
RAT	21	17	21	21
Ransomware	1	1	1	1
Rootkit	6	6	6	6

Table 5. Comparison of the results obtained with Volatility with the works from [10] and [11]

Features	Malware Type(s)	ML	Recall	F1-Score
This Paper	Multiple	RF	1.000	0.919
From **[10]**	Ransomware	RF	0.923	0.924
From **[10]**	RAT	RF	0.927	0.947
From **[11]**	Rootkit	RF	0.984	0.986

This paper contributes significantly in four key areas. Firstly, it recognizes Velociraptor as a powerful tool for generating features to train a machine-learning model for malware detection. Secondly, it presents an effective approach for data generation using a user simulation tool, HALO. Thirdly, it identifies features that can identify malware presence on an active computer. Lastly, it compares the performance of three machine learning algorithms in the context of malware detection.

In future works, the Velociraptor server could be utilized in a live setting to monitor and identify threats, replacing the Velociraptor offline collector. This would enhance the speed of defensive actions by network defenders, facilitating direct response actions such as quarantining or terminating malicious processes on infected hosts. Although the Velociraptor Server is not tailored for automated analysis, a bespoke program leveraging the Velociraptor API would be necessary for querying and processing features. The detection model could still be trained using the methodology outlined in this paper. Furthermore, to enhance the model's effectiveness, conducting experiments with a larger array of malware samples through acquiring more binary data or data augmentation methods is recommended.

In terms of future research, alternative algorithms could be explored for analyzing similar data. While Random Forest and SVM yielded positive outcomes in this study, future investigations could delve into the utilization of deep learning algorithms like neural networks. With Velociraptor being a dynamically evolving and customizable tool, the creation of new plugins for detecting specific types of malware could be considered.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used myessaywriter.ai in order to improve the readability and language of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] Securus360, 'EDR is great, but its Limitations can leave you open to Cyberattacks', [Online]. https://www.securus360.com/blog/edr-is-great-but-its-limitations-can-leave-you-open-to-cyberattacks (Accessed Date: April 8, 2022).

[2] Periyadi, G. A. Mutiara, and R. Wijaya, 'Digital forensics random access memory using live technique based on network attacked', in *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, Melaka, Malaysia, May 2017, pp. 1–6. DOI: 10.1109/ICoICT.2017.8074695.

[3] M. Murthaja, B. Sahayanathan, A. N. T. S. Munasinghe, D. Uthayakumar, L. Rupasinghe, and A. Senarathne, 'An Automated Tool for Memory Forensics', in *2019 International Conference on Advancements in Computing (ICAC)*, Malabe, Sri Lanka, Dec. 2019, pp. 1–6. DOI: 10.1109/ICAC49085.2019.9103416.

[4] M. Drolet, 'Enterprise Malware Detection Using Digital Forensic Artifacts And Machine Learning', M.A.Sc. Thesis, Royal Military College of Canada, Kingston, Ontario, 2022, [Online]. https://espace.rmc.ca/jspui/bitstream/11264/542/1/Thesis_Drolet_Completed.pdf (Accessed Date: April 13, 2024).

[5] D. Spiekermann and J. Keller, 'Unsupervised packet-based anomaly detection in virtual networks', *Computer Networks*, vol. 192, p. 108017, Jun. 2021, DOI: 10.1016/j.comnet.2021.108017.

[6] E. Aghaei and G. Serpen, 'Host-based anomaly detection using Eigentraces feature extraction and one-class classification on system call trace data', *Journal of Information Assurance & Security*, vol. 14, no. 4, p. 11, 2019. arXiv:1911.11284

[7] M. T. R. Laskar, J. Huang, V. Smetana, C. Stewart, K. Pouw, A. An, S. Chan and L. Liu, 'Extending Isolation Forest for Anomaly Detection in Big Data via K-Means', *ACM Trans. Cyber-Phys. Syst.*, vol. 5, no. 4, pp. 1–26, Oct. 2021, DOI: 10.1145/3460976.

[8] R. Mendonça, A. Teodoro, R. Rosa, Renata M. Saadi, D. C. Melgarejo, P. Nardelli and D. Rodríguez, 'Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network', *IEEE Access*, vol. 9, pp. 61024–61034, 2021, DOI: 10.1109/ACCESS.2021.3074664.

[9] R. M. A. Mohammad and M. Alqahtani, 'A comparison of machine learning techniques for file system forensics analysis', *Journal of Information Security and Applications*, vol. 46, pp. 53–61, Jun. 2019, DOI: 10.1016/j.jisa.2019.02.009.

[10] A. Cohen and N. Nissim, 'Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory', *Expert Systems with*

Applications, vol. 102, pp. 158–178, Jul. 2018, DOI: 10.1016/j.eswa.2018.02.039.

[11] X. Wang, J. Zhang, A. Zhang and J. Ren, 'TKRD: Trusted kernel rootkit detection for cybersecurity of VMs based on machine learning and memory forensic analysis', *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 2650–2667, 2019, DOI: 10.3934/mbe.2019132.

[12] T. Panker and N. Nissim, 'Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments', *Knowledge-Based Systems*, vol. 226, p. 107095, Aug. 2021, DOI: 10.1016/j.knosys.2021.107095.

[13] S. Lyles, *M. Desantis, J. Donaldson, M. Gallegos, H. Nyholm, C. Taylor and K. Monteith*, 'Machine Learning Analysis of Memory Images for Process Characterization and Malware Detection', in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Baltimore, MD, USA, Jun. 2022, pp. 162–169. DOI: 10.1109/DSN-W54100.2022.00035.

[14] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, 'Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision', *Computers & Security*, vol. 103, p. 102166, Apr. 2021, DOI: 10.1016/j.cose.2020.102166.

[15] G. Karantzas and C. Patsakis, 'An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors', *JCP*, vol. 1, no. 3, pp. 387–421, Jul. 2021, DOI: 10.3390/jcp1030021.

[16] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, 'Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains', *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 14, 2011.

[17] A. Hariyani, J. Undavia, N. Vaidya, and A. Patel, 'Forensic Evidence Collection From Windows Host Using Python Based Tool', in *2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, Goa, India: IEEE, Oct. 2022, pp. 85–90. DOI: 10.1109/ICCCMLA56841.2022.9989295.

[18] A. M. A. Hameed, M. Daley, and L. Espinosa-Anke, *'A Machine Learning Approach for Memory Forensic Investigation'*, Cardiff University, 2020.

[19] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, 'Spotless Sandboxes: Evading Malware Analysis Systems Using Wear-and-Tear Artifacts', in *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 1009–1024. DOI: 10.1109/SP.2017.42.

[20] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, 2nd ed. Sebastopol, CA: O'Reilly Media, Inc, 2019.

[21] B. Lachine, 'Machine Learning Introduction', Kingston, Ontario, Oct. 13, 2020, [Online]. https://moodle.rmc.ca (Accessed Date: October 13, 2021).

[22] F. T. Liu, K. M. Ting, and Z.-H. Zhou, 'Isolation Forest', in *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, Dec. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[23] A. Sutera, G. Louppe, V. A. Huynh-Thu, L. Wehenkel, and P. Geurts, 'From global to local MDI variable importances for random forests and when they are Shapley values', in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 3533–3543, arXiv:2111.02218 [Online]. https://proceedings.neurips.cc/paper/2021/hash/1cfa81af29c6f2d8cacb44921722e753-Abstract.html (Accessed Date: January 23, 2023)

[24] T. Yiu, 'Understanding Random Forest: How the Algorithm Works and Why it is So Effective', Towards Data Science, [Online]. https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (Accessed: March 23, 2022).

[25] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, 'A comprehensive survey on support vector machine classification: Applications, challenges and trends', *Neurocomputing*, vol. 408, pp. 189-215, Sep. 2020, DOI: 10.1016/j.neucom.2019.10.118.

[26] S. Khalid, T. Khalil, and S. Nasreen, 'A survey of feature selection and feature extraction techniques in machine learning', *Proceedings of 2014 Science and Information Conference, SAI 2014*, London, UK, pp. 372-378, Oct. 2014, DOI: 10.1109/SAI.2014.6918213.

[27] J. H. Moedjahedy and G. Pramudya, 'Student Achievement Classification using Power Predictive Score with Machine Learning', in *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*, Makasar, Indonesia, Oct. 2021, pp. 1–6. DOI: 10.1109/ICORIS52787.2021.9649467.

[28] F. Wetschoreck, 'RIP correlation. Introducing the Predictive Power Score', Towards Data Science, [Online]. https://towardsdatascience.com/rip-correlation-introducing-the-predictive-power-score-3d90808b9598 (Accessed: March 17, 2022).

[29] L. Yang and A. Shami, 'On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice', *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, DOI: 10.1016/j.neucom.2020.07.061.

[30] X. Ying, 'An Overview of Overfitting and its Solutions', *J. Phys.: Conf. Ser.*, vol. 1168, no. 2, p. 022022, Feb. 2019, DOI: 10.1088/1742-6596/1168/2/022022.

[31] P. Liashchynskyi and P. Liashchynskyi, 'Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS', *Computing Research Repository*, vol. abs/1912.06059, Dec. 2019, arXiv:1912.06059, [Online]. http://arxiv.org/abs/1912.06059 (Accessed Date: March 28, 2022).

[32] M. H. Ligh, *The Art of Memory Forensics*. Indianapolis, IN: Wiley, 2014.

[33] Rapid7, 'Velociraptor - Dig deeper!', [Online]. https://docs.velociraptor.app/ (Accessed Date: February 1, 2021).

[34] Field Effect, 'Field Effect: The most sophisticated cyber threat monitoring on the planet, made simple', Field Effect Software Inc., [Online]. https://fieldeffect.com/ (Accessed Date: Marcch 21, 2022).

[35] M. Fischer, 'Living Off The Land'. Apr. 04, 2022, [Online]. https://github.com/bytecode77/living-off-the-land (Accessed Date: April 04, 2022).

[36] R. Ancarani, 'Offensiveph'. Aug. 09, 2021, [Online]. https://github.com/RiccardoAncarani/OffensivePH (Accessed Date: April 4, 2022).

[37] M. Fischer, 'r77 Rootkit'. Nov. 04, 2021, [Online]. https://github.com/bytecode77/r77-rootkit (Accessed Date: November 04, 2021).

[38] J. Kornev, 'Hidden'. Apr. 02, 2022, [Online]. https://github.com/JKornev/hidden (Accessed Date: April 4, 2022).

[39] F. López, 'Class Imbalance: Random Sampling and Data Augmentation with Imbalanced-Learn', Medium, [Online]. https://towardsdatascience.com/class-imbalance-random-sampling-and-data-augmentation-with-imbalanced-learn-63f3a92ef04a (Accessed Date: June 13, 2022).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0 https://creativecommons.org/licenses/by/4.0/deed.en_US