Detection of Malicious Binaries and Executables Using Machine Learning-based Detectors

John Martin M. Ladrido [1] , Lawrence Materum[2]

[1]*De La Salle University, john_martin_ladrido@dlsu.edu.ph*
[2]*De La Salle University*
[2]*Tokyo City University*

Abstract
In digital networks, the most common goal of cybercriminals is to steal high-privilege credentials or valuable data. By obtaining high-privilege credentials, cybercriminals can easily navigate, destroy, or steal an organization's data, such as bank details, personal data, and intellectual properties. With the advent of information technology and operational technology convergence like the Internet of things (IoT), it becomes more critical on protecting the high-privilege credentials as cybercriminals can have the power to control operational technologies such as industrial control systems (ICS) and supervisory control and data acquisition (SCADA). Unfortunately, even with this information, many organizations are easily susceptible to these attacks, especially manufacturing firms. This paper presents how cybercriminals from the Internet can utilize malicious payloads and executables to compromise an organization. The proposed approach also shows how organizations can detect those using an (ML) machine learning-based detection by collecting the malicious executables and binaries used in the attacks. Doing so could help organizations to be equipped with proper knowledge in understanding the underlying attack and, at the same time, implementing their detection mechanism specific to the cybercriminals attacking their network. The results show that the machine learning-based detector can identify the samples, whether malicious or benign.
Keywords: advanced persistent threat, malware, machine learning, client-side attacks, privilege escalation, network lateral movement

I. INTRODUCTION

Today's data communication has evolved from the simple sender-receiver transfer of data into a web of information transmission from one network system to another network system. As the knowledge extracted from data and information becomes potentially more valuable than oil or gold, it is imperative to protect these resources from cybercriminals and adversaries. Aside from data and information, cybercriminals' common targets are industrial control systems and supervisory control and data acquisition, which leverages IoT technology. Microsoft's defense report (Microsoft Report Shows Increasing Sophistication of Cyber Threats, 2020) showed a noticeable shift of sophisticated attacks towards credential harvesting and ransomware. The total volume of attacks to IoT has increased by 35% in the first half of 2020. In addition, according to ZDNet's report using F5's statistics (Cimpanu, 2020), Brute Force or Credential Stuffing is the number one security incident at 41%, followed by Distributed Denial-of-Service (DDoS) at 32%.

On the other hand, according to (*Ransomware Upgrades with Credential-Stealing Tricks,* 2020), Ransomware is now being utilized to steal credentials in commonly used browsers such as Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer. In addition to its web browser attack vector, Ransomware has been upgraded to steal E-mail credentials in Mozilla Thunderbird and Microsoft Outlook.

By having high-privilege credentials to control workstations and servers, cybercriminals and adversaries have the power to sabotage manufacturing firms, power generating organizations, transportation controls, or, worst, nuclear and military weapons. For example, it was not long ago that a malware outbreak in Ukraine resulted in shutting down monitoring systems for radiation levels at the Chernobyl Nuclear power plant, according to ("Another Massive Ransomware Outbreak – or Was It?," 2017).

According to (Heritage, 2019), due to the 4th industrial revolution, cybercriminals and adversaries have more attack vectors nowadays because of information technology and operational technology convergence. The average life cycle of computer systems is around nine years. In comparison, machines and sensors from Internet-of-Things or robotics may last 20 to 30 years, giving cybercriminals and adversaries more time to discover vulnerabilities and exploits for such devices. According to (Lemay et al., 2018), the number of cybercriminals and adversaries is significantly increasing worldwide, making defending harder for small companies and organizations with a limited budget and personnel against cybersecurity threats. As such, cybersecurity is always depicted as an arms race between cybercriminals and security vendors, according to (Mansfield-Devine, 2017), and the battleground would be the companies' or organizations' digital environment. According to (*The 2020 Data Breach Investigations Report – a CSO's Perspective - ScienceDirect*, 2020), the top two reasons why security breaches happen do not have enough time to identify vulnerabilities and attack vectors at 31%, followed by not having the expertise or adequate knowledge to remediate the vulnerabilities found at 21%. Thus, companies and organizations usually do not know how these cyber-attacks formulate from cybercriminals and adversaries and defend themselves from getting compromised.

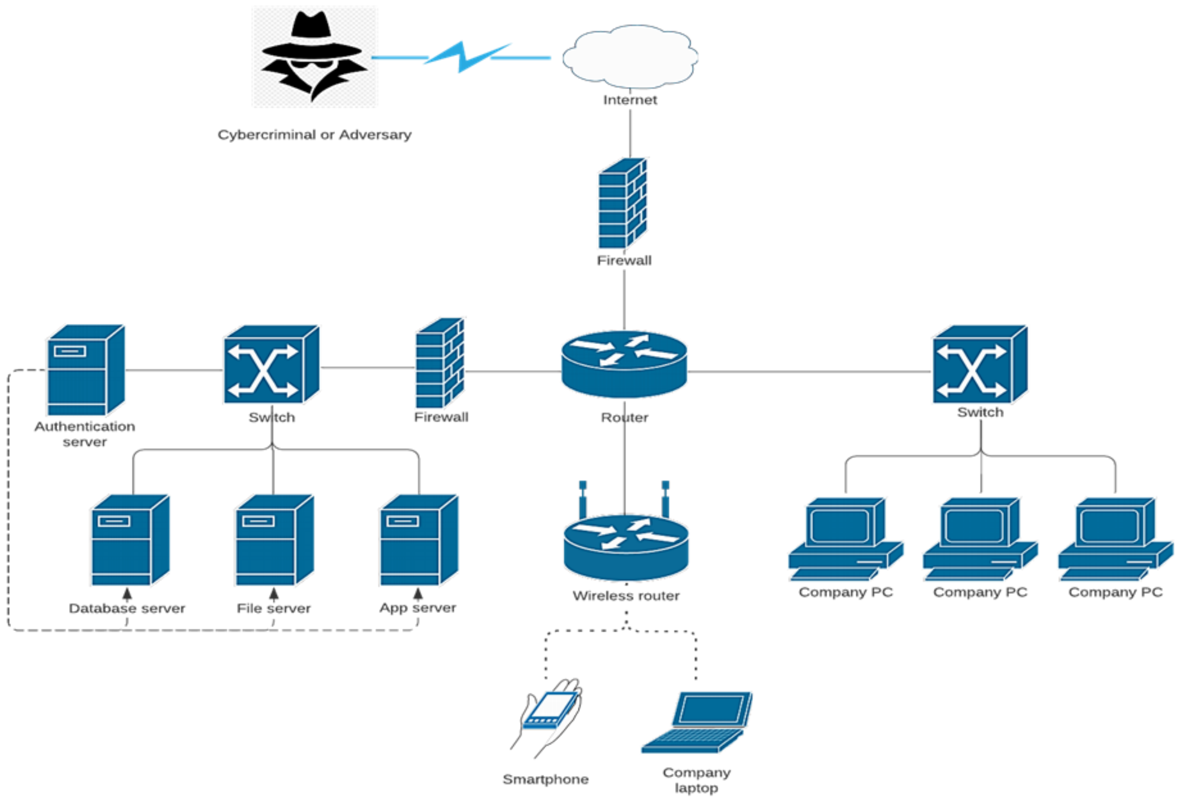

Figure 1. Simple Enterprise Network Architecture

In a company's or organization's network system environment, as shown in Fig. 1, cybercriminals or adversaries usually come from the Internet, and attack vectors start to formulate as companies and organizations open up or try to access the Internet. So, suppose a company or organization does not have any access to the Internet. In that case, the attack vectors that cybercriminals or adversaries could leverage significantly reduce local attacks, usually an insider attack. Thus, cybercriminals and adversaries need to go on-site and connect to the network physically. On the other hand, the top attacks coming from the Internet are Web, E-mail, and file transfer. These attacks usually compromise a workstation or client first before moving to servers. Employees or users using these workstations are easy to attack as human activities tend to give more openings to cybercriminals or adversaries, like browsing malicious websites, downloading and opening malicious files from the Web, E-mail, or file transfer storage. Aside from malicious files, client users in workstations tend to install more applications and drivers than servers, posing a risk if such applications have vulnerabilities or exploit that cybercriminals or adversaries could attack from the Internet.

According to Mitre's Windows Attack Matrix (*Matrix - Enterprise | MITRE ATT&CK®*, 2020), there are nine techniques for initial access or foothold, twelve techniques for privilege escalation, and nine techniques for lateral movement. The severity of initial access attacks remains low as long as the compromised credential is a low privilege user, such as desktop users. However, it becomes severe if the credentials obtained have administrative or system privilege access. On the other hand, privilege escalation severity is always high, as cybercriminals aim to achieve the highest privilege possible. Lateral movement's severity remains low, as long the compromised credentials do not have high or domain administrator access privilege. The most commonly utilized or encountered privileges in these attacks are the Low, Medium, High, and System mandatory level, which defines its integrity level from Low Mandatory Level as the lowest privilege and System as the highest privilege (Deland-Han, 2020).

On the other hand, Table 1 shows a list of attack types identified by the Mitre Attack Framework. General attacks mentioned above are composed of Layer 4 to 7 for the OSI (Open Systems Interconnection) model (Orlowski, 2021) and Layers 3 and 4 in the TCP/IP (Transmission Control Protocol and the Internet Protocol) model (Yuksel & Altunay, 2020). However, once a port and protocol have been chosen as vulnerable for the attack, the attacks mainly depend on the highest layer, the application layer, especially in credential-stealing attacks. According to (*Compromised Credential Attacks Are Frequent and Costly • Designed Privacy*, 2020), the average cost of compromised credentials is 4.77 million US dollars.

The majority of these attack techniques utilize the code execution vulnerabilities in the Microsoft Windows Environment. These code execution vulnerabilities are being utilized by malware to successfully compromised workstations and servers. Data from (*Microsoft : Products and Vulnerabilities*, 2020) shows that the top vulnerability in code execution over the last 20 years. This outcome is twice as much as the second top vulnerability, which is Overflow. Fig. 2 shows the number of Microsoft vulnerabilities per year, and Fig. 3 shows the breakdown

Table 1 Windows Matrix for Initial Access, Privilege Escalation, and Lateral Movement Based on Mitre Attack

Attack Types	Techniques	Severity	Remarks
Initial Access	Drive-by Compromise	Low to High	This attack varies depending on the level of access achieved through initial compromise – most common attacks are the Drive-by compromise, Phishing, and Valid Accounts
	Exploit Public-Facing Application		
	External Remote Services		
	Hardware Additions		
	Phishing		
	Replication Through Removable		

Attack Types	Techniques	Severity	Remarks
	Media		
	Supply Chain Compromise		
	Trusted Relationship		
	Valid Accounts		
Privilege Escalation	Abuse Elevation Control Mechanism	High	As the cybercriminal already has initial access or foothold with low privilege access, privilege escalation aims to achieve high or system privilege. The most common attacks are accessed token manipulation and valid accounts
	Access Token Manipulation		
	Boot or Login Autostart Execution		
	Boot or Login Initialization Scripts		
	Create or Modify System Process		
	Event-Triggered Execution		
	Exploitation for Privilege Escalation		
	Group Policy Modification		
	Hijack Execution Flow		
	Process Injection		
	Scheduled Task/Job		
	Valid Accounts		
Lateral Movement	Exploitation of Remote Services	Low to High	This method varies depending on the level of access being used to navigate through the network and if the destination is a domain server or not. The most common attack is the use of alternate authentication
	Internal Spearphishing		
	Lateral Tool Transfer		
	Remote Service Session Hijacking		
	Remote Services		
	Replication Through Removable Media		
	Software Deployment Tools		
	Taint Shared Content		
	Use Alternate Authentication Material		

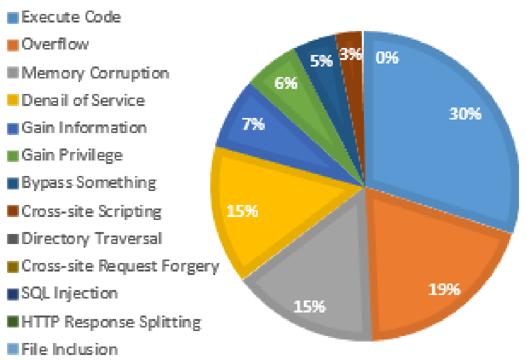

Figure 3. Microsoft Vulnerabilities By Year – Redrawn from [14]

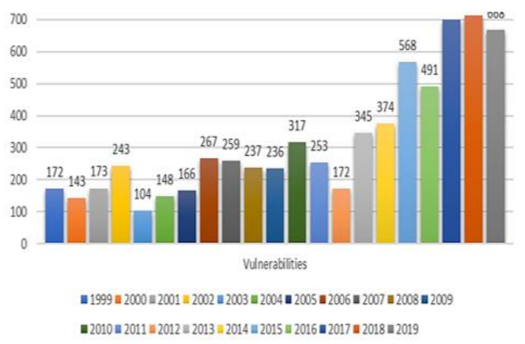

Figure 2. Microsoft Vulnerabity Types Breakdown from 1999 to 2019 – Redrawn from [14]

of vulnerability types in the last 20 years. These vulnerabilities have been weaponized and have been packaged in the form of malicious software or malware.

This paper identified scenarios to address these problems encountered by companies and organizations where an attacker gains access from the Internet until the cybercriminal or adversary steals or achieves credentials with high privilege-level access, and ultimately the Domain Administrator credentials using malicious binaries and executables. This setup can equip or provide knowledge to the person in charge or authority on how these attacks' methods are formulated from a cybercriminal's or adversary's perspective. After the malicious binaries and executables used in common attack techniques have been identified, each binary payloads and executables used in the attacks were evaluated in a third-party scanning engine and, at the same time, used to design and evaluate a machine learning-based detection technique for comparison and to protect these companies or organizations from cybercriminals' and adversaries' attacks and exploits.

II. LITERATURE REVIEW

Existing Work on Detection and Mitigation Methods for Windows Environment Attacks

Based on existing studies, detection and mitigation methods varies from manual configurations, using algorithms, and up to creating traffic visualization. Studies implementing algorithms in detecting malicious software and activity mostly use machine learning in evaluating Windows event viewers. Sample works that are using machine learning in detecting malicious payloads and activities are, (Mikhail et al., 2020), which has installed a Sysmon executable in a host to gather logs and implement Random Forest machine learning algorithm to detect credential dumping, service creation, scheduled task creation, process injection, and Regsvr32 attacks. The result of that study shows that the threshold is directly proportional to the false positive rate. This result means that if a system detects 100% true positive, false positive rate detection increases. Another work is by (Voris et al., 2019), which uses machine learning to monitor user system behavior patterns to detect masquerading or impersonation attacks. The result of (Voris et al., 2019) in Table 2 is similar to (Mikhail et al., 2020) as to accuracy in detecting true positive increases, false-positive increases.

On the other hand, (Matsuda et al., 2018) evaluated Support Vector Machine, Isolation Forest, and Local Outlier Factor machine learning algorithms to detect Golden Ticket and Privilege Escalation attacks with Windows Event logs. Support Vector Machine algorithm yields high precision and accuracy results compared to the other two algorithms shown in Table 2. Next, (Hsieh et al., 2015) used the Markov machine learning model to detect anomalous behavior in Windows Active Directory logins. The result in Table 2 shows that the achievable accuracy is only at 66%. (Hsieh et al., 2015) concluded that this performance in detecting anomalies is due to Active Directory log limitations.

Other works focused on detecting anomalous behaviors or related attacks and do not use machine learning-based algorithms but use simple algorithms such as filtering keywords or strings and event taggings in Windows event viewer are, (Kotlaba et al., 2020) uses Powershell with honeypots to detect credential dumping and Kerberos authentication attacks. (Fujimoto et al., 2018) proposed an algorithm to filter event IDs to detect Golden ticket or privilege escalation attacks. While (Siadati & Memon, 2017) uses a pattern mining algorithm to detect malicious logins or lateral movement attacks, and (Siadati et al., 2016) created a graphic user interface that illustrates pattern behavior to detect malicious logins using visual correlation and login events. It should be noted that future recommendations of these works may use a machine learning-based algorithm as recommended by authors.

Related works that do not use detection methods but manually configure the environment to enforce protection in the Windows environment are, (Nair & Sridaran, 2019), which implements best practice account and password creation to mitigate attacks such as Dictionary, Brute force, Rainbow Table, Phishing, Social Engineering, Malware, Offline Cracking, Shoulder Surfing, Spidering and Guessing attacks. (Sindiren & Ciylan, 2019) created an application model Privileged Account Access Control System to mitigate Dictionary, Brute force, Rainbow Table, Pass-the-Hash, Man in the Middle, and Sniffing attacks. While (Jillepalli et al., 2018) used Group Policy Object (GPO) and Active Directory Services to harden browser-related attacks such as JavaScript and Plugin-based and Phishing attacks. (Binduf et al., 2018) on the other hand, it applies the principle of least privilege to mitigate privilege escalation attacks, and (Wang & Gong, 2016) implements a centralized single sign-on (SSO) to protect application credentials. Aside from these manual configurations to harden the Windows environment, a different work by (Freitas et al., 2020) does not detect or mitigate Windows-related attacks but a framework to identify clients 2machines at-risk to lateral movement attacks.

The available related works on defending and securing the Windows environment from attacks are still a mix of static or manual configuration. At the same time, it is still low compared to full-blown machine learning-based algorithm detection methods. This literature review shows that many security researchers still consider machine learning-based detection to produce unreliable results because a robust or static configuration yields a 100% result in known or signature-based attacks. However, advantages show that machine learning-based detection methods can detect unknown or zero-day attacks.

Table 2 Studies Using Machine Learning Algorithm for Detection

Study	Detection Algorithm	False Positive Rate (%)	Recall (%)	Precision (%)	Accuracy (%)
(Mikhail et al., 2020) Sensitivity Plot to Detect New Service Creation	Random Forest	1.5	-	-	Able to detect (Binary Validation)
(Voris et al., 2019) Masqueraders Detection Accuracy	Gaussian Mixture Model	1	-	-	68
(Matsuda et al., 2018) Results for each Algorithm	One-Class SVM	-	100	100	100
	LOF	-	5	7	74
	Isolation Forest	-	43	90	50
(Hsieh et al., 2015) Performance Evaluation	Trendmicro and Markov Model	-	66.60	99.07	66.34

Lacking in the Approaches

The aforementioned related literature lacks the capabilities to detect unknown or zero-day attacks that this proposal intends to overcome. The studies of (Binduf et al., 2018; Freitas et al., 2020; Jillepalli et al., 2018; Nair & Sridaran, 2019; Sindiren & Ciylan, 2019; Wang & Gong, 2016) perform manual configuration to mitigate or block specific attacks would not be able to block unknown or zero-day attacks. On the other hand, related works in protecting the Windows environment that uses algorithms but does not use machine learning-based algorithms have identical capabilities in blocking Windows attacks. The only significant advantage of this is its automation compared to the studies listed above. These studies are as follows: (Kotlaba et al., 2020), which uses Splunk Processing Language, only detects attacks that are programmed to it for querying or filtering, (Fujimoto et al., 2018) proposed to utilized machine learning in future research to reduce false positive detection, (Siadati & Memon, 2017) use pattern mining with low accuracy results, and (Siadati et al., 2016) which uses a graphic user interface proposed exploring algorithm designs for future research.

On the other hand, related literature that is working on detection methods using machine learning, such as the work of (Mikhail et al., 2020), heavily relies on the logs produced by Sysmon, which is fed to a central logging aggregator or Security Incident and Event Manager (SIEM) for its input parameters. This method creates latency in producing results as the Sysmon installed in the host or client needs to be fed to the SIEM. On the other hand, the study of (Voris et al., 2019) uses file details in its input parameters. This input parameter or feature is not enough to determine whether a file is malicious or benign. The study of (Voris et al., 2019) yields an accuracy of 54% with a False Positive Rate (FPR) of 0.2%. (Matsuda et al., 2018) evaluated three machine learning algorithms, which is a good start. However, the input parameters are specific event ID logs, which are far too simple for input parameters for a machine learning-based algorithm. This method does not work for unrecognizable attacks with event IDs alone and to be used as an input feature to train the machine learning-based algorithm. The study in (Hsieh et al., 2015) utilized Active Directory logs as its input parameters, which were admitted in their conclusions to be insufficient to detect malicious behaviors or Windows attacks.

In addition to the lacking approaches, none of the related works deeply tackled how the attacks were performed to generate logs gathered with the corresponding attacks. Some enumerated the malicious binaries and executables they have used in their study, but none explained how it was used, executed, or weaponized. Some related works only gathered samples or pre-generated data, without even discussing how an attack from an attacker's point-of-view is executed. As a result, most of the studies used Windows event viewer logs as their input features and parameters.

Summary

Related studies are divided into protecting the Windows environment from manual configuration, non-machine learning-based algorithms, and machine learning-based algorithms. Currently, security researchers still opt for manual configuration and non-machine learning-based algorithms as these approaches give 100% reliability in detecting known and signature-based attacks. It is still experimental for machine learning-based algorithms. Most companies or organizations using machine learning-based detection methods are running it on top of the traditional or signature-based detection methods. The data obtained from these studies primarily differ as the authors evaluated different scenarios and used different input parameters. However, most of the studies used the Windows event viewer logs as their input parameters. No specific machine learning-based algorithm was proposed except for (Matsuda et al., 2018) Support Machine Vector. The challenge is still finding the optimum input feature parameters that can yield high accuracy and low false-positive rate.

III. WINDOWS COMMON ATTACK TECHNIQUES

Client-side Attacks

Client-side attacks are the stepping stones or initial access attacks to compromise the whole network or system fully. These attacks are commonly performed using the Web browser or HTTP/S (Hypertext Transfer Protocol or Hypertext Transfer Protocol Secure) and E-mail. Thus the Victim's machine must be connected to the Internet. The Web attack occurs when an unaware user browses malicious sites and tries to download a malicious file. This malicious file contains a malicious code that triggers a backdoor. This backdoor is a path or a connection from the Victim's machine to the Attacker's machine. The exact mechanism applies to the E-mail attack. An E-mail either contains a malicious URL link that deceives the recipient to click and browse the URL link, or the E-mail contains a malicious attachment that triggers the malicious code when downloaded and executed. Fig. 4 shows a high overview of how the adversary or cybercriminal gains initial access to the Victim's machine using this client-side attack. According to (*Malicious Attachments Remain a Cybercriminal Threat Vector*

Favorite, 2020), malicious attachments are the oldest trick in spreading malware. However, it is still one of the top threat vectors in the cybercriminal world.

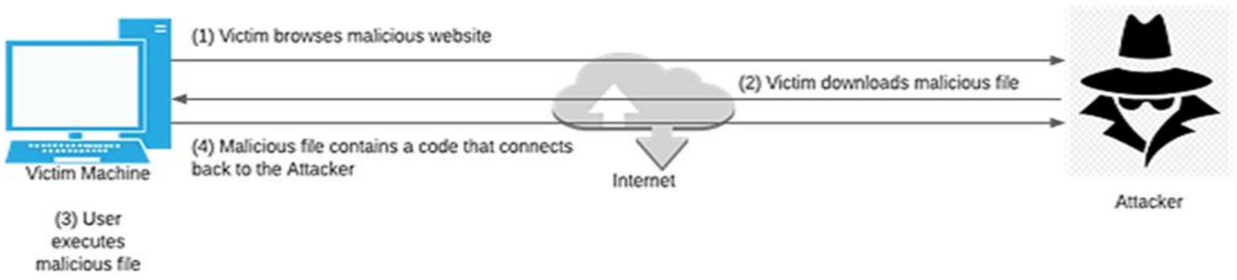
(1) Victim browses malicious website
(2) Victim downloads malicious file
(4) Malicious file contains a code that connects back to the Attacker
Victim Machine
(3) User executes malicious file
Internet
Attacker

Figure 4. Client-side Attack High Overview

The list of available Initial Access or Client-side attacks that weaponizes malicious binaries and executables are as follows:

- Hypertext Markup Language (HTML) Application – Victim Windows machine needs to browse the malicious Unifrom Resource Locator (URL).
- Microsoft Office Macro – Victim Windows machine needs to open up the malicious Word document.
- Object Linking and Embedding – Victim Windows machine needs to open up the malicious Word document and execute the embedded object.
- Malicious Code – Victim Windows machine needs to execute the malicious executable.
- Remote Process Memory Injection - Victim Windows machine needs to execute the malicious Powershell script.
- Shellcode Injection – Victim Windows machine needs to execute the malicious executable process by shellcode injection tool or Shellter (*Shellter | Shellter*, 2020).

Privilege Escalation Attacks

After the adversary or cybercriminal gains initial access, the next step is to perform a Privilege Escalation attack if the user privilege initially gained has insufficient privilege. Privilege Escalation enables the adversaries or cybercriminals to gain higher privileges that would allow them to explore the network further or gain access to sensitive files and data, including credentials. Samples of high privilege accounts are local administrators (Local admins), SYSTEM users, and Domain admins. According to (Fujimoto et al., 2018; Matsuda et al., 2018), domain admin privilege can be used to obtain long-term administrator privilege or access any machines that are part of the domain. This access falls under Lateral Movement attacks.

On the other hand, Local Administrators can be elevated further to System Users who are allowed to dump user credentials or password hashes coming from the Security Account Manager (SAM) database and capture Kerberos tickets at the same time. SAM database contains password hashes, either in LAN Manager (LM) or New Technology LAN Manager (NTLM) hash format. The most well-known binary that is being used in this credential dumping attack is Mimikatz.

The list of available Privilege Escalation attacks that weaponize malicious binaries and executables are as follows:

- Enumerate Readable and Writable Files and Directories – The AccessChk binary from (markruss, 2020a) is used to identify world-writable files or directories overwritten by malicious binaries or reverse shell to elevate privileges.
- Enumerate Device Drivers and Kernel Modules - The Driverquery binary from (eross-msft, 2020a) is used to list driver versions and verify if an exploit is available for the listed driver versions.
- Automated Enumeration – The Windows-privesc-check binary from (pentestmonkey, 2015/2020) is used to automatically enumerate the information and vulnerabilities of the system that can be used for privilege escalation.
- Additional Binaries (Sigcheck, Icalcs, Juicy Potato, and Mimikatz) - The Sigcheck binary from (markruss, 2020c) is used to verify

the integrity level and permission to run a process. The Icacls binary from (eross-msft, 2020b) is used to enumerate associated permissions. The Juicy Potato binary from (*Juicy Potato (Abusing the Golden Privileges)*, 2020) is used to exploit SeImpersonate privilege from Windows service accounts to system privilege. The Mimikatz binary from (dimi, 2020) elevates security tokens from administrator to system privilege. These four binaries are commonly abused for privilege escalation attacks.

Lateral Movement Attacks

Lateral Movement attacks are performed to gain high-valued targets or to explore the network entirely. By leveraging the Privilege Escalation attack or obtaining domain admin credentials, Kerberos tickets, or password hashes, this can be used to access other machines or servers within the network. The attacks that are commonly performed under Lateral Movement attack are:

• Pass the Hash – by using the stolen hash, the attacker can authenticate to a remote machine. This only works with NTLM authentication. The pth-winexe binary from (byt3bl33d3r, 2015/2020, p. 3) is used to authenticate using a password hash dump remotely.

• Overpass the Hash – Using the stolen NTLM hash, the attacker can gain a Kerberos Ticket Granting Ticket, allowing the attacker to authenticate to a remote machine. The PsExec binary from (markruss, 2020b) is used to obtain remote code execution using generated Kerberos tickets and impersonate a domain user.

• Pass the Ticket – This attack leverages Kerberos Ticket Granting Service, which offers more flexibility than Kerberos Ticket Granting Ticket since it can be used to a specific service and not only in a specific machine. This attack leverages the Mimikatz binary mentioned above to craft a silver ticket for remote code execution.

• Golden Ticket – This attack aims to create a custom-made Kerberos Ticket Granting Ticket by obtaining the KRBTGT password hash. This attack covers PsExec and Mimikatz binaries' combination usage.

IV. MACHINE LEARNING-BASED DETECTORS

Logistic Regression

Logistic Regression machine learning-based detector is implemented to identify malicious and benign samples. The Logistic Regression model creates a boundary to identify whether the sample is malicious or benign. The negative log-likelihood depicts the loss function of Logistic Regression is

$$\ell(\{p_i\},\{y_i\}) = \sum_i((1-y_i)\log(1-p_i) + y_i \log p_i \qquad (1)$$

where $\{p_i\}$ stands for probability predictions and $\{y_i\}$ stands for truth labels. The likelihood of all predictions is shown below as the product of each likelihood:

$$\mathcal{L}(\{p_i\},\{y_i\}) = \prod_{y_i=0} 1-p_i \cdot \prod_{y_i=1} p_i \qquad (2)$$

Logistic Regression aims to search for the best parameters that produce the probabilities that optimize or maximize the likelihood. Logistic Regression identifies the binaries or executables using a hyperplane. The hyperplane depends on the number of fed or configured features to the logistic regression algorithm, which geometrically separates malicious from benign samples. When a sample or an unseen binary or executable is fed into the detector, Logistic Regression classifies the sample on the malicious or benign side of the boundary.

Sklearn.Linear_model.LogisticRegression is used in constructing Logistic Regression machine learning detector (*Sklearn.Linear_model.LogisticRegression — Scikit-Learn 0.23.2 Documentation*, 2020).

Random Forest

Random Forest machine learning-based detector is implemented to identify malicious and benign samples. The Random Forest model heavily relies on decision trees, and each decision tree votes to identify whether the sample is malicious or benign. The Random Forest algorithm workflow is as follows:

1. A random subset of N samples (trained individual trees) from the training dataset is chosen.
2. Random X features are chosen from the available Y features on each split point, and the optimal split point is chosen among these X features, where X ≤ Y.
3. Do step 2 until each tree is trained.
4. Do steps 1, 2, and 3 until all trees in the forest are trained.

The probability that a binary or executable is identified, whether malicious or benign, depends on the number of decision tree votes divided by the total number of decision trees. Sklearn.Ensemble.RandomForestCLassifier is used in constructing the Random Forest machine learning detector (*3.2.4.3.1. Sklearn.Ensemble.RandomForestClassifier — Scikit-Learn 0.23.2 Documentation*, 2020).

Support Vector Machines

Support Vector Machine (SVM) learning-based detector is implemented to identify malicious and benign samples. SVM creates a hyperplane like logistic Regression, and the difference between the two is the loss function. SVM implements hinge loss which penalizes samples that are on the wrong side only. In contrast, logistic Regression implements a log-likelihood function that penalizes all samples proportionally to the probability error estimate. The loss function of the support vector machine is shown

$$\boldsymbol{\beta} + \boldsymbol{C}\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{N}} \boldsymbol{\xi}_{\boldsymbol{i}} \qquad (3)$$

where the margin is $\boldsymbol{\beta}$, the hyperparameter that is relative to the contribution of the two terms is $\boldsymbol{C}$, and the distance of the margin to the $\boldsymbol{i}$th support vector is $\boldsymbol{\xi_i}$. Sklearn.Svm.SVC is used in constructing the Support Vector Machine detector (*Sklearn.Svm.SVC — Scikit-Learn 0.24.2 Documentation*, 2021).

Neural Network

Another algorithm is the Neural Networks (NN) or Artificial Neural Network (ANN), as shown in Fig. 5. It is a profound interconnection of a primary computational factor known as a perceptron, which are fundamental models of neurons in the human brain. Its architecture and calculation are utterly parallel networks of distinct computational elements systematized in correlation to each other. The learning process in this kind of algorithm is apparent in a manner. It can also produce accurate and reliable expected results or outputs.

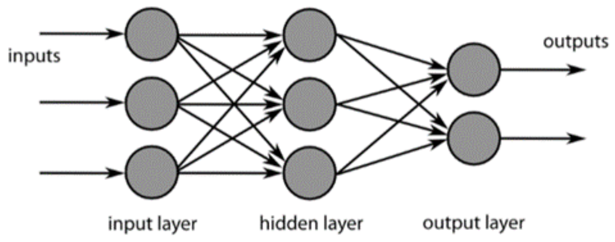

Figure 5. The Architecture of the Neural Network

The input layer on the left side consists of a set of new neurons $\boldsymbol{x_i}$ which represent the input

$$\{\boldsymbol{x_i}|\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots, \boldsymbol{x_n}\}. \qquad (4)$$

The middle, which is the hidden layer, transforms previous layers' values using linear weights w_i summation

$$\{\boldsymbol{w_i x_i} + \boldsymbol{w_1 x_1} + \boldsymbol{w_2 x_2} + \boldsymbol{w_3 x_3} + \dots + \boldsymbol{w_n x_n}\} \qquad (5)$$

and nonlinear activation function such as rectified linear unit $\boldsymbol{R}(\boldsymbol{z})$ (ReLU) with $\boldsymbol{z}$ as input is applied.

$$\boldsymbol{R}(\boldsymbol{z}) = \max(\boldsymbol{0}, \boldsymbol{z}) \qquad (6)$$

To optimize the parameters with backpropagation, this activation function or ReLU applies a nonlinear transformation to the weighted sum, resulting in the neuron's input data linear transformation. After that, the last layer or the right side, the output layer, retrieves the values from the last hidden layer, transforming and outputs them. Sklearn.Neural_network.MLPClassifier is used in constructing Neural Network detector (*1.17. Neural Network Models (Supervised) — Scikit-Learn 0.24.2 Documentation*, 2021).

V. RESULTS AND DISCUSSION

The previous section provided an overview of the three common Windows attack techniques and introduced the binaries and executables used in the attacks. Section Machine Learning-based Detectors discussed the machine learning algorithms that were used in constructing the detector. Scikit-Learn library for hashing input features such as FeatureHasher and creating a machine learning-based detector using Logistic Regression, Random Forest, Support Vector Machine, and Neural Network covered in Sec. Machine Learning-based Detectors. These tools and libraries have been used to complete this study. In addition to the machine-based detector results, samples from the attacks were also evaluated using VirusTotal (*VirusTotal*, 2020) for third-party anti-malware engine comparison.

Training and Testing

Table 3 shows the sample dataset that has been gathered from VirusTotal and Windows Server 2016 system32 folder. Additional samples for testing and used in sample attacks have been created in Kali Linux Operating System using MSFVenom (*MSFvenom | Offensive Security*, 2020). The sample dataset indicates the category, type, platform, alias, quantity, or the number of samples. This dataset is the sample used for training and testing the machine learning-based detector.

Table 3 Sample Dataset

Category	Type	Platform	Alias	Quantity	Alias	Quantity	Alias	Quantity
Malware	Backdoor	Win32	No-Alias	35	Delf	8	IRCBot	8
			Asper	1	Donbot	1	Koutodoor	1
			Banito	1	DsBot	19	LolBot	3
			Beastdoor	1	Dusta	1	MeSub	1
			Bifrose	106	FirstInj	5	Netbus	1
			BlackHole	54	Floder	1	Nucleroot	1
			Bredolab	26	FlyAgent	1	Papras	10
			Ciadoor	2	Gbot	32	PcClient	2
			Cinkel	1	Gnutler	1	Poison	31
			Clemag	7	Httpbot	1	Portless	1
			Curioso	1	Hupigon	50	Prorat	10
			DDOS	1	Inject	2		
For Testing	Customize	Win32	-	14				
Benign	DLL	Win32	-	405				
Benign	Executable	Win32	-	96				

The first step in this training and testing process is to extract the string features of the samples. However, the output of string feature extraction produces too many features that a machine learning algorithm could handle and causes memory issues. For example, if sample one contains the string "malicious sample" and the second sample contains a string "malicious sample!" this is treated as two separate features. This example means that it quickly encounters too many unique strings that end up being used for training the detector. Secondly, suppose that the output has around one thousand features, and the samples available are also around one thousand. In this case, the samples would not be enough to train the machine learning-based detector of what each of the features describes a given binary or simply known as the curse of dimensionality.

A hashing trick has been implemented—feature hashing from scikit-learn (*6.2. Feature Extraction — Scikit-Learn 0.23.2 Documentation*, 2020), also known as FeatureHasher, has been applied to overcome the problem of having too many features. By applying FeatureHasher, the extracted features are encoded in a vectorized form instead of building a hash table. A limit is also assigned for the length of the feature matrix.

After extracting and hashing the input features from binaries and executables, it can now train the machine learning-based detector. After the training has been performed, the detector can detect new binaries and executables to classify whether it is benign or malicious. Several detector algorithms were implemented and evaluated. The machine learning-based detectors evaluated are Logistic Regression, Random Forest, Support Vector Machine, and Neural Network. Due to the related studies of (Mikhail et al., 2020) and (Voris et al., 2019), not mentioning what type of machine learning they have implemented, Support Vector Machines has been chosen because of (Matsuda et al., 2018) study in Support Vector Machines which yielded a high accuracy result and uses regression analysis and binary linear classifier. In addition to this, Logistic Regression machine learning has also been evaluated because it is also under a linear classifier and one of the basic ones.

On the other hand, Random Forest machine learning has higher complexity. It is under the same supervised learning model as the Logistic Regression machine learning has been covered and evaluated because of its nature from the Binary or Decision Tree machine learning algorithm used to solve detection problems. Neural Network, a prerequisite for more complex algorithms such as Deep and Reinforcement Learning, is evaluated. Unsupervised learning is not covered, as there is no way to train the detector for false positives and false negatives for immediate results, making it less effective in detecting malicious or benign binaries and executables. Python programming language has been used in creating these detectors as scikit library is using Python language.

Machine Learning Results without Sample Training

Figure 6 was tested without adding the samples for testing in training samples. The result shown in Fig. 6 shows that around half of the samples for testing are indistinguishable without adding the sample for testing in training the machine learning-based detectors. Relevant accuracy value in results or removing outliers shows that Support Vector Machine has detection accuracy at 93.23%, followed by Random Forest at 75.25%. While without removing the outliers and including all the testing results, Support Vector Machine has detection accuracy at 53.35%, followed by Random Forest at 50.36%. This result shows that the Support Vector Machine followed by Random Forest has the top results in determining whether a binary or executable is malicious or benign.

Machine Learning Results with Sample Training

Figure 7 was tested by adding the samples for testing in training samples. The result shown in Fig. 7 shows that the detection or classification of malicious samples significantly improved when the testing samples are added in training the machine learning-based detectors. However, the Support Vector Machine's performance stays the same while the three detectors significantly improve their accuracy. The sample results are the same as above upon the removal of outliers. It shows that Artificial Neural Network has detection accuracy at 99.98%, followed by Logistic Regression at 99.71%. While without removing the outliers and including all the testing results, Logistic Regression has detection accuracy at 94.42%, followed by Artificial Neural Network at 93.15%. On this result, Logistic Regression and Artificial Neural Network take the lead in the accuracy of classifying or detecting malicious samples.

Machine Learning Processing Time

Figure 8 shows each machine learning-based detector's processing or execution time on classifying or identifying the testing samples, whether it is malicious or benign. The result shown in Fig. 8 shows that the processing time of the Support Vector Machine is high at 1809 milliseconds without adding testing samples in training samples and 1841 milliseconds with adding testing samples in training samples, followed by Artificial Neural Network at 979 milliseconds without adding testing samples in training samples and 966 milliseconds with adding testing samples in training samples. The least or best processing time for the machine learning algorithms is the Logistic Regression at 24 milliseconds without adding testing samples in training samples and 24 milliseconds as well with adding testing samples in training samples, followed by Random forest at 49 milliseconds without adding testing samples in training samples and 51 milliseconds with adding testing samples in training samples. Logistic Regression outperforms Random Forest in processing or execution time by 25 to 27 milliseconds with a difference in results.

Summarized Comparative Results

Figure 9 shows the empirical cumulative distribution results of Fig. 6 and Fig. 7. The result shown in Fig. 9 shows that both Logistic Regression and Artificial Neural Network have better results with an average accuracy of 94.42% and 93.15%, respectively, when a testing sample is added in training. Otherwise, when a testing sample is not added in training, Support Vector Machine followed by Random Forest shows better results with an average accuracy of 53.35% and 50.36%, respectively.

A third-party scanning engine or VirusTotal result was added in Fig. 9, and the data of VirusTotal is an aggregation of different anti-malware vendors or scanning engines, where the accuracy V_T results are equal to the total anti-malware engines that can classify the malicious sample correctly D_T divided by the total number of anti-malware engines E_T that was processed when the sample for testing was submitted in VirusTotal, as shown in (7).

$$V_T = {}^{D_T}/_{E_T} \qquad (7)$$

VirusTotal results in more than 50% of anti-malware engines cannot classify the malicious samples correctly, with a 39.02% overall accuracy results of all combined third-party scanning engines in identifying all testing samples used in machine learning-based detectors.

VI. CONCLUSION

Fig. 9 shows that the created machine learning-based detector performs better in identifying or classifying the malicious samples than current anti-malware products. The learning-based detector performs better in identifying or classifying the malicious. Among the list of machine learning-based detectors evaluated, with testing samples included in the training, Logistic Regression performed better in accuracy and processing time than other machine learning-based detectors. However, if testing samples are not included in the training, the Support Vector Machine came on top.

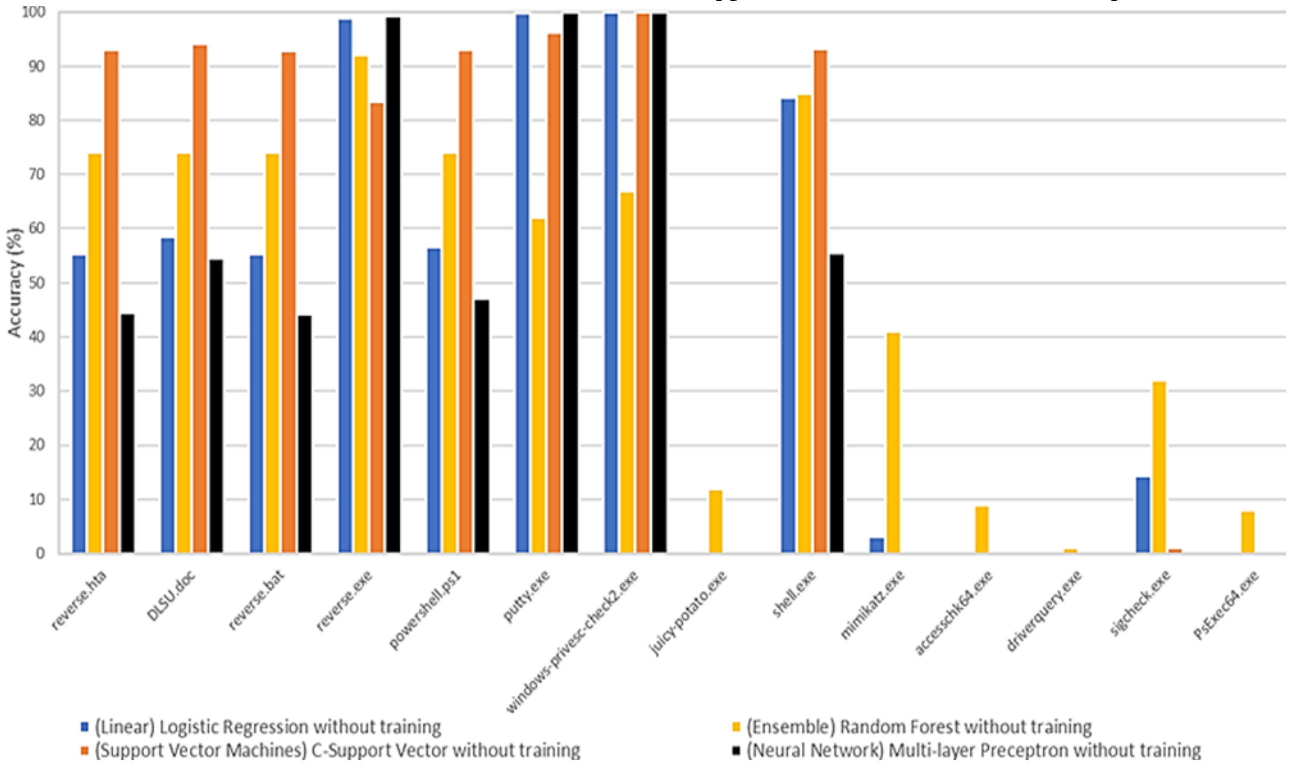

Figure 6. Machine Learning Accuracy Results (Without Sample Training)

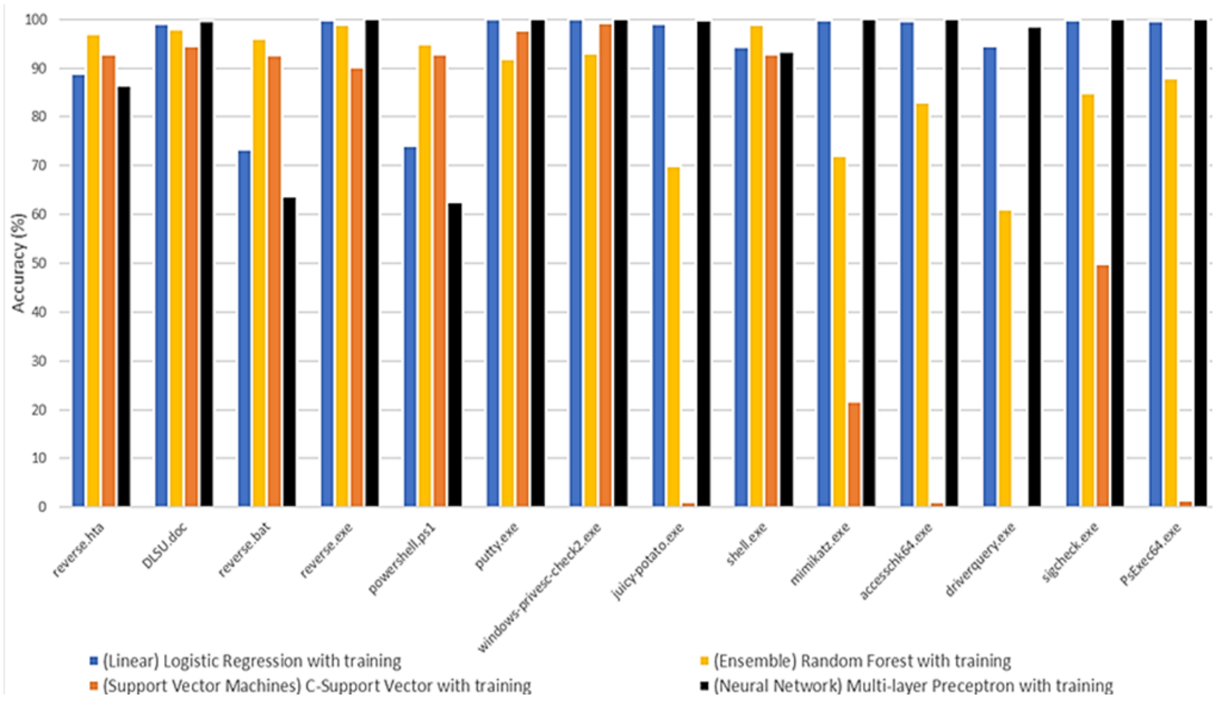

Figure 7. Machine Learning Accuracy Results (With Sample Training)

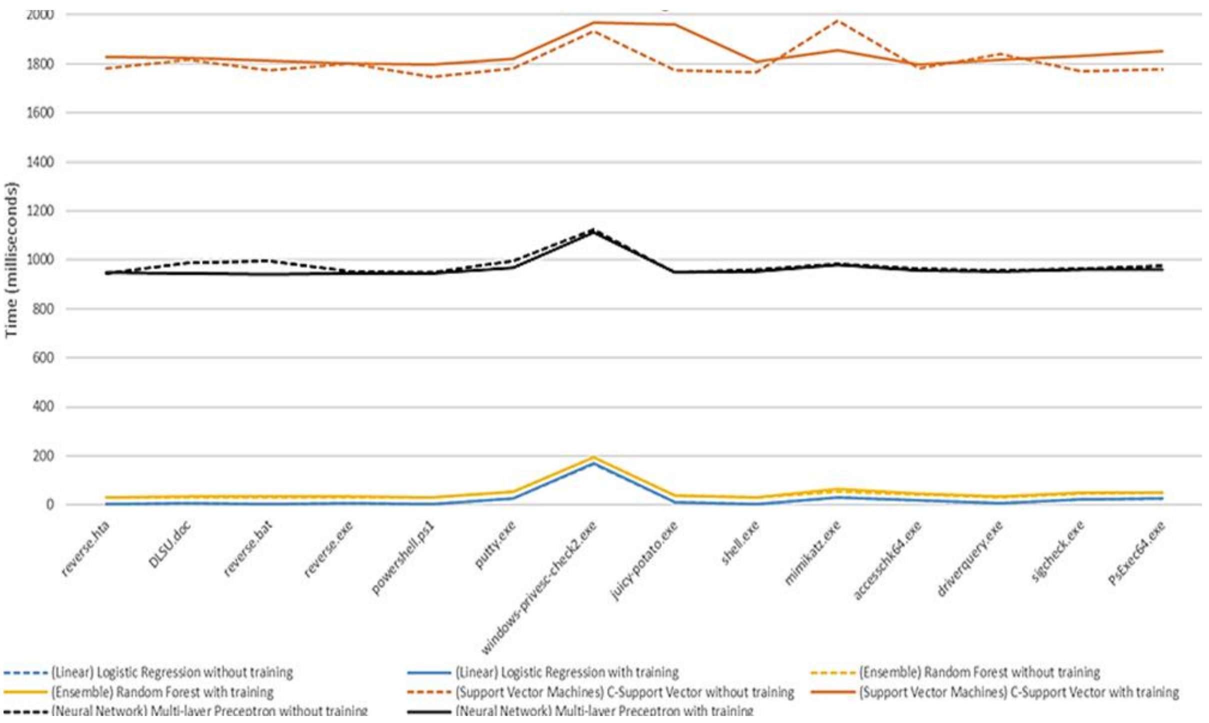

Figure 8. Machine Learning Processing Time Results

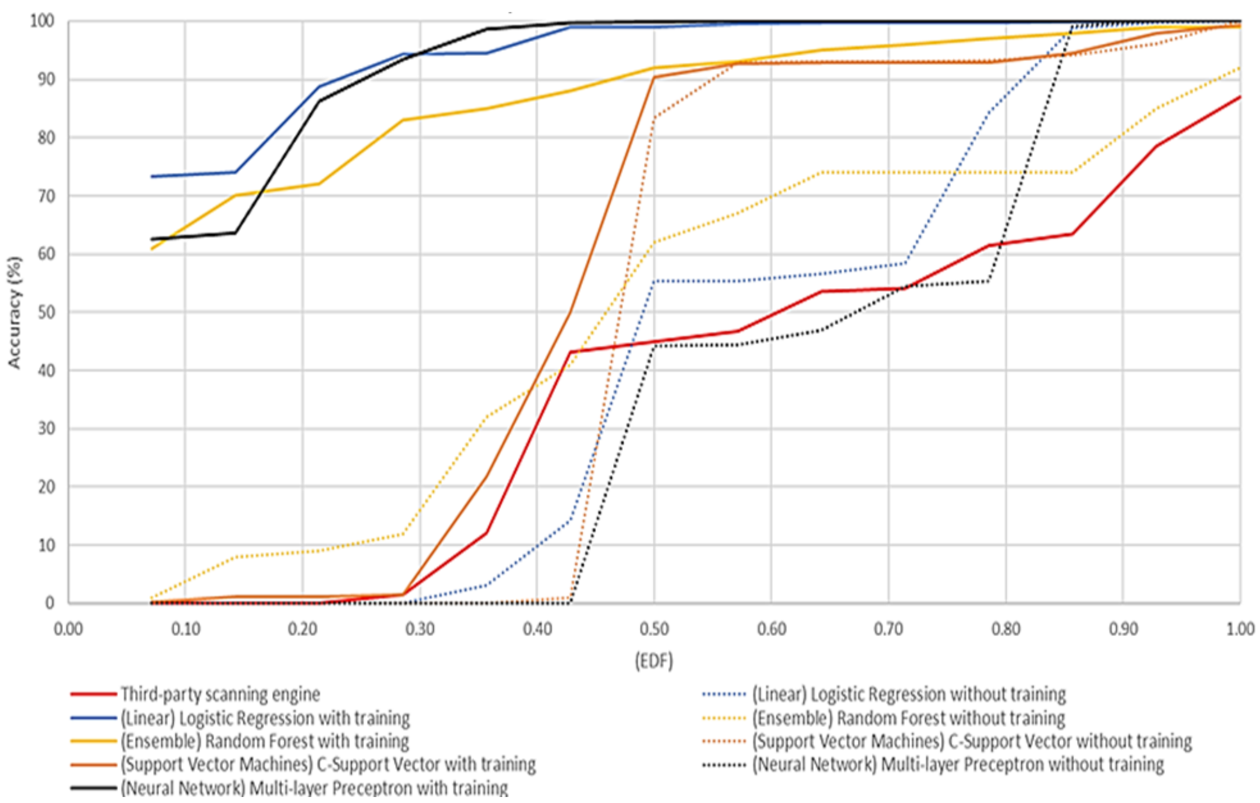

Figure 9. Empirical Cumulative Distribution of Results

Implementing this machine learning-based detector makes it possible to detect the malware used in zero-day attacks or attacks explicitly improvised for such an organization or company without relying on a third-party vendor or product. In addition, Organizations or companies do not need to wait for vendors or third-party malware detectors to release signatures or indicators to remediate this malware used in the attacks. Having the capability to block such attacks results in fewer organizations and companies being compromised and exploited by Cybercriminals.

For future studies, it is recommended to evaluate other machine learning algorithms such as Nearest Neighbors and Semi-supervised learning. In addition to string as an input feature, it would also be for future research to add or combine other input features such as portable executable headers, assembly instructions, import address translation, and N-grams with string. The aim would be to classify or detect benign or malicious samples more accurately with the shortest processing or execution time, equating to fewer resources consumed, such as memory and process.

ACKNOWLEDGMENT

De La Salle University is acknowledged for supporting this work.

BIBLIOGRAPHY

1. 1.17. Neural network models (supervised)—Scikit-learn 0.24.2 documentation. (2021). https://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification
2. 3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier—Scikit-learn 0.23.2 documentation. (2020). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
3. 6.2. Feature extraction—Scikit-learn 0.23.2 documentation. (2020). https://scikit-learn.org/stable/modules/feature_extraction.html
4. Another massive ransomware outbreak – or was it? (2017). Computer Fraud & Security, 2017(7), 1–3.

https://doi.org/10.1016/S1361-3723(17)30055-6

5. Binduf, A., Alamoudi, H., Balahmar, H., Alshamrani, S., Al-Omar, H., & Nagy, N. (2018). Active Directory and Related Aspects of Security. 4474–4479. https://doi.org/10.1109/NCG.2018.8593188
6. byt3bl33d3r. (2020). Byt3bl33d3r/pth-toolkit [Python]. https://github.com/byt3bl33d3r/pth-toolkit (Original work published 2015)
7. Cimpanu, C. (2020). Financial sector is seeing more credential stuffing than DDoS attacks. ZDNet. https://www.zdnet.com/article/financial-sector-has-been-seeing-more-credential-stuffing-than-ddos-attacks-in-recent-years/
8. Compromised Credential Attacks are Frequent and Costly • Designed Privacy. (2020). https://designedprivacy.com/compromised-credential-attacks-are-frequent-and-costly/
9. Deland-Han. (2020). Security identifiers in Windows—Windows Server. https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/security-identifiers-in-windows
10. dimi. (2020). (0x64 ∧ 0x6d) ∨ 0x69 ~ mimikatz: Deep dive on lsadump::lsa /patch and /inject. https://blog.3or.de/mimikatz-deep-dive-on-lsadumplsa-patch-and-inject.html
11. eross-msft. (2020a). Driverquery. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/driverquery
12. eross-msft. (2020b). Icacls. https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls
13. Freitas, S., Wicker, A., Chau, D. H., & Neil, J. (2020). D2M: Dynamic Defense and Modeling of Adversarial Movement in Networks.
14. Fujimoto, M., Matsuda, W., & Mitsunaga, T. (2018). Detecting Abuse of Domain Administrator Privilege Using Windows Event Log. 2018 IEEE Conference on Application, Information and Network Security (AINS), 15–20. https://doi.org/10.1109/AINS.2018.8631459
15. Heritage, I. (2019). Protecting Industry 4.0: Challenges and solutions as IT, OT and IP converge. Network Security, 2019(10), 6–9. https://doi.org/10.1016/S1353-4858(19)30120-5
16. Hsieh, C., Lai, C., Mao, C., Kao, T., & Lee, K. (2015). AD2: Anomaly detection on active directory log data for insider threat monitoring. 2015 International Carnahan Conference on Security Technology (ICCST), 287–292. https://doi.org/10.1109/CCST.2015.7389698
17. Jillepalli, A., Conte de Leon, D., Sheldon, F. T., & Haney, M. (2018). Enterprise-level Hardening of Web Browsers for Microsoft Windows. 7, 261–274. https://doi.org/10.12785/ijcds/070501
18. Juicy Potato (abusing the golden privileges). (2020). Juicy-Potato. http://ohpe.it/juicy-potato/
19. Kotlaba, L., Buchovecká, S., & Lórencz, R. (2020). Active Directory Kerberoasting Attack: Monitoring and Detection Techniques. ICISSP.
20. Lemay, A., Calvet, J., Menet, F., & Fernandez, J. M. (2018). Survey of publicly available reports on advanced persistent threat actors. Computers & Security, 72, 26–59. https://doi.org/10.1016/j.cose.2017.08.005

21. Malicious Attachments Remain a Cybercriminal Threat Vector Favorite. (2020). https://threatpost.com/malicious-attachments-remain-a-cybercriminal-threat-vector-favorite/158631/
22. Mansfield-Devine, S. (2017). Editorial. Computer Fraud & Security, 2017(7), 2. https://doi.org/10.1016/S1361-3723(17)30056-8
23. markruss. (2020a). AccessChk—Windows Sysinternals. https://docs.microsoft.com/en-us/sysinternals/downloads/accesschk
24. markruss. (2020b). PsExec—Windows Sysinternals. https://docs.microsoft.com/en-us/sysinternals/downloads/psexec
25. markruss. (2020c). Sigcheck—Windows Sysinternals. https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck
26. Matrix—Enterprise | MITRE ATT&CK®. (2020). https://attack.mitre.org/matrices/enterprise/windows/#
27. Matsuda, W., Fujimoto, M., & Mitsunaga, T. (2018). Detecting APT Attacks Against Active Directory Using Machine Leaning. 2018 IEEE Conference on Application, Information and Network Security (AINS), 60–65. https://doi.org/10.1109/AINS.2018.8631486
28. Microsoft: Products and vulnerabilities. (2020). https://www.cvedetails.com/vendor/26/Microsoft.html
29. Microsoft report shows increasing sophistication of cyber threats. (2020, September 29). Microsoft on the Issues. https://blogs.microsoft.com/on-the-issues/2020/09/29/microsoft-digital-defense-report-cyber-threats/
30. Mikhail, J. W., Williams, J. C., & Roelke, G. R. (2020). procmonML: Generating evasion resilient host-based behavioral analytics from tree ensembles. Computers & Security, 98, 102002. https://doi.org/10.1016/j.cose.2020.102002
31. MSFvenom | Offensive Security. (2020). https://www.offensive-security.com/metasploit-unleashed/msfvenom/
32. Nair, H., & Sridaran, R. (2019). An Innovative Model (HS) to Enhance the Security in Windows Operating System—A Case Study. 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), 1207–1211.
33. Orlowski, C. (2021). Chapter 2—Architectures and reference models in designing Internet of Things systems. In C. Orlowski (Ed.), Management of IOT Open Data Projects in Smart Cities (pp. 43–86). Academic Press. https://doi.org/10.1016/B978-0-12-818779-1.00002-X
34. pentestmonkey. (2020). Pentestmonkey/windows-privesc-check [Python]. https://github.com/pentestmonkey/windows-privesc-check (Original work published 2015)
35. Ransomware Upgrades with Credential-Stealing Tricks. (2020). Dark Reading. https://www.darkreading.com/attacks-breaches/ransomware-upgrades-with-credential-stealing-tricks/d/d-id/1336846
36. Shellter | Shellter. (2020). https://www.ShellterProject.com/introducing-shellter/
37. Siadati, H., & Memon, N. (2017). Detecting Structurally Anomalous Logins Within Enterprise Networks. 1273–1284. https://doi.org/10.1145/3133956.3134003

38. Siadati, H., Saket, B., & Memon, N. (2016). Detecting malicious logins in enterprise networks using visualization. 2016 IEEE Symposium on Visualization for Cyber Security (VizSec), 1–8. https://doi.org/10.1109/VIZSEC.2016.7739582
39. Sindiren, E., & Ciylan, B. (2019). Application model for privileged account access control system in enterprise networks. Computers & Security, 83, 52–67. https://doi.org/10.1016/j.cose.2019.01.008
40. sklearn.linear_model.LogisticRegression—Scikit-learn 0.23.2 documentation. (2020). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
41. Sklearn.svm.SVC — scikit-learn 0.24.2 documentation. (2021). https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
42. The 2020 Data Breach Investigations Report – a CSO's perspective—ScienceDirect. (2020). https://0-www-sciencedirect-com.lib1000.dlsu.edu.ph/science/article/pii/S1353485820300799
43. VirusTotal. (2020). https://www.virustotal.com/gui/
44. Voris, J., Song, Y., Salem, M. B., Hershkop, S., & Stolfo, S. (2019). Active authentication using file system decoys and user behavior modeling: Results of a large scale study. Computers & Security, 87, 101412. https://doi.org/10.1016/j.cose.2018.07.021
45. Wang, H., & Gong, C. (2016). Design and Implementation of Unified Identity Authentication Service Based on AD. 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), 394–398. https://doi.org/10.1109/CICN.2016.84
46. Yuksel, H., & Altunay, Ö. (2020). Host-to-host TCP/IP connection over serial ports using visible light communication. Physical Communication, 43, 101222. https://doi.org/10.1016/j.phycom.2020.101222