ISSN: 0974-5823 Vol. 6 No. 3 December, 2021

International Journal of Mechanical Engineering

Evaluation of Detecting Malicious Binaries and Executables Using Machine Learning-Based Detectors

John Martin M. Ladrido
De La Salle University, Philippines.

Lawrence Materum
De La Salle University, Philippines.

Abstract **- Cyberattacks nowadays are prevalent, and cybercriminals do not choose their target, whether small or large. The most common form of an attack utilizes malware where the victim users execute this malware in their workstations. By executing this malware, cybercriminals gains access to these machines and gain an initial foothold. The result of these attacks causes organization financial losses. Thus, preventing and detecting this malware or at the same time protecting the users from this malware is imperative. This paper evaluates several machine learning algorithms in detecting malware and comparing its accuracy results with a 3rd party Antivirus product.**

Index Terms – Malware, Viruses, Machine Learning, Classification, Detection, Cybersecurity.

INTRODUCTION

A Data communication has progressed from a primary sender-receiver data transfer to a web of information transmission from one network system to the next. It is critical to protect data and information from hackers and enemies as knowledge gleaned from data and information becomes potentially more valuable than oil or gold. Aside from data and information, cybercriminals frequently target industrial control systems, supervisory control and data acquisition, and other IoT-enabled systems. According to Microsoft's defense report [1], sophisticated attacks increasingly focus on credential harvesting and ransomware. In the first half of 2020, the total number of attacks on IoT climbed by 35%. Furthermore, according to a ZDNet analysis based on F5 statistics [2], Brute Force or Credential Stuffing is the most common security issue, accounting for 41%, followed by Distributed Denial-of-Service (DDoS) at 32%.

On the other hand, ransomware is now being used to steal credentials in popular browsers like Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer [3]. Ransomware has been improved to steal E-mail passwords in Mozilla Thunderbird and Microsoft Outlook, in addition to its web browser attack vector. Cybercriminals and enemies with high-privilege credentials to manage workstations and servers can disrupt manufacturing companies, power generation organizations, transportation controllers, and, worst of all, nuclear and military weapons. According to [4], a malware outbreak in Ukraine recently led to the shutting down of monitoring equipment for radiation levels at the Chernobyl Nuclear Power Plant.

According to [5], because of the convergence of information and operational technologies, cybercriminals and adversaries now have more attack avenues as a result of the 4th industrial revolution. Computer systems have an average life cycle of nine years. Machines and sensors from the Internet of Things or robotics, on the other hand, may endure 20 to 30 years, providing cybercriminals and adversaries more time to uncover weaknesses and attacks. According to [6], the number of cybercriminals and adversaries is on the rise around the world, making it more difficult for small businesses and organizations with limited budgets and manpower to defend themselves against cybersecurity threats. As a result, according to [7], cybersecurity is always depicted as an arms race between cybercriminals and security firms, with the digital environment serving as the battleground. According to [8], the top two reasons for security breaches are a lack of time to find vulnerabilities and attack vectors (31%) and a lack of competence or appropriate knowledge to remediate the vulnerabilities discovered (21%). As a result, most businesses and organizations are unaware of how cybercriminals and adversaries create cyber-attacks and how to protect themselves from being hacked.

Cybercriminals or adversaries usually emerge from the

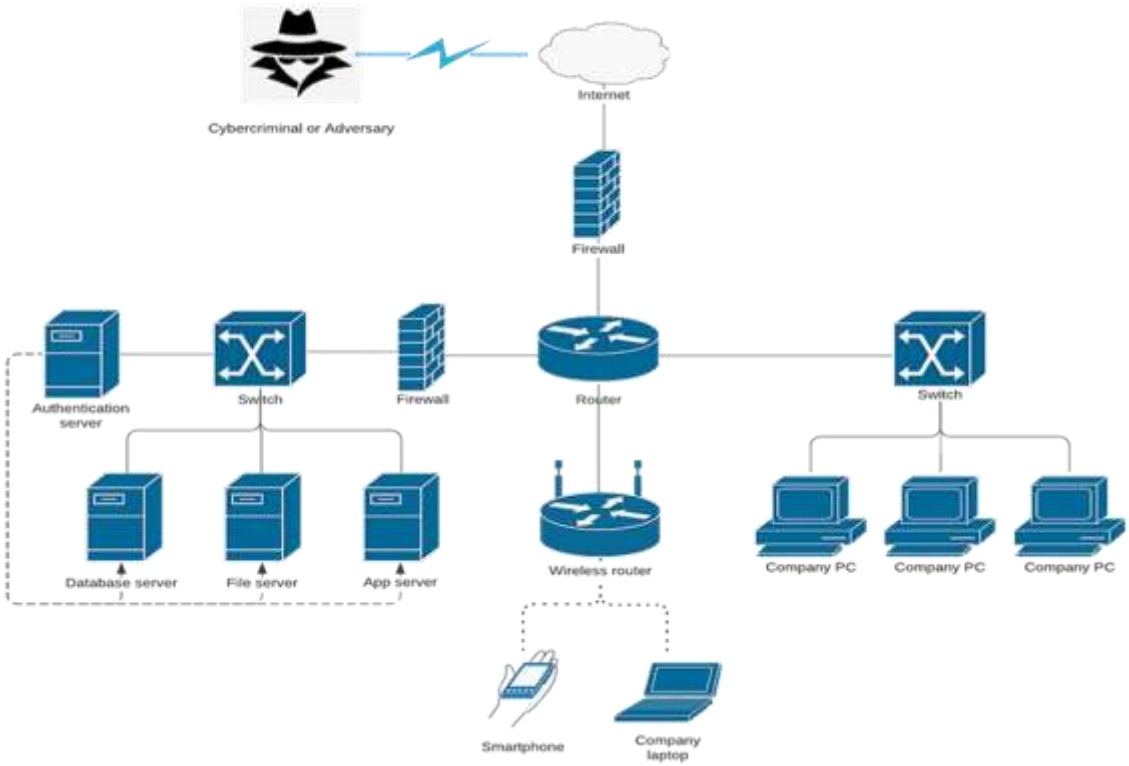

FIGURE 1
Simple Enterprise Network Architecture

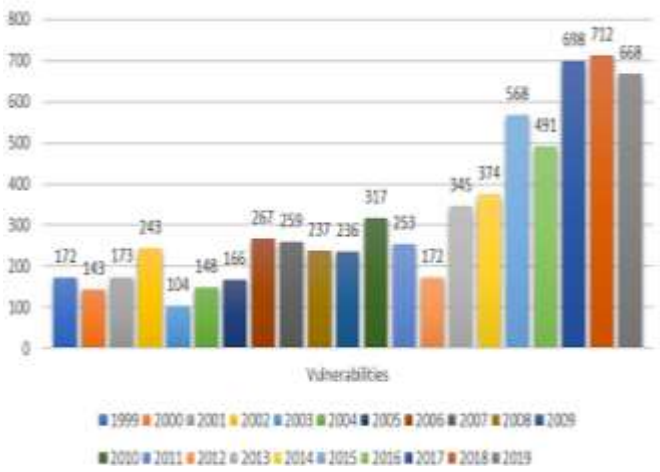

FIGURE 2
Microsoft Vulnerabilities By Year – Redrawn from [9]

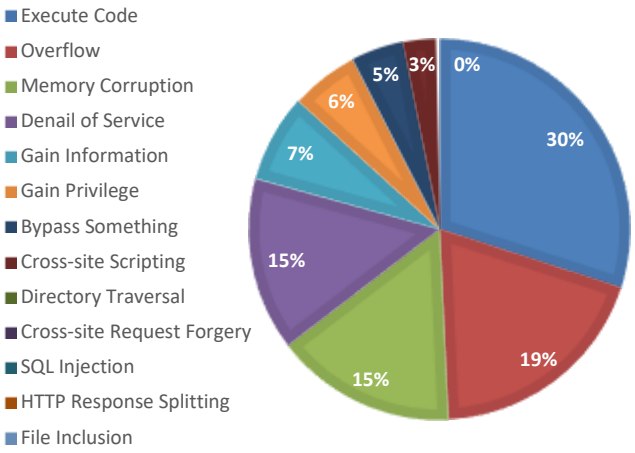

FIGURE 3
Microsoft Vulnerability Types Breakdown from 1999 to 2019 – Redrawn from [9]

Internet in a company's or organization's network system environment, as depicted in Fig. 1, and attack vectors start to form as corporations and organizations open up or try to access the Internet. Assume that a corporation or organization does not have an Internet connection. In that circumstance, the attack routes that cybercriminals or adversaries may use to minimize local attacks, usually insider attacks, and the attack vector is considerably reduced. As a result, fraudsters and enemies must physically connect to the network. On the other hand, web, e-mail, and file transfer are the most common Internet attacks. Before moving on to servers, these attacks normally target a workstation or client. Employees or users using these workstations are vulnerable to attack because human behaviors such as viewing dangerous websites, downloading and opening

malicious files from the Web, E-mail, or file transfer storage provide more opportunities for cybercriminals or adversaries. Apart from harmful files, client users on workstations install more applications and drivers than servers, providing a risk if such applications have vulnerabilities or exploits that cybercriminals or adversaries might use the Internet.

The majority of these attacks take advantage of code execution flaws in Microsoft's Windows operating system. Malware takes advantage of these code execution flaws to attack workstations and servers successfully. The top vulnerability is code execution over the last 20 years, according to data from [9]. This vulnerability is twice as much as the second most common one, which is Overflow. Figure 2 depicts the annual number of Microsoft vulnerabilities, whereas Figure 3 depicts the vulnerability types over the last 20 years. These flaws have been weaponized, and malicious software or malware has been packaged to exploit them.

Machine learning strategies are evaluated in this paper to detect malware binaries and executables. The machine learning-based detector's results are then compared against those of a third-party antivirus provider.

RELATED LITERATURE

According to existing research, detection and mitigation approaches range from manual setups to the use of algorithms to the creation of traffic visualization. Machine learning is commonly used in assessing Windows event viewers in studies building algorithms for detecting dangerous software and behavior. [10], which has installed a Sysmon executable in a host to gather logs and implement the Random Forest machine-learning algorithm to detect credential dumping, service creation, scheduled task creation, process injection, and Regsvr32 attacks, is an example of a work that uses machine learning to detect malicious payloads and activities. The study's findings reveal that the threshold is linked to the rate of false positives. This outcome indicates that if a system identifies 100% true positives, the rate of false-positive detection increases. Another paper [11] employs machine learning to detect masquerading or impersonation attempts by monitoring user system behavior patterns. In terms of accuracy in detecting, [10] and [11] in Table I shows that false-positive increases as true positive increases.

In contrast, [12] used Windows Event data to test Support Vector Machine, Isolation Forest, and Local Outlier Factor machine learning algorithms for detecting Golden Ticket and Privilege Escalation attacks. Compared to the other two methods in Table I, the Support Vector Machine approach produces high precision and accuracy outcomes. The Markov machine learning model was then utilized to detect unusual activity in Windows Active Directory logins by [13]. Table I reveals that the accuracy that can be achieved is just 66 percent. [13] concluded that Active Directory log restrictions are to blame for the poor performance in detecting anomalies.

Other works that employ simple methods such as keyword or string filtering and event taggings in Windows event viewer instead of machine learning-based algorithms to detect anomalous behaviors or related attacks include [14], which detects credential dumping and Kerberos authentication attacks using Powershell and honeypots. [15] suggested an algorithm for detecting Golden ticket or privilege escalation threats by filtering event IDs. While [16] employs a pattern mining technique to detect malicious logins or lateral movement attacks, [17] developed a graphic user interface that depicts pattern behavior to detect malicious logins via visual correlation and login events. Future suggestions of these papers may employ a machine learning-based method, as suggested by the authors.

Nair and Sridaran's work [18] 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, is a related work that does not use detection methods but manually configures the environment to enforce protection in the Windows environment. [19] developed the Privileged Account Access Control System application model to protect against a dictionary, brute force, rainbow tables, pass-the-hash, man-in-the-middle, and sniffing attacks. While [20] employed Group Policy Objects (GPO) and Active Directory Services to protect against browser-related vulnerabilities like JavaScript and Plugin-based attacks, as well as Phishing. [21] on the other hand, it implements a principle of least privilege to reduce privilege escalation attacks. While [22] implements a centralized single sign-on (SSO) to protect application credentials. Aside from these manual adjustments to secure the Windows environment, [23] has developed a system to identify machine clients at danger of lateral movement attacks rather than detecting or mitigating Windows-related attacks.

The available efforts on defending and securing the Windows environment from attacks still use a mix of static and manual settings. However, when compared to full-fledged machine learning-based algorithm detection methods, it is still low. Because a resilient or static setup gives a 100 percent result in known or signature-based threats, many security researchers still believe machine learning-based detection produces incorrect findings, according to this literature review. Machine learning-based detection methods, on the other hand, have shown to be capable of detecting unknown or zero-day attacks.

I. Lacking in the Approaches

The related literature cited above lacks the skills to identify unknown or zero-day threats, which this proposal aims to address. The research [18]–[23] use manual settings to mitigate or stop specific attacks, but they cannot protect against unknown or zero-day attacks. Similar works in defending the Windows environment that employs algorithms but does not use machine learning-based algorithms, on the other hand, have similar capabilities in fighting

Windows attacks. In comparison to the research stated above, the sole notable advantage is that it is automated. These studies are as follows: [14], which uses the Splunk Processing Language, only detects attacks that have been programmed into it for querying or filtering, [15], which proposes using machine learning in future research to reduce false positive detection, [16], which uses pattern mining with low accuracy results, and [17], which uses a graphic user interface, proposes exploring algorithm designs for future research.

On the other hand, related research that focuses on machine learning detection approaches, such as [10], extensively relies on Sysmon logs as input parameters, which are supplied to a central logging aggregator or Security Incident and Event Manager (SIEM). Because the Sysmon installed on the server or client must be given to the SIEM, this method adds latency to the results. The study of [11], on the other hand, uses file details as input parameters. This feature or input parameter is insufficient to detect whether a file is malicious or benign. [11] found that at a False Positive Rate (FPR) of 0.2 percent, the accuracy is 54 percent. [12] did an excellent job of evaluating three machine learning methods. On the other hand, the input parameters are individual event ID logs, which are far too simple for a machine learning-based method. This strategy does not work for unidentifiable attacks using event IDs alone, but it may be used to train the machine learning-based algorithm. The study in [13] used Active Directory logs as input parameters, which they recognized were insufficient to detect malicious activities or Windows attacks in their conclusions.

Aside from the insufficient approaches, most studies employed Windows event viewer logs as input features and parameters.

II. Summary

Protecting the Windows environment using human configuration, non-machine learning-based algorithms, and machine learning-based algorithms are the three categories of related research. Manual setup and non-machine learning-based techniques are still preferred by security experts because they provide 100 percent accuracy in detecting known and signature-based attacks. Machine learning-based algorithms are still in the experimental stage. The majority of businesses and organizations that use machine learning-based detection methods do so in addition to classical or signature-based detection approaches. Because the authors analyzed distinct scenarios and employed different input factors, the data generated from these researches primarily differ. The majority of the studies, however, employed the Windows event viewer logs as input parameters. Except for [12] Support Machine Vector, no specific machine learning-based technique has been suggested. The challenge remains in determining the best input feature parameters and machine learning method for high accuracy and a low false-positive rate.

MACHINE LEARNING-BASED DETECTORS

I. Logistic Regression

A machine learning-based detector based on Logistic Regression is used to distinguish between malicious and benign samples. The Logistic Regression model produces a boundary that may be used to determine if a sample is malicious or benign. The loss function of logistic regression is depicted by the negative log-likelihood.

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

where $\{y_i\}$ represents for truth labels and $\{p_i\}$ stands for probability predictions. As a product of each likelihood, the likelihood of all forecasts is indicated below.

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

The goal of logistic regression is to find the best parameters that create probabilities that maximize or optimize the likelihood. A hyperplane is used to identify the binaries or executables in Logistic Regression. The amount of fed or configured features to the logistic regression algorithm, which geometrically differentiates malicious from benign samples, determines the hyperplane. Logistic regression classifies a sample or an unknown binary or executable on the malicious or benign side of the border when it is input into the detector.

Sklearn.Linear_model.LogisticRegression is used in building Logistic Regression machine learning detector [24].

II. Random Forest

A machine learning-based Random Forest detector is used to distinguish between malicious and benign samples. The Random Forest approach depends largely on decision trees, with each decision tree casting a vote to determine whether the sample is malicious or benign. The Random Forest algorithm follows the following steps:

a) A random subset of N samples (trained individual trees) from the training dataset is chosen.
b) 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 \le Y.
c) Do step 2 until each tree is trained.
d) Do steps 1, 2, and 3 until all trees in the forest are trained.

The number of decision tree votes divided by the total number of decision trees determines the likelihood that a binary or executable, whether malicious or benign, is to be identified.

Sklearn.Ensemble.RandomForestCLassifier is used in building the Random Forest machine learning detector [25].

TABLE I
Studies Using Machine Learning Algorithm for Detection

Study	Detection Algorithm	False Positive Rate (%)	Recall (%)	Precision (%)	Accuracy (%)
[10] Sensitivity Plot to Detect New Service Creation	Random Forest	1.5	-	-	Able to detect (Binary Validation)
[11] Masqueraders Detection Accuracy	Gaussian Mixture Model	1	-	-	68
[12] Results for each Algorithm	One-Class SVM	-	100	100	100
	LOF	-	5	7	74
	Isolation Forest	-	43	90	50
[13] Performance Evaluation	Trendmicro and Markov Model	-	66.60	99.07	66.34

III. Support Vector Machines

A machine learning-based Support Vector Machine (SVM) detector is used to distinguish between malicious and benign samples. The loss function differs between SVM and logistic regression in that it forms a hyperplane. Hinge loss is implemented in SVM, which penalizes samples that are solely on the wrong side. Logistic regression, on the other hand, uses a log-likelihood function to penalize all samples according to the probability error estimate. The support vector machine's loss function is shown below.

$$\beta + C\sum_{i=1}^{N}\xi_i \quad (3)$$

where the margin is β, the hyperparameter that is relative to the contribution of the two terms is C, and the distance of the margin to the ith support vector is ξ_i. Sklearn.Svm.SVC is used in building the Support Vector Machine detector [26].

IV. Neural Networks

As indicated in Fig. 4, another algorithm is Neural Networks (NN) or Artificial Neural Networks (ANN). It is a complex network of a critical computational element called a perceptron, which are basic simulations of neurons in the brain. Its architecture and computing are based on a network of completely parallel networks of different computational parts that are systematized in connection to one another. In this type of algorithm, the learning process is visible in some way. It can also deliver accurate and dependable expected outcomes.

The input layer on the left side consists of a set of new neurons x_i which represent the input

$$\{x_i|x_1, x_2, x_3, \ldots, x_n\}. \quad (4)$$

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

$$\{w_i x_i + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n\} \quad (5)$$

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

$$R(z) = \max(0, z) \quad (6)$$

This activation function or ReLU applies a nonlinear transformation on the weighted sum to optimize the parameters with backpropagation, resulting in a linear transformation of the neuron's input data. The output layer, which is on the right side, then obtains the values from the final hidden layer, transforms them, and outputs them.

Sklearn.Neural_network.MLPClassifier is used in building a Neural Network detector [27].

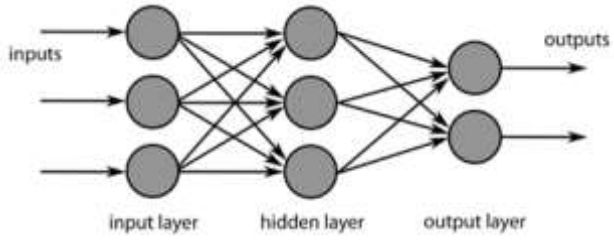

FIGURE 4
The Architecture of the Neural Network

TRAINING AND TESTING

I. Sample Dataset

The benign samples have been copied from Windows 2016 Server System32 folder with the following commands:

- cd c:\Windows\System32
- copy *.dll c:\tmp
- copy *.exe c:\tmp

Samples were transferred to the machine learning-based detector host machine.

On the other hand, malware samples h-ave been collected from https://www.virustotal.com/ and https://github.com/vxunderground.

Table II shows the samples used in training and testing the machine learning-based detector, and Table III shows a summary of the sample dataset. The benign samples are all the available samples contained in the Windows system32 folder. For matching or balancing with the number of benign samples, the same number of malicious samples were gathered. The 14 customized malware samples are specially created to demonstrate the ability of the machine learning-based detectors to detect the binaries and executables, such as zero-day attacks, and compared its detecting efficiency with a 3rd party antivirus vendor.

II. Evaluation Criteria

Each detector's results were evaluated, and the input parameters and settings were fine-tuned to obtain optimum

TABLE II
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	VXunderground	61
For Testing	Customize	Win32	-	14				
Benign	DLL	Win32	-	405				
Benign	Executable	Win32	-	96				

TABLE III
Sample Dataset Summary

Category	Quantity
Malware	501
Benign	501

accuracy results in identifying samples, whether malicious or benign. The Accuracy formula is shown below, where TP stands for True Positive, TN as True Negative, FP as False Positive, and FN as False Negative.

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \quad (7)$$

Testing Scenario 1: Evaluate individual customized malware samples against the Machine Learning-based Detectors.

Testing Scenario 2: Perform train-test data slicing on samples as shown in Table IV. The process in Table IV shows that samples are divided into four, and four experiments are performed. In experiment 1, fold 4 was used for testing. In experiment 2, fold 3 was used in testing. In experiment 3, fold 2 was used in testing, and in experiment 4, fold 1 was used.

Two separate tests have been performed. The first one is the evaluation of the machine learning-based detector to detect the 14 customized malware samples. In this scenario, the machine learning-based detector was trained and tested twice. The first testing included all the available benign and malware samples, excluding the 14 customized malware samples. The second testing including all benign and malware samples, and this includes the 14 customized malware samples. The only testing samples in this first evaluation were the 14 customized malware samples. On the second evaluation, data slicing was applied for training and testing. The benign and malicious samples are sliced or divided for training and testing in the following:

- Test 1 – 50% training and 50% testing
- Test 2 – 66.66% training and 33.33% testing
- Test 3 – 75% training and 25% testing
- Test 4 – 80% training and 20% testing

TABLE IV
Train-test Data Slicing

	Fold 1 (1-250 samples)	Fold 2 (251-500 samples)	Fold 3 (501-750 samples)	Fold 4 (751- 1000 samples)
Experiment 1	Training	Training	Training	Testing
Experiment 2	Training	Training	Testing	Training
Experiment 3	Training	Testing	Training	Training
Experiment 4	Testing	Training	Training	Training

III. Evaluation using 14 Customized Malware Samples for Testing

In Table V, individual malware samples were evaluated. A value of 1 means that the detector could identify that the sample was malware, while 0 means that it could not identify the sample. For Logistic Regression, Random Forest, Support Vector Machine, Neural Network, and VirusTotal, a threshold of 50% accuracy were if the samples are run in this system, and it shows 50% or greater, then the value would be 1, or it can identify the malicious samples. For Avira and Webroot, the samples were just scanned using these installed antiviruses. It can be observed in Table V that the machine learning-based detectors that do not include the 14 customized samples in training performed poorly. The same results were observed in VirusTotal, Avira, and Webroot. This

TABLE V
Detection of Malware Samples Comparison

Samples	Logistic regression - without training	Logistic regression - with training	Random Forest - without training	Random Forest - with training	Support Vector Machines - without training	Support Vector Machines - with training	Neural Network - without training	Neural Network - with training	VirusTotal	Avira	Webroot
reverse.hta	1	1	1	1	1	1	1	1	0	0	0
DLSU.doc	1	1	1	1	1	1	1	1	1	1	0
reverse.bat	1	1	1	1	1	1	0	1	0	1	0
reverse.exe	1	1	1	1	1	1	1	1	1	1	1
powershell.ps1	1	1	1	1	1	1	0	1	0	0	0
putty.exe	1	1	1	1	1	1	1	1	1	1	0
windows-privesc-check2.exe	1	1	1	1	1	1	1	1	0	0	0
juicy-potato.exe	0	1	0	1	0	0	0	1	1	1	1
shell.exe	1	1	1	1	1	1	1	1	1	1	0
mimikatz.exe	0	1	1	1	0	0	0	1	1	1	1
accesschk64.exe	0	1	0	1	0	0	0	1	0	0	0
driverquery.exe	0	1	0	1	0	0	0	1	0	0	0
sigcheck.exe	0	1	0	1	0	0	0	1	0	0	0
PsExec64.exe	0	1	0	1	0	0	0	1	0	0	0

result means that the installed antivirus was not able to identify the malware samples. When the 14 customized samples are included in training with the machine learning-based detector, values show that the machine learning-based detector has been able to identify all malware samples correctly.

IV. Evaluation with Data Slicing

In this evaluation, the available samples have been divided were as shown in Table VI. 50-50 means 50% of the samples were used for training, and the remaining 50% is for testing. 67-33 means 67% of the samples were used for training, and the remaining 33% is for testing. 75-25 means 75% of the samples were used for training, and the remaining 25% is for testing. 80-20 means 80% of the samples were used for training, and the remaining 20% is for testing. For the two antivirus vendors, Webroot and Avira, the samples and sample slicing used in the machine were transferred to the Windows victim machine. The folder where the samples are stored was scanned using the installed antivirus to obtain the antivirus' accuracy.

It can be observed in the mean values that whether the sample for testing is using 50%, 33%, 25%, or 20% of the samples, does impact its accuracy value. Fig. 5 shows that the machine learning-based detectors have outperformed both Webroot and Avira as the mean accuracy values for Webroot and Avira are around 91.64% and 89.34%, respectively. While the mean values for Logistic Regression, Random Forest, Support Vector Machine, and Neural Network are around 99.13%, 99.20%, 95.20%, and 97.29% respectively. This result means that the Random Forest followed by Logistic Regression has performed better than Support Vector Machine, Neural Network, Webroot, and Avira in this data slicing evaluation to identify whether the samples are malicious or benign.

V. Performance Analysis

Section III utilized the 14 customized samples for testing, while section IV performed a data slicing, and samples are divided by 50%, 33%, 25%, and 20% for testing. Results show that Random Forest and Logistic Regression outperform the other machine learning algorithms. Avira and Webroot were also evaluated if these antivirus vendors can detect the malicious samples but have performed poorly compared to the machine learning-based detectors, especially when the 14 customized samples are added in training. Section IV shows that slicing the samples to varying quantities such as 50%, 33%, 25%, and 20% for testing does affect the mean accuracy. It is observed that all machine learning-based detectors have outperformed Avira and Webroot in this scenario. On the other hand, the best performing machine learning algorithm on this is the Random Forest at 99.20%, followed by Logistic Regression at 99.13%.

TABLE VI
Results using Data Slicing

Slicing		Logistic Regression	Random Forest	Support Vector Machine	Neural Network	Webroot	Avira
	Test 1	99.20%	99.20%	97.61%	96.61%	96.80%	93.60%
	Test 2	98.60%	98.80%	92.60%	97.20%	86.45%	85.06%
50-50	Mean	98.90%	99.00%	95.10%	96.91%	91.63%	89.33%
	Test 1	99.40%	100.00%	98.20%	97.31%	97.01%	94.61%
	Test 2	98.80%	99.10%	95.51%	97.31%	97.31%	94.31%
	Test 3	99.40%	99.40%	94.01%	97.60%	80.54%	79.04%
67-33	Mean	99.20%	99.50%	95.91%	97.41%	91.62%	89.32%
	Test 1	99.21%	99.60%	98.41%	96.43%	96.40%	97.20%
	Test 2	99.60%	99.20%	96.80%	98.80%	97.20%	90.00%
	Test 3	98.40%	98.40%	94.80%	96.40%	98.80%	96.40%
	Test 4	99.20%	99.20%	94.40%	97.20%	74.21%	73.81%
75-25	Mean	99.10%	99.10%	96.10%	97.21%	91.65%	89.35%
	Test 1	99.01%	99.50%	99.01%	97.03%	95.50%	96.50%
	Test 2	100.00%	100.00%	96.50%	98.50%	98.50%	88.50%
	Test 3	99.50%	99.00%	95.50%	98.50%	97.50%	98.50%
	Test 4	99.00%	98.00%	95.00%	96.00%	99.00%	96.00%
	Test 5	99.00%	99.50%	94.50%	98.00%	67.82%	67.33%
80-20	Mean	99.30%	99.20%	96.10%	97.61%	91.66%	89.37%

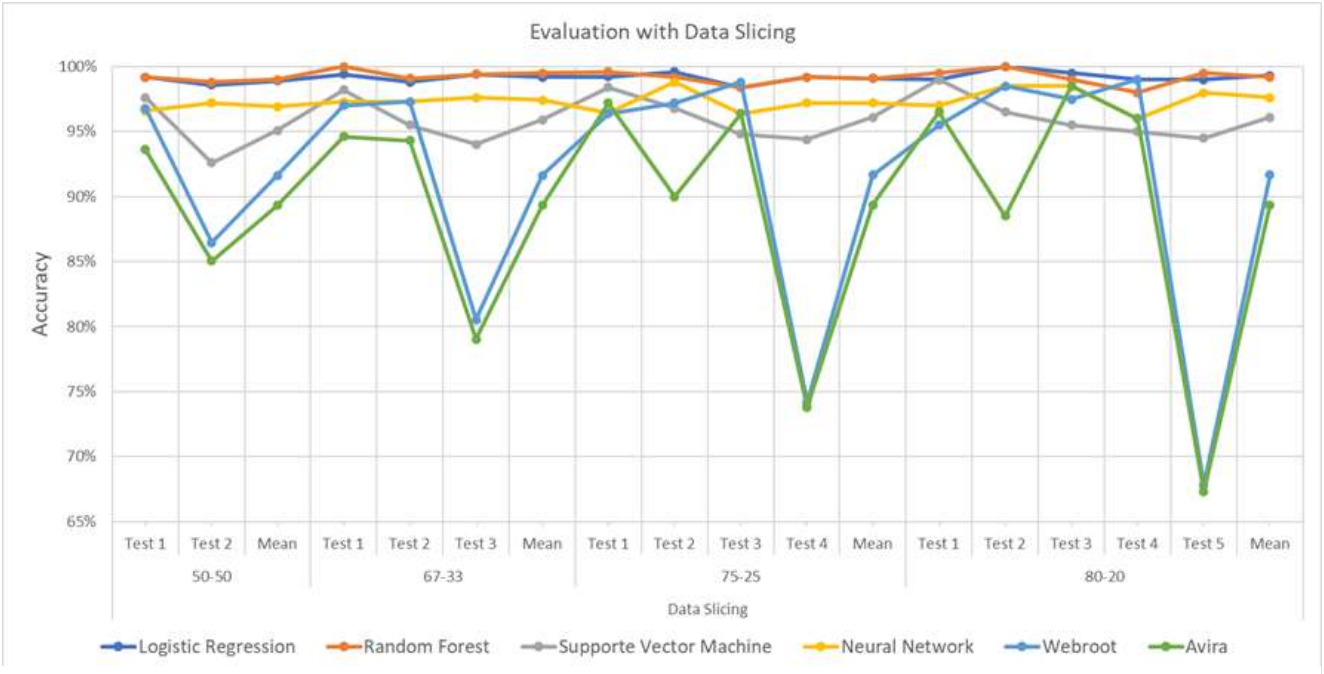

FIGURE 5
Evaluation with Data Slicing Results

COONCLUSION AND RECOMMENDATIONS

In this paper, results show that the created machine learning-based detector performs better in identifying or classifying the malicious samples than current antimalware products. This conclusion is reinforced with the results in Table VI and Fig. 5 when Avira and Webroot were added to the evaluation. The outcomes show that the machine learning-based detector outperformed these antiviruses.

Two testing scenarios were performed. The first scenario was the testing samples only used the 14 customized samples, while the second scenario testing performed a data slicing. For the first scenario, the list of machine learning-based detectors evaluated, with testing samples included in the training, Random Forest and Logistic Regression, performed better in accuracy than other machine learning-based detectors. 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.

For the second scenario, results of the first scenario were reinforced as all four machine learning-based detectors outperformed Avira and Webroot in identifying the samples, whether it is malicious or benign. This conclusion provides the feasibility of the proposal that by implementing machine learning-based detectors, 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.

It is recommended to implement a machine learning-based detector on top of the current antimalware products. A machine learning-based detector enables an organization to detect and block custom or specific attacks that traditional and enterprise antimalware vendors cannot detect and block.

REFERENCES

[1] "Microsoft report shows increasing sophistication of cyber threats," *Microsoft on the Issues*, Sep. 29, 2020. https://blogs.microsoft.com/on-the-issues/2020/09/29/microsoft-digital-defense-report-cyber-threats/ (accessed Oct. 14, 2020).

[2] C. Cimpanu, "Financial sector is seeing more credential stuffing than DDoS attacks," *ZDNet*, 2020. https://www.zdnet.com/article/financial-sector-has-been-seeing-more-credential-stuffing-than-ddos-attacks-in-recent-years/ (accessed Oct. 14, 2020).

[3] "Ransomware Upgrades with Credential-Stealing Tricks," *Dark Reading*, 2020. https://www.darkreading.com/attacks-breaches/ransomware-upgrades-with-credential-stealing-tricks/d/d-id/1336846 (accessed Oct. 23, 2020).

[4] "Another massive ransomware outbreak – or was it?," *Computer Fraud & Security*, vol. 2017, no. 7, pp. 1–3, Jul. 2017, doi: 10.1016/S1361-3723(17)30055-6.

[5] I. Heritage, "Protecting Industry 4.0: challenges and solutions as IT, OT and IP converge," *Network Security*, vol. 2019, no. 10, pp. 6–9, 2019, doi: https://doi.org/10.1016/S1353-4858(19)30120-5.

[6] A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Computers & Security*, vol. 72, pp. 26–59, Jan. 2018, doi: 10.1016/j.cose.2017.08.005.

[7] S. Mansfield-Devine, "Editorial," *Computer Fraud & Security*, vol. 2017, no. 7, p. 2, Jul. 2017, doi: 10.1016/S1361-3723(17)30056-8.

[8] "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 (accessed Oct. 14, 2020).

[9] "Microsoft : Products and vulnerabilities," 2020. https://www.cvedetails.com/vendor/26/Microsoft.html (accessed Oct. 24, 2020).

[10] J. W. Mikhail, J. C. Williams, and G. R. Roelke, "procmonML: Generating evasion resilient host-based behavioral analytics from tree ensembles," *Computers & Security*, vol. 98, p. 102002, Nov. 2020, doi: 10.1016/j.cose.2020.102002.

[11] J. Voris, Y. Song, M. B. Salem, S. Hershkop, and S. Stolfo, "Active authentication using file system decoys and user behavior modeling: results of a large scale study," *Computers & Security*, vol. 87, p. 101412, 2019, doi: https://doi.org/10.1016/j.cose.2018.07.021.

[12] W. Matsuda, M. Fujimoto, and T. Mitsunaga, "Detecting APT Attacks Against Active Directory Using Machine Leaning," in *2018 IEEE Conference on Application, Information and Network Security (AINS)*, Nov. 2018, pp. 60–65. doi: 10.1109/AINS.2018.8631486.

[13] C. Hsieh, C. Lai, C. Mao, T. Kao, and K. Lee, "AD2: Anomaly detection on active directory log data for insider threat monitoring," in *2015 International Carnahan Conference on Security Technology (ICCST)*, Sep. 2015, pp. 287–292. doi: 10.1109/CCST.2015.7389698.

[14] L. Kotlaba, S. Buchovecká, and R. Lórencz, "Active Directory Kerberoasting Attack: Monitoring and Detection Techniques," 2020.

[15] M. Fujimoto, W. Matsuda, and T. Mitsunaga, "Detecting Abuse of Domain Administrator Privilege Using Windows Event Log," in *2018 IEEE Conference on Application, Information and Network Security (AINS)*, Nov. 2018, pp. 15–20. doi: 10.1109/AINS.2018.8631459.

[16] H. Siadati and N. Memon, "Detecting Structurally Anomalous Logins Within Enterprise Networks," 2017, pp. 1273–1284. doi: 10.1145/3133956.3134003.

[17] H. Siadati, B. Saket, and N. Memon, "Detecting malicious logins in enterprise networks using visualization," in *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Oct. 2016, pp. 1–8. doi: 10.1109/VIZSEC.2016.7739582.

[18] H. Nair and R. Sridaran, "An Innovative Model (HS) to Enhance the Security in Windows Operating System - A Case Study," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019, pp. 1207–1211.

[19] E. Sindiren and B. Ciylan, "Application model for privileged account access control system in enterprise networks," *Computers & Security*, vol. 83, pp. 52–67, 2019, doi: https://doi.org/10.1016/j.cose.2019.01.008.

[20] A. Jillepalli, D. Conte de Leon, F. T. Sheldon, and M. Haney, "Enterprise-level Hardening of Web Browsers for Microsoft Windows," vol. 7, pp. 261–274, 2018, doi: 10.12785/ijcds/070501.

[21] A. Binduf, H. Alamoudi, H. Balahmar, S. Alshamrani, H. Al-Omar, and N. Nagy, "Active Directory and Related Aspects of Security," 2018, pp. 4474–4479. doi: 10.1109/NCG.2018.8593188.

[22] H. Wang and C. Gong, "Design and Implementation of Unified Identity Authentication Service Based on AD," in *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec. 2016, pp. 394–398. doi: 10.1109/CICN.2016.84.

[23] S. Freitas, A. Wicker, D. H. Chau, and J. Neil, "D2M: Dynamic Defense and Modeling of Adversarial Movement in Networks," 2020.

[24] "sklearn.linear_model.LogisticRegression — scikit-learn 0.23.2 documentation," 2020. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed Dec. 13, 2020).

[25] "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 (accessed Dec. 13, 2020).
[26] "sklearn.svm.SVC — scikit-learn 0.24.2 documentation," 2021. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC (accessed May 16, 2021).
[27] "sklearn.neural_network.MLPClassifier — scikit-learn 0.24.2 documentation," 2021. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier (accessed May 16, 2021)