Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Recealer: A malware detection method based on machine learning and deep learning models

Asia Othman Aljahdali *, Elaf Maqadmi, Atouf Ghabashi, Deem Alsuoilme, Bayader Alluhaybi, Edmy Alboqami

College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 22 April 2025
Received in revised form
26 August 2025
Accepted 3 September 2025

Keywords:
Malware detection
Machine learning
Deep learning
Hybrid model
Feature extraction

ABSTRACT

Raccoon Stealer malware is difficult to detect as it actively evades traditional methods. This study proposes Recealer, a hybrid detection model that integrates machine learning (ML) and deep learning (DL). The model applies static and dynamic analysis to extract features from sample files, which are first classified using an ML algorithm. Files with uncertain classification results are then transformed into grayscale images and analyzed by a convolutional neural network for improved precision. Experimental results demonstrate that Recealer achieves 94% overall detection accuracy, with the random forest algorithm attaining 97.53% in the ML stage and the DL stage reaching 95%. These findings indicate that Recealer is both efficient and reliable for detecting Raccoon Stealer malware.

1. Introduction

In the late 1980s, malware began spreading rapidly across computer and network systems, enabling attackers to carry out various malicious activities. Malware, short for malicious software, is software designed with the intent of disrupting or acquiring unauthorized access to a computer system. As a result, it is critical to detect and respond to malware accurately and swiftly. Raccoon Stealer is an indistinct info-stealer malware that was first discovered in 2019 and sold as Malware-as-a-service (MaaS) on underground forums. MaaS is a business model used by cybercriminals to charge users for access to malicious software and support infrastructure. A new version of the malware, known as Raccoon StealerV2, was released in 2022, but considering it is relatively new, it hasn't been properly identified and eliminated. This malware is a trojan that infiltrates a system to capture an immense amount of data, including credit cards, usernames, passwords, history, cookies, auto-filled browser passwords, cryptocurrency wallets, and other sensitive data. The victim's information is often collected with the intent to sell it on the dark web. For instance, one marketplace sold off 4,368,909 units of victim data between January 2021 and March 2022. Info-stealer victims get infected by malicious actors; they use phishing emails (most commonly), cracked and pirated software, cheating packages for games, browser extensions, and cryptocurrency-related software. For example, one info-stealer spreads phishing emails by sending Excel spreadsheets through emails that contain macros, which results in downloading a persistent executable (Nurmi et al., 2023).

To reduce the malware's impact, the structure of the malware must be recognized to fully comprehend its characteristics and behaviours; this can be achieved by thoroughly analysing the malware. Comprehending the actions and intentions of a malware structure to stop further cyberattacks is known as malware analysis.

This research is being conducted to analyze and detect Raccoon Stealer malware. Since most detection methods are signature-based, many of them cannot discover the behavior of the malware. Research indicates that malware lacking a signature exhibits certain behavioral traits at a higher level and is more accurate in exposing the true intention of the malware. Static analysis, which will assist in learning the characteristics and structure of the malware without running it, and dynamic analysis, which will be used to execute the malware and observe its functionality in the test environment (Yucel et al., 2021), In order to detect new malware efficiently, both static and dynamic analyses will be used, and the features generated from them will be used to train the machine learning (ML) model. In addition, deep learning will be implemented to guarantee a higher level of malware detection.

* Corresponding Author.
Email Address: aoaljahdali@uj.edu.sa (A. O. Aljahdali)
https://doi.org/10.21833/ijaas.2025.10.005
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-9013-9465

According to a study, Windows is one of the most widely used operating systems, and an excessive amount of malware targets it compared to other operating systems. Consequently, it is now more crucial than ever to create efficient methods for detecting Windows malware (Prachi et al., 2023). The Raccoon Stealer's payload is intended to infect Windows-based 32-bit and 64-bit PCs. This research proposes a Recealer malware detection technique model, which performs the following:

- Taking advantage of both dynamic and static features of the malware to understand the malware behavior.
- Utilizing the properties of machine learning for faster detection and deep learning for precise and accurate results.
- Providing a brief report or alert after handling the file.

The proposed model aims to assist in malware detection by enhancing established techniques and methods used by previous researchers.

Malicious software, or malware, is designed to damage computer systems and programs. Trojan horses, worms, viruses, and malware are just a few of the various forms it can take. Viruses are a segment of code that attaches to a file or program. Viruses replicate themselves by altering other computer programs and infecting them with their bits of code. This happens silently when a user runs an infected program. Worms are programs that duplicate themselves, run separately from other programs, and move from computer to computer across a network without human interaction. The primary difference between worms and viruses is that worms function independently, whereas viruses always hide within software and need human interaction to be activated. Spyware is installed on a computer; it gathers data about users' online and offline activities and sends it back to a central location (Saeed, 2020). A Trojan horse is hidden within another program that seems innocuous; a Trojan horse can take control of the computer (Sreekumari, 2020). Because Trojans are not identical in any way, it can be challenging to tell the difference between files that could be risky and ones that are not. Also, new Trojans are being written continuously, and because of that, their signatures differ from the ones that are already identified.

Raccoon Stealer first emerged in 2019, acting as one of the most impactful information-stealing malware. A new enhanced version in 2022 was released, which has become more sophisticated and cumbersome to detect by security technologies. Recealer is initially delivered to the target user by applying manipulative techniques that urge the victim to click, download, and execute received attachments on which the malware resides. This transmission could be done through phishing emails, fake websites, and other allurement tactics. Once the Recealer arrives in the Windows system, it establishes its command and control (C2) channels that are used for extracting valuable information and data from the victim. C2's channel-specific source and destination information are hidden in the executable of the Recealer itself by Rivest cipher 4 (RC4) encryption and Base64 encoding. The raccoon will then be injected into the process level of the computer to execute itself by seeming benign, as these locations are mostly run by the operating system. To further proceed with the information extractions, dynamic link libraries are going to be imported and used to find application cache default locations that store credentials such as usernames and passwords, browser cookies, email agents, digital currencies, and other sensitive data. These caches are copied to a temporary file location that can be encrypted and extracted to the C2 covert channels. Detecting and stopping those actions conducted by the raccoon stealer is essential to mitigating further attacks that were initially aided by the collected information, as this will also help protect the victims' privacy.

Malware detection refers to the collection of defensive methods and tools needed to identify and prevent the negative impacts of malware. Depending on the kind of malware that infected the device, this preventive strategy consists of a wide variety of techniques that are strengthened by different technologies. Such techniques include behavior-based detection and signature-based detection. Signature-based detection is a technique that utilizes a database of known malware signatures and compares them with suspicious files or network traffic. A signature is a unique byte pattern, or fingerprint, that identifies a specific malware. Despite its rapid and precise identification of known malware variants, it has several drawbacks. To start, it is unable to identify brand-new or unidentified malware that does not yet have a signature in the database.

To stay up to speed with the ever-changing environment of malware, the signature database needs to be updated frequently. Secondly, malware that modifies or obscures its signature to evade detection can successfully deceive signature-based detection. Behavior-based detection is a novel approach for cybersecurity that uses user behavior monitoring to identify and block malicious activity. It's a proactive strategy that keeps a watchful eye on every relevant activity to swiftly spot and handle any alterations to typical behavior patterns. Beyond identifying specific attack signatures, behavior-based detection can discover and examine unusual or malicious patterns of behavior. This kind of technology analyzes immense amounts of data and network traffic using machine learning, artificial intelligence, and statistics to identify anomalies. Behavior-based detection increases the probability of discovering and terminating a malicious action before the network becomes compromised by monitoring behaviors that may be linked to the attacker, as opposed to looking for patterns associated with types of attacks. Although both techniques can have satisfying results, especially

when combined, malware detection techniques can be enhanced by applying these two approaches: threat intelligence and threat hunting.

Threat hunting is a proactive approach to cybersecurity in which specially trained personnel, referred to as threat hunters, actively seek out, identify, and isolate sophisticated threats that evade existing security measures. Instead of waiting for a warning to act, it actively looks for any hidden threats that may be present in the system. Threat hunting's primary objective is to identify and eliminate threats before they have a chance to cause damage. It's about reducing the possible damage of an intrusion and remaining one step ahead of the attackers. Whereas threat intelligence, a reactive approach, is the information gathered that gives the ability to stop or mitigate cyber threats. It's all about getting to know the enemy—who they are, what motivates them, and what techniques they employ. Having this knowledge is essential for creating proactive security plans and techniques. The goal of threat intelligence is to comprehend the threat environment. It's about being aware of possible attackers, understanding their strategies, and being prepared for their attacks.

Analyzing malware is the first step towards stopping it and has several benefits, including obtaining sufficient data regarding the malware, implementing adequate defense and response mechanisms, assessing the capability of detecting malware, and understanding the risks associated with malware and its intentions. It is possible to analyze malware in many ways, including static code analysis, where the analysis is performed without execution; dynamic code analysis, in which the code line is executed with the help of the debugger; and behavioral/dynamic analysis, in which malware is executed, and a variety of information is gathered based on its interactions with the environment.

CUCKOO SANDBOX is a dynamic malware analysis tool that provides a secure and isolated environment. Additionally, it provides detailed reports on malware samples, and its design allows for integration with external tools and platforms. Using ANY.RUN, researchers can interact with malware samples and analyze their activities through this powerful interactive malware analysis platform. It has many features, such as network tracking, process monitoring, and so much more. PROCESS HACKER is an open-source malware analysis tool that helps identify the processes affected by malware in the system and provides real-time data on the system and network usage. GHIDRA is also an open-source static analysis tool that was recently released. It is used to disassemble malware, allowing users to examine malware's functionality to gain a better understanding of it. Lastly, X64DBG for Windows systems is a dynamic code analysis tool with a debugger that can be used to disassemble malware and execute it line by line (Bermejo Higuera et al., 2020).

Over the last few years, machine learning has been increasingly utilized to assist people in a variety of fields, including healthcare to detect diseases, streets to detect speeding cars, finance to predict the stock market, and many others. Future predictions, automation of numerous tools, and reducing human error are some of its main benefits. As a result of its integration into the cybersecurity field, many tools and devices have been improved. Deep learning (DL) is a specific area of machine learning that uses artificial neural networks to interpret raw data directly. In fact, deep neural networks make it possible to create end-to-end prediction models by handling every processing step, including feature extraction and learning that is often required to create a classic machine learning model. DL models can automatically extract imaging features to optimize the model's performance for the given task. Representation-learning algorithms known as deep neural networks are made up of a stack of processing layers with a finite number of nonlinear units (Castiglioni et al., 2021). Some of the deep learning methods are recurrent neural network (RNN) and convolutional neural network (CNN). Recurrent Neural Networks: at any input stage, the RNN considers the results of earlier calculations along with the present circumstances. By using these techniques, the model can be trained with the least amount of data loss possible (Elsayed et al., 2020). Convolutional Neural Networks (CNNs) are neural networks that combine deep learning functionality. It possesses the self-adaptive ability to encode the combination of less important data aspects into more important ones, and the feature extraction process is integrated into the calibration procedures (Chen et al., 2020).

2. Literature review

The malware analysis and intelligence tool (MAIT) tool (Yucel et al., 2021) consists of 4 main parts: the analysis package, the reporting package, the cyber threat intelligence (CTI) generator, and the dispatcher, which will connect the packages. What concerns this project most is the analysis package, which includes the interfaces with the chosen tools for sandboxing and analysis, as well as scripts for administering the sandboxing virtualization tool.

A system that combines static analysis and machine learning to detect malware in Windows operating systems existed in past research (Hussain et al., 2022). The system is divided into four layers: data acquisition layer, pre-processing layer, prediction layer, and performance evaluation layer. The malware data used from static analysis is the PE header information that will be used to train the models. The models' performance was assessed using a variety of measures, such as accuracy, precision, recall, F-score, and support. Random forest (RF) and decision tree (DT) algorithms gave the best results compared to other algorithms.

Sun et al. (2022) presented a novel approach with the ability to detect malware based on system calls in real-time, where both machine learning and deep learning have been combined to get fast

detection features of ML and accurate results of DL. In this research, the performance was based on accuracy, precision, recall, F1 score, and FP rate. The first step was to measure performance without combining the DL and ML algorithms using different hyperparameter values. The DL model DEEPMalware, among the different models, performed the best. Secondly, the performance evaluation was conducted when combining both ML and DL algorithms; the performance evaluation was based on using specific interval values. As a first step, ML will be used to classify a sample, and if the value received by the classifier is less than the lower bound, then the sample is considered benign; if it is higher than the upper bound, it will be considered malicious; and if it is within the interval, the DL model will be used to evaluate it.

Rabadi and Teo (2020) proposed a malware detection and type classification approach that includes four main components as follows: 1. Behavior monitoring: The behavior of the malicious and benign samples is monitored while executing them in an isolated virtual machine using Cuckoo Sandbox3. 2. Feature extraction and generation: This component aims to extract API (application programming interface)-based features and prepare them for the next step (e.g., machine learning). There are two methods for extracting API-based features. In Method 1, each API call and its arguments are treated as one token. Method 2, on the other hand, treats each argument element of each API call as a separate token. 3. Malware detection using machine learning algorithms: This component includes five different machine learning algorithms (e.g., Support Vector Machine (SVM), Extreme Gradient Boosting (XGBoost), Random Forest (RF), DT, and Passive-aggressive (PA)) that are trained using the API-based dynamic features (i.e., bit-vectors from the previous step), resulting in classifying samples as benign or malicious. 4. Malware type classification using machine learning algorithms: After detecting the malware samples in the previous step, the bit vectors from Methods 1 and 2 are utilized to train the same five machine learning algorithms (SVM, XGBoost, RF, DT, and PA), which result in the classification of malicious samples into their respective classes.

The malware detection technique in Singh and Singh (2020) consisted of four primary components: runtime behavior, feature extraction, feature processing, and classifier training. The runtime activities describe the behavior of an executing file. Both benign and malicious samples are run in a controlled setting. Using the Cuckoo Sandbox, the activities are recorded while the binary files are being executed. The runtime features are chosen by the feature extraction module from the generated analysis reports. It has been observed that malicious files differ from benign ones in terms of printable strings. Additionally, they discovered several odd strings in behavior reports, such as earning money, winning a lucky present, or downloading free software or movie content. Processing of the extracted features results in the following feature sets: (i) Printable string information (PSI); (ii) API calls; and (iii) registry, file, and network activities. Many machine learning algorithms are applied across the feature sets to construct the malware detection system. The proposed strategy is assessed using the 10-fold cross-validation method. Ten sections make up the original dataset in this evaluation method. The first nine are used for training, while the tenth is used to test the outcomes. To consider every aspect of the feature set for training and testing, this process is repeated ten times. This assessment method prevents the classifier's overfitting and underfitting issues. Several performance indicators, including F-scores, accuracy, recall, and precision, are utilized to assess the effectiveness of the suggested strategy.

Malware analysis intermediate language (MAIL) (Alam et al., 2013) is a new proposed intermediate language for malware analysis, which can improve the detection of metamorphic malware. The majority of malware enters a computer system through binaries, which are computer-interpretable and executable instructions. Any assembly language has hundreds of different instructions. To maximize the static analysis of any such assembly program for malware detection, we must significantly decrease and simplify these instructions. MAIL gives an assembly program an abstract representation, enabling a tool to automatically analyze and detect malware. Annotated patterns are included in every MAIL statement, which a tool might exploit to maximize malware analysis and detection.

Pant and Bista (2021) have used a grayscale malware images dataset to train and validate different deep learning models to be able to classify malware samples into different families. Table 1 presents a comparison of the current studies that use the same components suggested in this research technique.

3. Proposed solution

3.1. System architecture

The proposed model architecture and its details are depicted in Fig. 1, which is implemented through two different stages. Two datasets are constructed, one for ML and the other for DL.

In the first stage, static and dynamic analysis are conducted on the raccoon stealer samples to create the dataset needed for training and testing the machine learning model. Cuckoo Sandbox is used for conducting such analysis. The ML model will eventually classify the input file as malware if the value generated by the classifier is higher than the maximum assigned interval value; otherwise, if it is less than the minimum assigned interval value, it will be classified as benign. These interval values are going to be defined based on many performance metrics, and the chosen pair with the best values is used. In the second stage, if the value generated by the ML classifier has fallen between the lower and

the upper interval value, the file would be converted to a grayscale image and further subjected to the CNN deep learning model to identify the image as malware or benign. The dataset that is used in the training and testing of the CNN model is generated and consists of malware and benign file images. The model's workflow is described below:

- Step 1: Collect Recealer malware samples from the "Malware Bazaar" website.
- Step 2: Run the Cuckoo Sandbox tool to collect static and dynamic features from the collected samples that will be used to form datasets.
- Step 3: For the development of the deep learning dataset, we will convert the Raccoon Stealer malware samples along with benign files into images and use these images to create the dataset to train and test the CNN model.
- Step 4: The ML datasets will be divided into two parts; one will be used for training the model, and the other will be used for testing. For the ML model, a classification probability will be generated, from which a decision will be made. Depending on an interval value (lower, upper), three different scenarios can occur. If the value generated by the classifier is less than the lower value, it will be considered a benign sample; else, if it is higher than the upper value, it will be malicious, and action will be taken to delete it.
- Step 5: The sample would be further subjected to the deep learning model if its value was located within the interval. The best interval value will be decided based on many performance evaluation metrics, such as accuracy, precision, recall, and F1 score.
- Step 6: The file will be converted to a grayscale image for the deep learning model that has already been trained on benign and Raccoon Stealer malware images. Portable Executable (PE) files, which are usually associated with Windows operating systems, can be represented by binary code that can be translated as an 8-bit unsigned integer vector. This vector may be organized into a 2-dimensional array to generate a grayscale picture that represents the malware. This representation assigns grayscale tones inside the (0, 255) range, where 0 represents black and 255 represents white (Fig. 2).
- Step 7: The deep learning model will identify if the file is benign or malicious, and if it is malicious, it will take action to delete it.
- Step 8: A report is generated for the user, giving information such as whether the file was malware or benign, etc.

Table 1: Current malware detection analysis

Reference	Components/methods	Strengths	Limitations
Yucel et al. (2021)	Dynamic and static analysis (Cuckoo Sandbox, Radare2, Regular Expressions)	Automated tool integrating static and dynamic analysis. Radare2 extracts static features. Regex identifies indicators of compromise (IoC).	Generates enormous data, causing overhead for analysts.
Hussain et al. (2022)	Static analysis, ML algorithms (RF, DT, GB, AdaBoost, SVM, Gaussian naive bayes (GNB))	Random Forest achieved 99.4% accuracy. Provides real-time scanning of executables.	Lacks integration with dynamic analysis, which could improve outcomes.
Sun et al. (2022)	ML (RF, XGBoost, AdaBoost), DL (DEEPMalware), Dynamic analysis	Real-time malware detection. Efficient combination of ML and DL under resource constraints.	Complex framework; risk of recurrent loops between ML and DL algorithms.
Rabadi and Teo (2020)	Ml algorithms (SVM, XGBoost, RF, DT, PA), Cuckoo sandbox, Hashing vectorizer	Resilient to mutation/obfuscation (not reliant on API call order/frequency).	Small number of misclassifications leading to false positives.
Singh and Singh (2020)	Cuckoo sandbox, Text mining, Singular value decomposition (SVD), Shannon entropy, Ml (K-nearest neighbors (KNN), Naive Bayes (NB), SVM, RF, DT, AdaBoost, GB)	High accuracy (99.54%) using string components + SVD for classification.	Limited prior use of string features → uncertainties in scalability/resilience. Accuracy only reported, lacking detailed False Positive Rate (FPR) and the False Negative Rate (FNR) evaluation.
Alam et al. (2013)	Google rapid response (GRR), Wireshark, Virustotal, Static analysis	Control flow graph (CFG)-based behavioral signatures improve malware classification.	Only static analysis, not dynamic. Patterns alone insufficient—unknown malware may go undetected.
Pant and Bista (2021)	Malware as gray-scale images, CNN models (VGG16, ResNet-18, InceptionV3, Custom)	Achieved high accuracy and low loss in image classification.	Unrealistic for real-world malware (files not naturally images). Missing method for converting malware into images.

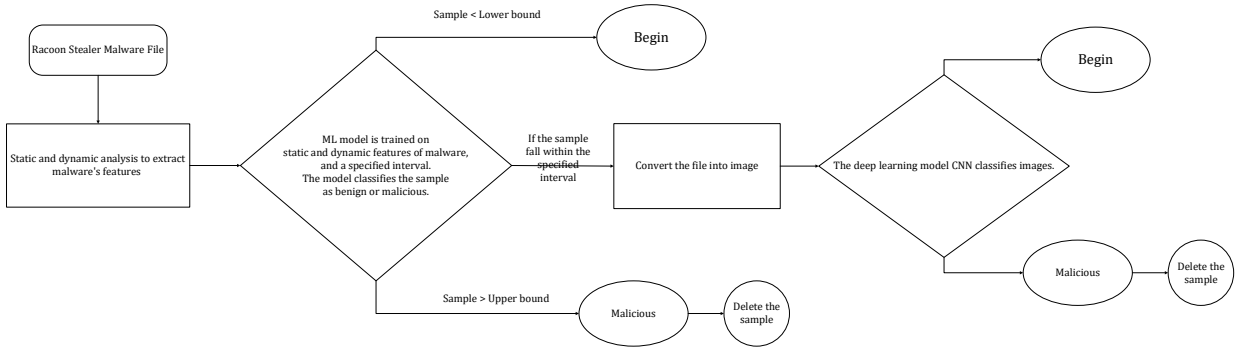

Fig. 1: Recealer detection model architecture

The proposed approach provides several important advantages. It employs a machine learning model that combines static and dynamic features to achieve rapid classification. In cases where higher accuracy is needed, a deep learning model trained on grayscale images can be applied for more reliable

results. The model also incorporates an interval-based decision-making method, which categorizes files by comparing classification probabilities with defined threshold ranges. These thresholds can be adjusted and fine-tuned to meet specific requirements or performance standards. Furthermore, the use of deep learning analysis allows for a more comprehensive understanding of the files, ensuring that malicious files are quickly identified and removed to prevent system or network damage. For implementation, Cuckoo Sandbox is used to execute the Raccoon Stealer malware in an isolated environment, producing a JSON report for further analysis. From this report, features such as API call frequencies, SHA256 hash values, and file entropy are extracted to support both static and dynamic analysis. Nested virtualization is enabled through hypervisors, including VirtualBox, VMware Workstation Pro, and Parallels Desktop, while Ubuntu Linux serves as the operating system for running the sandbox. The machine learning and deep learning models are developed and executed using Python within integrated development environments such as Spyder and PyCharm.

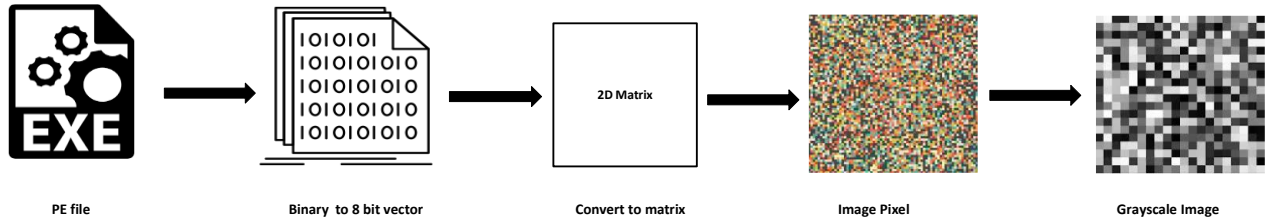

Fig. 2: Conversion of a PE file to a grayscale image

3.2. Sample collection

The collection of samples is an essential step in developing effective machine learning (ML) and deep learning (DL) models. Since Raccoon Stealer malware is relatively new, no publicly available datasets exist for it. To address this, this research constructs datasets using malware samples obtained from "Malware Bazaar," a reliable repository offering a variety of malware samples. Additionally, benign files are sourced from GitHub to ensure balanced datasets. The proposed solution utilized three datasets: one for the ML model, another for the DL model, and a third to evaluate the integrated ML and DL model. The Machine learning dataset contains static and dynamic features extracted from 200 Raccoon Stealer samples and 200 benign samples, totaling 400 samples. Feature extraction is performed using the Cuckoo Sandbox malware analysis tool. This dataset is used for training and testing the ML model. For the DL model, the research converts 275 raccoon stealer malware files and 275 benign files into grayscale images (both sizes of 96x96 and 192x192), creating a dataset of 550 samples. These images are used to train and test the DL model. To test the final integrated model combining ML and DL approaches, a separate dataset consisting of 60 samples (30 benign and 30 malware) is used. These datasets enable comprehensive experimentation and evaluation of the proposed models, ensuring robust performance against the Raccoon Stealer malware.

4. Evaluation and discussion

4.1. Performance evaluation for standalone ML and DL models

We evaluate the performance of seven different machine learning algorithms as part of the process of selecting the most efficient algorithm for our detection model. These models are Random Forest Classifier, Decision Tree Classifier, K-Neighbors Classifier, AdaBoost Classifier, Stochastic Gradient Descent (SGD) Classifier, Extra Trees Classifier, and Gaussian Naive Bayes Classifier. The evaluation is conducted using metrics such as precision, recall, F1 score, accuracy, and the confusion matrix. The performance results are summarized in Table 2, which shows the performance evaluation results of all the ML algorithms used. By observing the values, we can conclude that the best-performing models were the Random Forest Classifier, Extra Trees Classifier, K-Neighbors Classifier, and AdaBoost Classifier. It is noteworthy that the accuracy, precision, recall, and F1 score of the Extra Trees Classifier are like those of the Random Forest Classifier. However, the Random Forest Classifier was chosen to detect Raccoon Stealer malware due to its robustness, consistent high performance, and common use in practical malware detection scenarios. Its ability to handle imbalanced datasets and complex data patterns made it the best choice for this research. The random forest classifier outperformed the rest of the models based on achieving 98.44% overall accuracy, which is like the SGD and Extra Trees classifiers. Random forest's precision is 100% for class 0 and 97% for class 1, in addition to a recall score of 97% for class 0 and 100% for class 1. Both metrics ensure the reduction of false positives and false negatives. Based on cross-validation score values, both random forest and SGD achieved 97.81%, but Extra Trees yielded a score of 97.50%. The decision tree classifier achieved 100% accuracy on all metrics, which wasn't selected due to overfitting; unlike random forest, it comprises many decision trees. Random forest has a feature of tuning bootstrapping and tree weights, which makes it the most reliable algorithm for machine learning malware detection.

Both AdaBoost and Decision Tree achieved 100% precision, recall, and F1-score for both classes, indicating zero false positives and zero false negatives. This makes them highly reliable for malware detection in this dataset. In the case of Random Forest, SGD, and Extra Trees models, these models achieved strong precision and recall scores (~97–100%) for both classes, but they showed minor misclassification. For example, Random Forest and SGD had a recall of 97% for class 0, indicating that ~3% of benign samples were falsely flagged as malware (false positives). Similarly, a precision of 97% for class 1 implies that ~3% of predicted malware samples were benign (false positives for class 1). Among the models evaluated, AdaBoost and Decision Tree achieved perfect classification, minimizing both false positives and false negatives. In contrast, Gaussian Naive Bayes demonstrated poor performance with high false positive rates and limited reliability.

When evaluating the deep learning model, we tested both 96x96 and 192x192 grayscale images of benign and malware samples. In the end, both image sizes obtained the same accuracy of 95%, as indicated in Tables 3 and 4. However, we decided to proceed with the 96x96 image size due to its superior efficiency, as it significantly reduces processing time compared to the 192x192 option.

The outcomes presented in Table 3 compare a Convolutional Neural Network (CNN) trained with Spyder on images of 96x96 pixels. Overall accuracy, F1 score, precision, and recall are the evaluation measures used for this research. In this case, malware was classified as class 1, while benign was classified as class 0. Every trial shows a significant improvement in CNN performance. Trial 3 had the highest F1 scores for both classes: 96% for class 0 and 95% for class 1. Similarly, overall accuracy stabilizes at 95% in trials 2 and 3.

The CNN model's performance across three trials demonstrates strong predictive ability, with overall accuracy ranging from 95% to 97%. Trial 3 shows the best overall performance. The recall for both classes is very high (98% for class 0, 96% for class 1), indicating minimal false positives and false negatives. Trial 3 achieved the best balance between precision and recall, minimizing both false positives and false negatives.

4.2. Performance evaluation for combined ML and DL models

Based on the evaluation of each machine learning algorithm, the ML algorithm with the best metrics will be used (Random Forest), along with the CNN deep learning algorithm. By combining these two algorithms together, we will evaluate their performance using different borderline values inspired by research (Prachi et al., 2023) and choose the best pair. As shown in Table 5, four pairs of values are set in the experiment. The lower bounds represent a boundary that, when exceeded by a lower percentage, the file is classified by the ML algorithm as normal, whereas the file will be classified as malicious when the percentage is greater than the upper bound. The upper value of 60% and the lower value of 40% produce the best-fitting choice as an interval. According to the testing results shown in Table 5, the performance metrics with 94% accuracy, 92% precision, 97% recall, and an F1 score of 94% surpass the rest of the tested pairs; therefore, this interval is selected.

The results presented in Table 5 assess how various borderline thresholds (both lower and upper bounds) perform when combining machine learning (ML) and deep learning (DL) algorithms for detecting raccoon stealer malware. The evaluation metrics include precision, recall, F1 score, and overall accuracy. Here, class 0 denotes benign, while class 1 denotes malware. In conclusion, finding an optimum borderline range like 40%–60% remarkably improves the trade-off between Precision and Recall and hence increases the robustness of detection. Wider ranges like 10%–90% yield poor recall for some classes, while narrower but less optimal ranges, such as 20%–80% or 30%–70% still managed to improve but could not achieve the highest performance levels. The optimum range was 40%–60%, which has the best trade-off, yielding high scores of precision, recall, F1 score, and overall accuracy, thus making it the optimum configuration for classification in this research.

5. Discussion

A malware detection system that combines machine learning (ML) and deep learning (DL) models provides a reliable way to differentiate between harmful and benign files. The proposed Recealer model and PROPEDEUTICA (Sun et al., 2022) show remarkable effectiveness by attaining an astounding 94% accuracy rate. The Recealer DL model's 94% accuracy was marginally lower than the work presented in Pant and Bista (2021), which achieves 98.07% accuracy. However, this disparity might be mostly caused by differences in the number of test samples used, as their model was tested on 1868 samples compared to the 550 used in our study. The comparison of machine learning models showed a similar pattern. Singh and Singh (2020) claimed a greater accuracy of 99.54%, while our machine learning model reached a noteworthy accuracy of 97.53%. Again, this disparity can be attributed to the much larger dataset they used, which included 16,489 malicious files in addition to 8,422 benign files. Even with varying dataset sizes, our models—both machine learning and deep learning—performed quite well despite these variances, suggesting the potential of these techniques in malware detection.

6. Conclusion

The Raccoon Stealer and its variant V2 are recognized to be the most alarming information-stealing malware in 2022. After reviewing several

studies in malware static and dynamic analysis, the integration of ML and DL models proposes an assertive solution to distinguishing between a benign and a legitimate malicious file. For this solution, a dataset is constructed and used to attain promising results that lead to the detection of Recealer's malicious behavior by bringing together ML and DL using a pretested interval of choice. The CNN deep learning model has proved to be reliable in terms of ML Random Forest's unfamiliar interval outcome, which was aided by training the model using grayscale images. This approach studied the Recealer's behaviors on a deeper level of neural networks. Due to the time constraints of implementing this research project, the Cuckoo Sandbox could've served as a tool by integrating it into the ML model.

The mentioned integration serves a purpose in the automation of real-time malware detection, which enhances a better solution for advanced malware detection. For future work, we intend to incorporate explainable artificial intelligence (XAI) and federated learning (FL) into our malware detection framework. This integration will enhance the interpretability of our detection processes while maintaining data privacy across distributed systems.

Table 2: Performance evaluation of ML algorithms

Models	Cross-validation accuracy	Class	Precision (%)	Recall (%)	F1-score (%)	Overall accuracy (%)
Random forest	97.81%	0	100	97	98	98.44%
		1	97	100	98	
Decision tree	97.81%	0	100	100	100	100%
		1	100	100	100	
KNeighbors	96.86%	0	94	97	95	95.31%
		1	97	94	95	
Adaboost	98.12%	0	100	100	100	100%
		1	100	100	100	
SGD	97.81%	0	100	97	98	98.44%
		1	97	100	98	
Extra trees	97.50%	0	100	97	98	98.44%
		1	97	100	98	
Gaussian NB	69.32%	0	86	38	52	65.62%
		1	94	94	73	

Table 3: CNN performance metrics for 96x96 image size trained using Spyder

Trial	Class	Precision (%)	Recall (%)	F1-score (%)	Overall accuracy (%)
1	0	76	100	86	83
	1	100	63	77	
2	0	92	98	95	95
	1	98	90	94	
3	0	97	95	96	95
	1	94	96	95	

Table 4: CNN performance metrics for 192x192 image size trained using Spyder

Trial	Class	Precision (%)	Recall (%)	F1-score (%)	Overall accuracy (%)
1	0	91	100	95	95
	1	100	88	94	
2	0	95	98	97	96
	1	98	94	96	
3	0	97	98	97	97
	1	98	96	97	

Table 5: Performance evaluation of different borderlines for the combined ML and DL algorithms

Lower bound	Upper bound	Class	Precision (%)	Recall (%)	F1-score (%)	Overall accuracy (%)
10%	90%	0	100	3	6	52
		1	51	100	67	
20%	80%	0	100	29	45	64
		1	58	100	74	
30%	70%	0	100	66	80	83
		1	75	100	85	
40%	60%	0	97	92	94	94
		1	92	97	94	

List of abbreviations

API	Application programming interface
C2	Command and control
CFG	Control flow graph
CNN	Convolutional neural network
CTI	Cyber threat intelligence
DL	Deep learning
DT	Decision tree
FL	Federate learning
FNR	False negative rate
FPR	False positive rate
FPR	False positive rate
GB	Gradient boosting
GNB	Gaussian naive bayes
GRR	Google rapid response
IoC	Indicators of compromise
KNN	K-nearest neighbors
MaaS	Malware-as-a-service
MAIL	Malware analysis intermediate language
MAIT	Malware analysis and intelligence tool
ML	Machine learning
NB	Naive bayes
PA	Passive-aggressive

PE	Portable executable
PSI	Printable string information
RC4	Rivest cipher 4
RF	Random forest
RNN	Recurrent neural network
SGD	Stochastic gradient descent
SVD	Singular value decomposition
SVM	Support vector machine
XAI	Explainable AI
XGBoost	Extreme gradient boosting

Data availability

The datasets generated and analyzed during the current study are available in the GitHub repository, https://github.com/ElafFayk/RecealerMalwareDetection.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Alam S, Horspool RN, and Traore I (2013). MAIL: Malware Analysis Intermediate Language: A step towards automating and optimizing malware detection. In the Proceedings of the 6th International Conference on Security of Information and Networks, ACM, Aksaray, Turkey: 233-240. https://doi.org/10.1145/2523514.2527006

Bermejo Higuera J, Abad Aramburu C, Bermejo Higuera JR, Sicilia Urban MA, and Sicilia Montalvo JA (2020). Systematic approach to malware analysis (SAMA). Applied Sciences, 10(4): 1360. https://doi.org/10.3390/app10041360

Castiglioni I, Rundo L, Codari M et al. (2021). AI applications to medical images: From machine learning to deep learning. Physica Medica, 83: 9-24. https://doi.org/10.1016/j.ejmp.2021.02.006 **PMid:33662856**

Chen A, Chen H, Xu L, Xie H, Qiao H, Lin Q, and Cai K (2020). A deep learning CNN architecture applied in smart near-infrared analysis of water pollution for agricultural irrigation resources. Agricultural Water Management, 240: 106303. https://doi.org/10.1016/j.agwat.2020.106303

Elsayed MS, Le-Khac NA, Dev S, and Jurcut AD (2020). DDoSNet: A deep-learning model for detecting network attacks. In the IEEE 21st International Symposium on a World of Wireless, Mobile and Multimedia Networks, IEEE, Cork, Ireland: 391-396. https://doi.org/10.1109/WoWMoM49955.2020.00072

Hussain A, Asif M, Ahmad MB, Mahmood T, and Raza MA (2022). Malware detection using machine learning algorithms for Windows platform. In: Ullah A, Anwar S, Rocha Á, and Gill S (Eds.), Proceedings of International Conference on Information Technology and Applications. Lecture Notes in Networks and Systems, 350: 619–632. Springer, Singapore, Singapore. https://doi.org/10.1007/978-981-16-7618-5_53

Nurmi J, Niemelä M, and Brumley BB (2023). Malware Finances and operations: A data-driven study of the value chain for infections and compromised access. In the Proceedings of the 18th International Conference on Availability, Reliability and Security, ACM, Benevento, Italy: 1-12. https://doi.org/10.1145/3600160.3605047

Pant D and Bista R (2021). Image-based malware classification using deep convolutional neural network and transfer learning. In the Proceedings of the 3rd International Conference on Advanced Information Science and System, ACM, Sanya, China: 1-6. https://doi.org/10.1145/3503047.3503081 **PMid:33929876**

Prachi, Dabas N, and Sharma P (2023). MalAnalyser: An effective and efficient Windows malware detection method based on API call sequences. Expert Systems with Applications, 230: 120756. https://doi.org/10.1016/j.eswa.2023.120756

Rabadi D and Teo SG (2020). Advanced Windows methods on malware detection and classification. In the Proceedings of the 36th Annual Computer Security Applications Conference, ACM, Austin, USA: 54-68. https://doi.org/10.1145/3427228.3427242

Saeed MAH (2020). Malware in computer systems: Problems and solutions. International Journal on Informatics for Development, 9(1): 1-8. https://doi.org/10.14421/ijid.2020.09101

Singh J and Singh J (2020). Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. Information and Software Technology, 121: 106273. https://doi.org/10.1016/j.infsof.2020.106273

Sreekumari P (2020). Malware detection techniques based on deep learning. In the IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), IEEE, Baltimore, USA: 65-70. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00023

Sun R, Yuan X, He P, Zhu Q, Chen A, Gregio A, Oliveira D, and Li X (2022). Learning fast and slow: Propedeutica for real-time malware detection. IEEE Transactions on Neural Networks and Learning Systems, 33(6): 2518-2529. https://doi.org/10.1109/TNNLS.2021.3121248 **PMid:34723811**

Yucel C, Lockett A, Chalkias K, Mallis D, and Katos V (2021). MAIT: Malware analysis and intelligence tool. Information & Security, 50(1): 49-65. https://doi.org/10.11610/isij.5024