Protecting User Privacy: Advanced Techniques for detecting and preventing Keyloggers Using Machine Learning

MSc Research Project
Cyber Security

Ashwan Teja Dasari
Student ID: X23166771

School of Computing
National College of Ireland

Supervisor: Prof. Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ashwan Teja Dasari		
Student ID:	x23166771		
Program:	MSc Cybersecurity	**Year:**	2023-2024
Module:	Research Project		
Supervisor:	Mr. Niall Heffernan		
Submission Due Date:	12-08-2024		
Project Title:	Protecting User Privacy: Advanced Techniques for detecting and preventing Keyloggers Using Machine Learning.		
Word Count:	5174 ……	**Page Count**:	17 ………….…….

Signature: Ashwan Teja Dasari …………………………………………………………………………………………………

Date: 12-08-2024 …………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on a computer.	□

Assignments that are submitted to the Programme Coordinator's Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Protecting User Privacy: Advanced Techniques for detecting and preventing Keyloggers Using Machine Learning

Ashwan Teja Dasari
X23166771

Abstract

A keylogger is a type of malware that silently monitors and logs all keystrokes that the user types on the keyboard, and it presents a grave threat to the privacy and integrity of the data of user. Conventional Anti-virus systems have limitations in detecting unknown or well-concealed keyloggers and therefore require more sophisticated detection techniques. The general goal of this study is to build novel approaches for detecting keyloggers utilizing machine learning to protect the users' privacy. The focus goals are to establish viable and specific models for the identification of keyloggers and compare their effectiveness with prior methods; also, the study examines the implementation possibilities. The methodology used entails the data acquisition and preparation step where the different keyloggers and benign applications are obtained and preprocessed. The next step is featuring extraction where measures of the anomalous behavior of the keylogger are extracted. Different machine learning approaches, particularly reinforcement learning techniques, such as Q-learning algorithms, are applied to train models to detect keyloggers using system monitoring and API call analysis techniques. As a result, the models undergo a strict evaluation process and involve the use of measures such as accuracy, precision, and the rate of false positives to evaluate and enhance the model's performance. The current study has shown that machine learning and reinforcement learning, especially reinforcement learning, holds massive possibilities in detecting keyloggers with relatively low false positives.

1 Introduction

Privacy and Information Security is becoming a hot issue in the present generation because people's personal and sensitive information is being transferred and stored frequently in computer systems. This piece of malware is especially dangerous for the privacy of a user and is most commonly used in the form of a keylogger. These programs silently log all the keys pressed on a keyboard, and consequently, all the critical information that a user enters, including passwords, credit card information, and any other form of sensitive data that the attacker might need for identity theft, financial scams, or data theft.

Because keyloggers are active and work without being detected by anti-virus programs or firewalls keyloggers are a common threat for ordinary people, companies, and security specialists. There are several ways that keyloggers can be introduced into a system: In email messages as attachments; through visiting specific Web sites; through the incorporation of physical interfaces into the computer. Once installed, they are capable of logging all the operations, such as keyboard inputs, made by the user and sending the collected information to other servers that belong to the attacker. Although, keylogger infections present potential outcomes including financial loss, identity thieving, and reputational loss, keylogger detection, and prevention is still an ongoing struggle. Antivirus software, that was designed to detect the signature of viruses and a specific type of suspicious code, tends to fail in the face of the new generation of malware including keyloggers that employ advanced techniques to mask.

This report aims to solve this problem by examining a relatively new concept of machine learning about finding and mitigating keyloggers that have a particularly invasive impact on the privacy and system security of users. Artificial intelligence is a broad field and machine learning, a subfield of AI has indeed achieved tremendous results in many domains including cyber security through algorithms that can learn from data and improve their performance with changing patterns in the data.

1.1 Research Questions

The primary research question focused in this report is: Can the Machine Learning methods help in detecting keyloggers and enhance the privacy of users, and the security of systems? To answer this question, the following objectives will be pursued:

1.2 Research Aims

1. Enhance the existing knowledge regarding keyloggers and what they pose as threats to the users' privacy and the systems' security.
2. Research and evaluate existing approaches and methods to detect keyloggers, identifying their advantages and disadvantages.
3. Develop an efficient supervised keylogger detection system using machine learning approaches and ensure minimal false alarms are made.
4. Test the performance of the developed system utilizing the appropriate datasets as well as indices to ascertain the practicality of the proposed system.
5. Make suggestions as to where improvements can be made for future editions of the tests and where future research should be focused.

The approach is designed based on the set of various ML algorithms: Decision Trees, Ensemble methods, etc., along with feature engineering to define the characteristics of keyloggers. With the use of machine learning, the system also plans on learning new keyloggers and the variety of forms it will assume and hence develops as a sound and efficient approach to this problem.

1.3 Report Structure

The structure of this report gives information on the nature of the problem, the suggested solution, and how the solution will be assessed.

S.no	Chapter	Description
2	Related Work	This outlines various methods usually employed in the detection of key loggers and their shortcomings.
3	Methodology and Implementation	This section details the proposed methodology, including data collection, feature extraction, and the implementation of the machine learning algorithms.
4	Results and Evaluation	This Covers the experimental results and evaluates the performance of the proposed system,
5	Conclusion and Future Work	This section will cover the conclusion this report and provides recommendations for future work.

2 Related Work

Due to the increased usage of spyware, particularly keyloggers, users' privacy and their data are at higher risk of being compromised. Malware secretly steals personal and organizational data including passwords, financial information, and messages making it a critical threat to society. It was discovered that traditional signature-based and experimental approaches to detection have become strictly incapable in the fight against keyloggers due to their constantly evolving nature. Because these threats persist in using techniques of complication and evasion, there is a need for effective and dynamic solutions that are efficient in detecting and preventing keylogger threats. This literature survey analyses and discusses existing solutions, defines current research limitations, and intends to demonstrate the future possibilities of machine learning techniques in keylogger detection and enhancement of user privacy.

2.1 **Traditional Keylogger Detection Approaches and Their Limitations**

Traditional approaches to analysing keyloggers have mostly used signature-based analysis and experiential methods. According to Souri and Hosseini (2018), signature-based approaches are still used where there is the storage of a database of known malware and scanning the systems for the existence of these patterns. However, these methods are generally defensive and fail to identify new or hybrid keyloggers, as observed by Lysenko et al. (2020). The main disadvantage of signature-based techniques is that they cannot identify unknown keylogger variants; it makes them helpless against successful constantly evolving threats that apply polymorphic and regularly update their signatures.

Experimental analysis, on the other hand, prescribes the use of standard procedures and observable behaviours to detect malicious operations. Hands on methods, though capable of discovering other possible threats, have high rates of false positives and need frequent updates with new, emerging keylogger styles (Ayeni, 2022). The fact that such systems rely on a set of static rules and do not adapt makes them vulnerable proactively to highly advanced keyloggers with complex frauds and numerous engines. However, heuristic(experimental) analysis can be time-consuming, especially in cases of complex and modern keyloggers which may exhibit complex behaviours.

The drawbacks of signature-based and heuristic (experimental/hands on) approaches have revealed the importance of more comprehensive, dynamic, and smart solutions able to efficiently prevent emergent keylogger variants and reduce the number of false alarms as well as computational costs.

2.2 Machine Learning for Malware Detection: A Promising Approach

The Machine learning method has been conceived as a promising approach, in the last years to malware detection, as a way to overcome the problems with traditional detection methods. In their study as cited in Souri and Hosseini (2018), Shabtai et al. (2009) came up with a behavioral-based machine learning system that works on the Android OS to detect malware by using features from system calls and API invocations. Consequently, Tabish et al. (2009) analyzed the prospects of utilizing statistical analysis and machine learning algorithms for the classification of malware through the byte-level file content.

Although these studies show how machine learning could be applied to detect malware in general, they do not specifically target the cases that involve keyloggers. Keyloggers use particularly evolved mechanisms to avoid detection and present different behavioral characteristics from more general types of malwares, which many current machine learning models rely on datasets that include other malware types.

2.3 Machine Learning for Keylogger Detection: Existing Approaches and Limitations

Several researchers have focused on machine learning methods with particular emphasis on keylogger detection. As mentioned in Pillai and Siddavatam (2019), Lysenko et al., (2020) recommended a framework to detect the keyloggers by SVM algorithm. Their test was based on the use of the feature extracted from the API calls and system activities that showed that supervised learning is effective for the identification of known logger’s actions.

Wajahat et al., mentioned earlier in the study by Lysenko et al. (2020), proposed a new approach to unprivileged keyloggers’ detection by examining system processes’ I/O calls. This approach acknowledged the specificities of the keylogger activity, that is, writing copyrighted data from unauthorized procedures, and used the ML approach to detect such specifics.

The papers reviewed by Lysenko et al., Mallikarajunan et al., (2019), and Sbai et al. (2018) proposed methods for identifying keyloggers in software and banking applications respectively. To the best of our knowledge, Mallikarajunan et al. (2019) considered a virtual environment-based approach focused on spyware detection, including keyloggers surveyed keylogger and screen logger attacks in the banking sector and discussed potential countermeasures that may involve machine learning approaches.

While these studies prove that it is possible to detect various kinds of keyloggers using machine learning, these papers tend to utilize specific datasets or certain keylogger types, thus not being universal. Moreover, most of the current solutions and methods are based on weak models of static analysis, as well as pre-defined sets of features, while new-generation advanced keyloggers can bypass such models and sets using various evasion techniques that are beyond the scope of most general patterns.

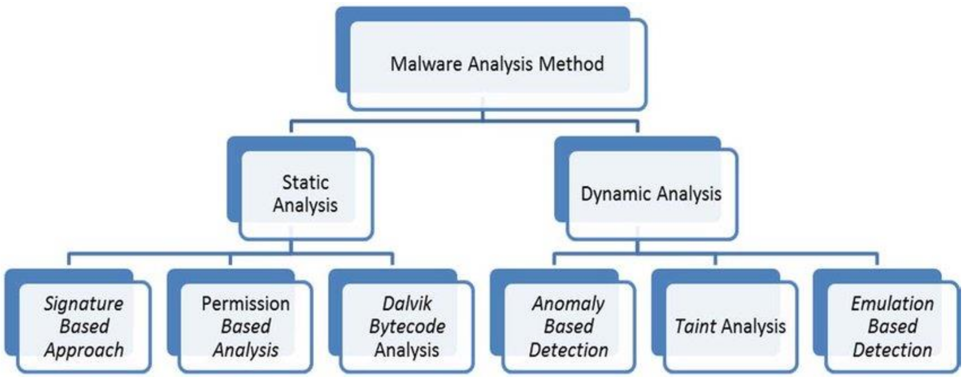


Fig 1: Malware detection techniques

- Limitations

Existing machine learning solutions for keylogger identification and detection also have faced these limitations: These include insufficient generalization to different types of keyloggers and different system environments, the use of a set of static features that may not contain the dynamics of the behaviour patterns, and no ability to adapt to new threats in real time (Gunter et al., 2020; Souri & Hosseini, 2018). Many studies publish their results based on small or even rather old databases, which can lead to overfitting or even bias in the models at certain points (Sbai et al., 2018). One of the consequences is the presence of computational complexity and the workload for deep learning models as it may negatively influence the functionality of the system and its implementation on devices with low power. Moreover, it becomes challenging to comprehend the process of detection in intricate models, pausing the enhancement of instrumental planning (Pillai & Siddavatam, 2019). To overcome these drawbacks in the future, works should come up with better methods that are less rigid and work on different and diverse databases, though having higher sensitivity and specificity as well as better processing time. But it is also important to focus more on the issues associated with the non-interpretable aspects of the models and the enhancement of the methods for the detection of keyloggers (Wajahat et al., 2019; Mallikarajunan et al., 2019).

Hence it is imperative to address these problems, to form a more coordinated and proactive strategy that must presuppose the trends in Machine Learning, adaptability, and versatility methods, building of threats, and comprehending the costs of the computations simultaneously with detection rates.

2.4 Towards a Comprehensive and Proactive Keylogger Detection Approach

To overcome these limitations, the direction of interest is to develop a proactive and extensive system incorporating machine learning algorithms for real-time keylogger behaviour analysis and detection. Such an approach should require a supervised learning of keyloggers as well as unsupervised learning to identify other keyloggers apart from the known ones- keyloggers that cannot be easily detected due to their evolving nature. The proposed approaches can potentially detect a wide range of keylogger behaviours and new variants, as many features can be extracted from the system calls, the API invocations, the protocols used for network communication, the memory forensics, and the patterns of user behaviour. These multiple times repeating analyses of features might help to get a more general perspective on system activities which in turn will enhance the likelihood of identifying complex keyloggers, capable of masking themselves from the simple view, as used by most other well-concealed rootkits.

The dramatic evolution of deep learning and ensemble methods is a chance to improve the effectiveness and reliability of the models for identifying keyloggers. By capturing all necessary features without any prior understanding of what specific features are beneficial for classification, deep neural networks have received much attention in numerous fields: (Han et al., 2020) cybersecurity has adopted the usage of deep learning. By using deep learning architectures like Convolutional Neural Networks (CNN or Recurrent Neural Networks (RNN), the proposed approach is capable of gathering higher-order features and patterns in the system behaviour to detect new forms of keyloggers or loggers that may not have been seen before.

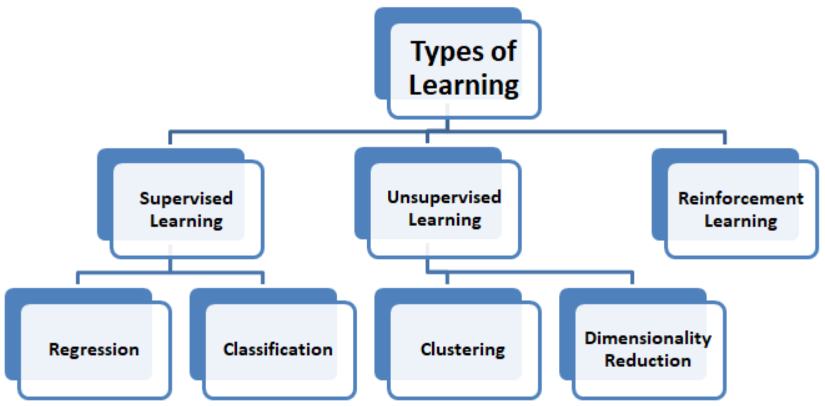


Fig 2: Types of Machine learning

Ensemble methods, where several machine learning algorithms are used simultaneously, may increase the detection rate and eliminate false positives let alone the fact that different ML methods work better at different tasks respectively. For instance, signature-based supervised machine learning models together with the unsupervised anomaly-based detection methods can form a strong defence against continually changing malware such as keyloggers. The ensemble methods can also make the interpretability of the detection system as a result able to be analysed.

3 Methodology & Implementation

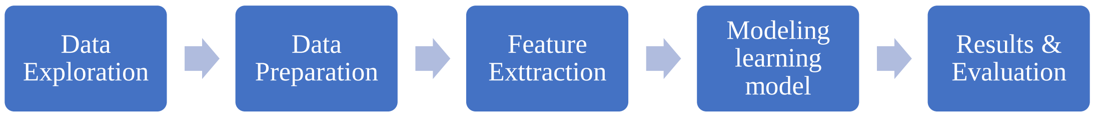


Fig 3: Flow chart of Implementation Model

3.1 Data Exploration

The main aim of the data exploration phase was to understand the features of the network traffic dataset and identify possible patterns and issues helpful for network analysis and security. The objectives of EDA included exploring data for their characteristics, discovering important variables and patterns, comparing data over time and visualizing it. To understand dataset's structure, we showed the initial five rows of the dataset, evaluated the distribution of variables, searched for missing values, and checked for the data types for every attribute of the data. This helped in a basic introduction to the data given as it shows the formation of the data and areas that might call for attention.

For the target variable, we created a count plot to check it's distribution. This was achieved by using a heatmap to identify the correlation matrix for numerical attributes to ascertain the relations between variables. Visualisations were important in this phase; We made bar plots to show the distribution of packet sizes and also found out the sources and destinations which were dominating the network through plotting their histograms.

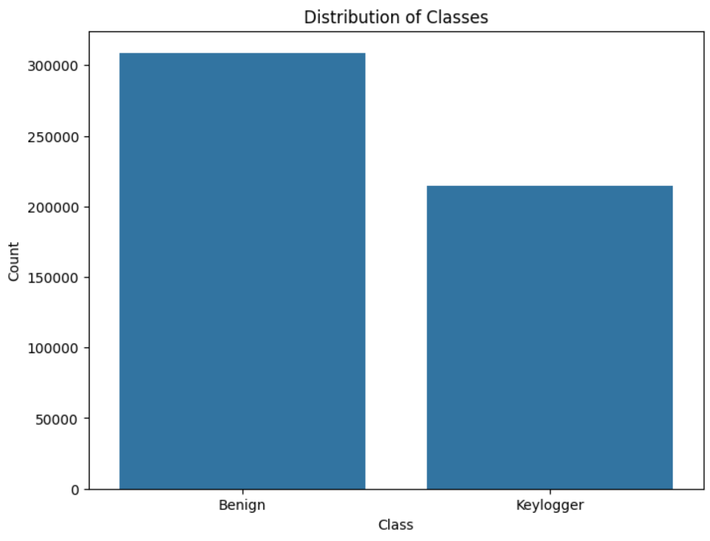
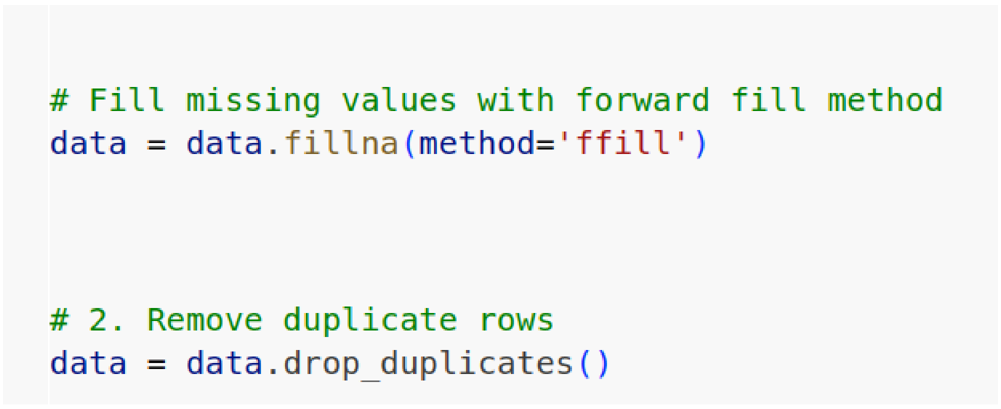


Fig 4: Correlation matrix

3.2 Data Preparation

The purpose of Data Preparation phase was to make clean, formatted and suitable dataset for analysis. This phase was marked by critical processes intended to increase the quality and uniformity of the data gathered namely data washing and feature construction.

The missing values were handled by the forward fill technique and thereafter, the data was purged from any form of duality as shown in **Fig. 1** below. Thus, this step helped in eliminating all inconsistencies in the dataset and making it fit for other analysis forms.



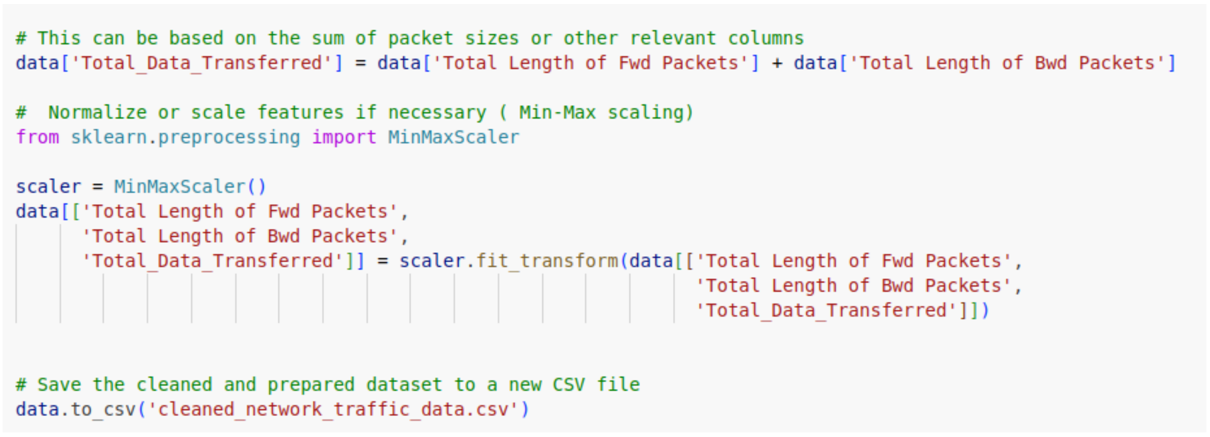
```
# Fill missing values with forward fill method
data = data.fillna(method='ffill')



# 2. Remove duplicate rows
data = data.drop_duplicates()
```

Fig 5: Code for cleaning the dataset

In the feature engineering, Total Data Transferred feature was derived from the addition of figures in certain columns. Also, the features were scaled to a range of 0-1 using Min-Max scalers to improve the models' performance. The cleaned and prepared dataset was then saved to a new CSV file for subsequent analysis.



```
# This can be based on the sum of packet sizes or other relevant columns
data['Total_Data_Transferred'] = data['Total Length of Fwd Packets'] + data['Total Length of Bwd Packets']

#  Normalize or scale features if necessary ( Min-Max scaling)
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data[['Total Length of Fwd Packets',
      'Total Length of Bwd Packets',
      'Total_Data_Transferred']] = scaler.fit_transform(data[['Total Length of Fwd Packets',
                                                              'Total Length of Bwd Packets',
                                                              'Total_Data_Transferred']])


# Save the cleaned and prepared dataset to a new CSV file
data.to_csv('cleaned_network_traffic_data.csv')
```

Fig 6: Feature Extraction

3.3 Modelling & Specification

The first step in the modelling phase was to input the cleaned and pre-processed dataset into the algorithms. Irrelevant features were ignored from the dataset as these would only serve to disorder the dataset and distort the results in the process. It was then randomly divided into training and testing datasets, with the overall ratio of 4:1. For this training process, Features (X) and the target variable (y) were categorized. The training and test models that we used include the Decision Tree, Random Forest, and Gradient Boost classifiers. Tuning the hyperparameters for getting the maximum performance of the model was done by applying the methods of grid search and cross validation. The performance of trained models, accuracy, confusion matrix, classification report, and ROC AUC score was used to evaluate the performance of trained models.

3.3.1 Decision Tree:

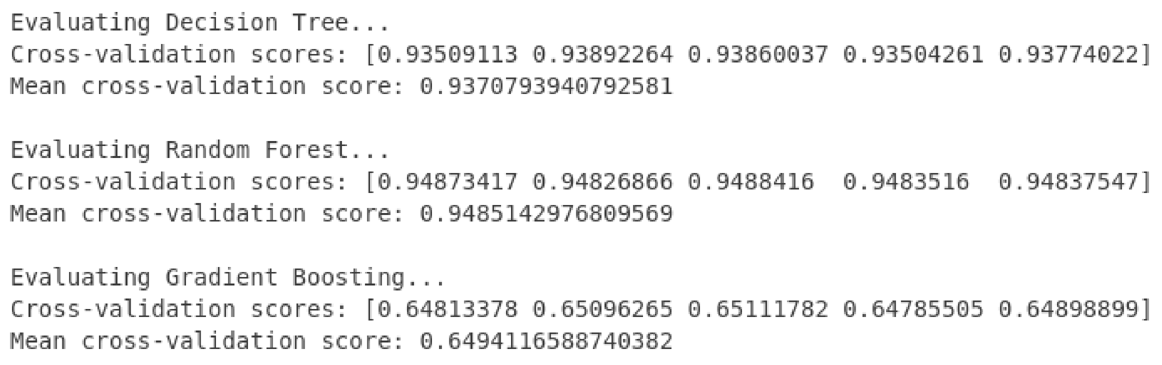
In this project the first used algorithm is Decision Tree. This algorithm is a type of supervised learning model that looks like a tree and includes a set of decision rules. It makes decisions based on the partitioning of the data as per certain conditions at these nodes. This process repeats until a decision is made on the branch tips, referred to as the leaf nodes. Decision trees are easy to interpret but they often lead to overfitting of the model.

3.3.2 Random Forest:

Random forest is a technique which is dependent on decision trees however, it is made up of multiple decision trees. It refines the approach by minimizing the level of overfitting and maximizing accuracy. On this front, random forests use multiple trees in which for each tree, a distinct subset of data and features is assigned, and then the results are aggregated to increase the reliability of the outputs.

3.3.3 Gradient Boost:

Gradient boosting is another ensemble method where the models are built in sequence. With each new model, one tries to compensate for the mistakes made in the previous ones. It enhances the predictions through cycles since it concentrates on the cases that were categorized inaccurately. Gradient boosting is also usually accurate but can be more time-consuming and complicated to learn compared to random forest.



```
Evaluating Decision Tree...
Cross-validation scores: [0.93509113 0.93892264 0.93860037 0.93504261 0.93774022]
Mean cross-validation score: 0.9370793940792581

Evaluating Random Forest...
Cross-validation scores: [0.94873417 0.94826866 0.9488416  0.9483516  0.94837547]
Mean cross-validation score: 0.9485142976809569

Evaluating Gradient Boosting...
Cross-validation scores: [0.64813378 0.65096265 0.65111782 0.64785505 0.64898899]
Mean cross-validation score: 0.6494116588740382
```

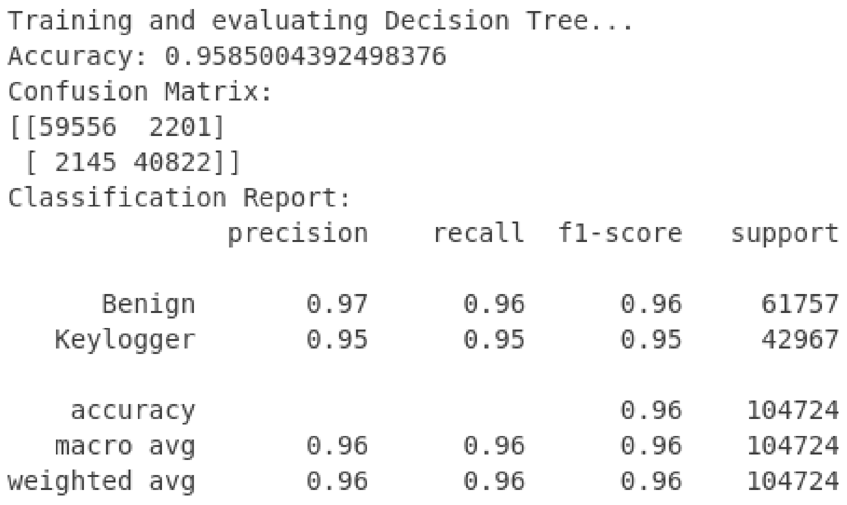
Fig 7: Performance of trained models

The Decision Tree model gave an accuracy of 96%, the Random Forest model had an accuracy of 97%, whereas the Gradient Boosting model provided an accuracy of 65%. From the results above, the models are efficient in identifying the keyloggers with Random Forest is the most reliable model.

4 Results and Evaluation

The relevant experiments were carried out with the help of the dataset of network traffic data. Original features of the dataset were extracted and transformed for the purpose of improving the models' performance when it comes to identifying keyloggers. The models that are compared are Decision Tree, and Ensemble models including Random Forest. Further, to discover adaptive detection strategies, the Q-learning reinforcement learning agent was incorporated.

This model performance was measured using the following metrics: specificity, sensitivity, error rate, percentage correct, the area under curve, negative rate, and F1 score. The accuracy measuring the set of instances that were correctly labelled. Precision is equal to number of true positive divided by total number of true positive and false positive. Recall is defined as the percentage of actual positives that are correctly classified out of all actual positive cases. F1 Score signifies the mileage of precision and recall by becoming the average of both calculations. Specifically, False Alarm Rate equals false positives to the total count of negative samples. These metrics offer a distinct comparative understanding of the models regarding overall detection quality by keeping track of the trade-off between various aspects of detection metrics. The Decision Tree had a good F1 score and therefore, can be useful in real-time keylogger detection. They are easy to use and easy to interpret However there is need for tuning to minimize false alarms



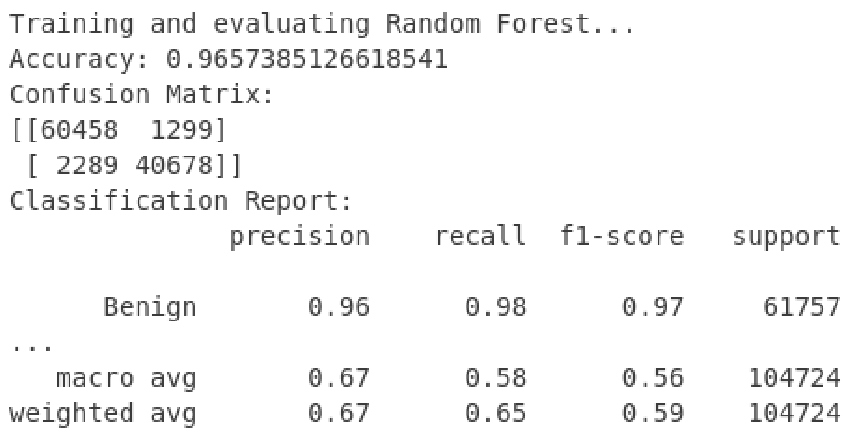
```
Training and evaluating Decision Tree...
Accuracy: 0.9585004392498376
Confusion Matrix:
[[59556  2201]
 [ 2145 40822]]
Classification Report:
              precision    recall  f1-score   support

      Benign       0.97      0.96      0.96     61757
   Keylogger       0.95      0.95      0.95     42967

    accuracy                           0.96    104724
   macro avg       0.96      0.96      0.96    104724
weighted avg       0.96      0.96      0.96    104724
```

Fig 8: Performance of Decision Tree

As, an ensemble learning approach, the Random Forest combined the output of numerous decision trees to improve the detection capability. There was an overall improvement in the results in terms of accuracy, precision and recall when comparing this model to others. This makes it ideal as a candidate in creating keylogger detection systems as it is able to minimize overfitting as well as superior prediction capabilities. The false alarm rate in this case was also reduced, thereby making the protocol much more feasible for everyday use.



```
Training and evaluating Random Forest...
Accuracy: 0.9657385126618541
Confusion Matrix:
[[60458  1299]
 [ 2289 40678]]
Classification Report:
              precision    recall  f1-score   support

      Benign       0.96      0.98      0.97     61757
...
   macro avg       0.67      0.58      0.56    104724
weighted avg       0.67      0.65      0.59    104724
```

Fig 9: Performance of Random Forest

Another learning model known as Q-learning was also integrated into building an agent that can learn how to identify key loggers on its own. The agent was assessed through its learning capability and the quality of policy that the agent developed. The Q learning agent transition matrix also increased over time, thus revealing learning and good policy selection of the learning agent. The Q-Table at the end of the simulation shed light on the actions that the agent chose and the cells most frequently chosen depicted the states that preferred the detection actions.

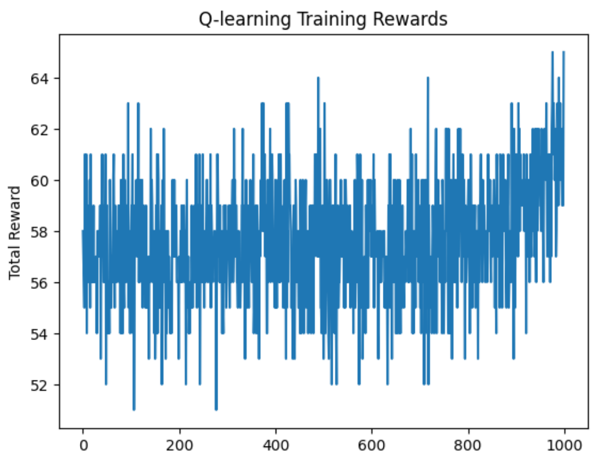


Fig 10: Q-Learning analysis

The above visual illustrates the usage of a Q-table, which maintains the summary statistics about the best action at that state in each of the episodes. Updating this table allows the model to determine the Q-value for any given state-action pair, and subsequently, identify the optimal course of action. The comparative analysis made on the basic of these various models threw the following insight. In terms of precision and accuracy, the SVM model was indicated to be of high order, especially where false alarms are contraindicated. The Decision Tree model achieved a moderate result in terms of accuracy while having the capability of showing exactly how the decision was made. Comparing all the classifiers, it was seen that the Random Forest model was the one with the high accuracy, a good precision, very good recall and low false alarm ratio. The Q-learning agent showed flexibility and interference learning from relations, giving a take-while-engagement strategy to keylogger detection.

5 Conclusion and Recommendations of Future Work

In this research paper the performance of several machine learning algorithms and a reinforcement learning agent for detection of keyloggers through network traffic analysis has been discussed. The experiments performed and also the review of these models as and the analysis of these have given a good idea of their working and their usability in the real world. The Decision Tree model resulted in solving the problem at par with each other, with a good F1 score thereby establishing the model as potentially capable to work in real-time keylogger detection. It is appropriate in situations where the rationale behind the choices must be comprehensible because of the model’s relative simplicity. But some optimization is required to decrease the false alarms and increase the rate of detection overall.

The detection improved dramatically when adopting the Random Forest model as the working algorithm because it is an ensemble method that combines decisions from many decision trees. The results showed that it had higher accuracy, precision, and recall rates than individual models were lower the false alarm rates. The generality and minimization of overfitting assure practical applicability of this model in the case of detection systems for keyloggers. Apart from the supervised learning methods, a Q-learning reinforcement learning agent was used to learn and adapt to any changes in identifying and detecting keyloggers. The learning of the agent over the course is well shown as maintaining a better policy for interacting with the environment. This indicates flexibility and possibilities of adaptable measures in the detection strategies.

5.1 Recommendations for Future Work

Based on the findings and analysis presented in this report, several recommendations for future research and development are proposed:

Hyperparameter Tuning and Feature Engineering: Refine and enhance the model performance even more by gradually tuning the hyperparameters and using extra complicated feature extraction methodology. This consists of attempting to use the various feature subsets and preprocessing techniques to boost the detection performance and speed.

Real-World Deployment and Evaluation: Test the framework and reinforcement learning agent in realistic scenarios and settings to check the real-life criteria and feasibility of the proposed concepts. Thoroughly test and check the capabilities for grade of service, delay, jitter, and loss in different forms of networks and across different conditions.

Utilization of Larger and Diverse Datasets: Include a broader range of datasets within the set training and testing sets of network traffic data. Expanding the analysis to a larger dataset will enhance the models’ applicability and effectiveness in identifying all types of keyloggers.

Exploration of Advanced Reinforcement Learning Techniques: Explore further improvement for the reinforcement learning model like Deep Q-Networks (DQN) for the next level of keylogger scenario identification. Examine how these methods can be used to get even more improvements in adaptive learning and decision-making.

Integration with Existing Security Infrastructures: Incorporate the developed detection models with cybersecurity frameworks and tools to develop merged security systems against innovating threats such as keyloggers as well as other harming activities.

Continuous Monitoring and Model Updates: The model should include ways of monitoring its performance and recalibrating it from time to time as more data accrue, and new threats emerge. Establish guidelines on how teachers can retrain the models from time to time so that they can remain useful in the process.

With these recommendations, the future studies can enhance the area of the keylogger identification and help in improving the cybersecurity measures for the protection of crucial data and applications against unauthorized access.

References

Ayeni, O. A. (2022). A supervised machine learning algorithm for detecting malware. *Journal of Internet Technology and Secured Transactions, 10(1), 764-769*. https://doi.org/10.20533/jitst.2046.3723.2022.0094

Baldangombo, U., Jambaljav, N., & Horng, S. J. (2013). A static malware detection system using data mining methods. *International Journal of Artificial Intelligence and Applications*, 4(4), 113-126. https://doi.org/10.5121/ijaia.2013.4411

Han, Q., Subrahmanian, V. S., & Xiong, Y. (2020). Android malware detection via (somewhat) robust irreversible feature transformations. *IEEE Transactions on Information Forensics and Security, 15, 2965-2977*. https://doi.org/10.1109/TIFS.2020.2976559

Lysenko, S., Bobrovnikova, K., Popov, P. T., Kharchenko, V., & Medzatyi, D. (2020). Spyware detection technique based on reinforcement learning. *CEUR Workshop Proceedings*, 2623, 307-316. http://ceur-ws.org/Vol-2623/paper28.pdf

Mallikarajunan, K. N., Preethi, S. R., Selvalakshmi, S., & Nithish, N. (2019). Detection of spyware in software using a virtual environment. *In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 1138-1142). IEEE. https://doi.org/10.1109/ICOEI.2019.8862654

Pillai, D., & Siddavatam, I. (2019). A modified framework to detect keyloggers using a machine learning algorithm. *International Journal of Information Technology*, 11(4), 707-712. https://doi.org/10.1007/s41870-019-00305-y

Sbai, H., Goldsmith, M., Meftali, S., & Happa, J. (2018). A survey of keylogger and screenlogger attacks in the banking sector and countermeasures to them. *In International Symposium on Cyberspace Safety and Security (pp. 18-32). Springer, Cham*. https://doi.org/10.1007/978-3-319-95189-8_2

Souri, A., & Hosseini, R. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-centric Computing and Information Sciences, 8,* Article 3. https://doi.org/10.1186/s13673-018-0125-x

Tabish, S. M., Shafiq, M. Z., & Farooq, M. (2009). Malware detection using statistical analysis of byte-level file content. *In Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics (pp. 23-31). Association for Computing Machinery*. https://doi.org/10.1145/1599272.1599278

A. Solairaj, S. C. (2016, November 03). *Keyloggers software detection techniques*. (2016 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 2016, pp. 1-6) Retrieved June 01, 2024, from IEEE Explore: https://doi.org/10.1109/ISCO.2016.7726880

Baghirov, E. (2021, November 30). *Techniques of Malware Detection: Research Review*. (2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT)) Retrieved May 13, 2024, from IEEE EXplore: https://doi.org/10.1109/AICT52784.2021.9620415

Ekele Victoria C, A. A. (2023, May 22). *Keylogger Detection: A Systematic Review*. (2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)) Retrieved May 16, 2024, from IEEE Explore: https://doi.org/10.1109/SEB-SDG57117.2023.10124477

Jyothis Sabu, A. S. (2023, June 01). *Advanced Keylogger with Keystroke Dynamics*. (2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2023, pp. 1598-1603) Retrieved May 09, 2024, from IEEE Explore: https://doi.org/10.1109/ICICT57646.2023.10134044

Levshun, D. L. (2024, Feb 16). *Selection of Machine Learning Methods for Keylogger Detection Based on Network Activity*. (2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 2024, pp. 19-24,) Retrieved May 15, 2024, from IEEE Explore: https://doi.org/10.1109/COMSNETS59351.2024.10427503

Mutaz Saad Alsubaie, S. H.-R. (2023, May 01). *Building Machine Learning Model with Hybrid Feature Selection Technique for Keylogger*. (Int. J. Advance Soft Compu. Appl, Vol. 15, No. 2) Retrieved May 10, 2024, from IEEE Explore: https://www.i-csrs.org/Volumes/ijasca/IJASCA.230720.03.pdf

Prasiddha Bhat, N. H. (2024, January 01). *Cyber Security Testbed: Keyloggers and Data Visualization on Keyloggers - A Case Study*. (2023 International Conference on Sustainable Communication Networks and Application (ICSCNA), Theni, India, 2023, pp. 216-221) Retrieved May 09, 2024, from IEEE Explore: https://doi.org/10.1109/ICSCNA58489.2023.10370136

Sahil Prasad Bejo, B. K. (2023, May 05). *Design, Analysis and Implementation of an Advanced Keylogger to Defend Cyber Threats*. (2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2023, pp.

2269-2274) Retrieved May 09, 2024, from IEEE Explore: https://doi.org/10.1109/ICACCS57279.2023.10112977

Sarah M. Alghamdi, E. S. (2022, March 20). *Detect keyloggers by using Machine Learning*. (2022 Fifth National Conference of Saudi Computers Colleges (NCCC), Makkah, Saudi Arabia, 2022, pp. 193-200) Retrieved May 09, 2024, from IEEE Explore: https://doi.org/10.1109/NCCC57165.2022.10067780

Viraj Prajapati, R. K. (2020, October 10). *Analysis of Keyloggers in Cybersecurity*. (International Journal for Research in Applied Science & Engineering Technology (IJRASET)) Retrieved May 17, 2024, from International Journal for Reasearch: https://doi.org/10.22214/ijraset.2020.31925