International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(5s) 2026 pp. 1011-1025
https://doi.org/10.70917/ijcisim-2026-2848
Article

Enhancing Cybersecurity Through Honeypot Integration with Deep Learning: A Dual-Phase Approach to Threat Detection

Sweety Jachak[1], Shital Aher[2], Gokul Mahajan[3], Megha Rode[4], Gulrukh Nazneen[5]

[1]Dmepartent of Computer Engineering Department, Guru Gobind Singh College of Engineering and Research Centre, Nashik, Maharashtra, India
[1]Email ID: sweety.3288@gmail.com@gmail.com, [1]ORCID ID: 0009-0001-2719-0153
[2]Information Technology Department, Sir Visvesvaraya Institute of Technology, Nashik, Maharashtra, India
[2]Email ID: saher40@gmail.com, [2]ORCID ID: 0009-0001-3090-5690
[3]Department of Mechanical engineering, Shatabdi Institute of Engineering and Research, Nashik, Maharashtra, India
[3]Email ID: gokul.3688@gmail.com , [3]ORCID ID: 0000-0002-6282-8319
[4]Department of Computer Science Engineering, Ramdeobaba University, Nagpur, Maharashtra, India
[4]Email ID: megha.rode14@gmail.com [4]ORCID ID: 009-0003-5597-9957
[5]Department of Computer Science and Engineering, Ramdeobaba University, Nagpur, Maharashtra, India
[5]Email ID: gulrukh1002@gmail.com, [5]ORCID ID: 0009-0005-9870-9820

Abstract: This research aims at assessing the effectiveness of a two-stage approach to enhance cybersecurity through the use of honeypot systems and machine learning algorithms. The first stage is to use honeypots with the Pentbox tool and a Flask-based application to detect and report malicious activity like SYN-flood attacks. The Pentbox tool was used as a quick-deployable tool to record the detailed attack information, in tandem with Flask-based honeypot that allowed the extensive analysis of the HTTP requests and responses. In the second phase, the collected data was pre-processed and fed into deep learning - machine learning hybrid algorithms like CNNs hybridized with Random Forest, Naive Bayes, K-Nearest Neighbour (KNN), and Logistic Regression to identify and classify the cyber risks. The CNN with Random Forest model gave the best results as compared to the other models with an accuracy of 0.96, precision of 0.98, recall of 0.82, and F1-score of 0.88. This higher performance is as a result of its ensemble-ness which helps in minimizing over-fitting and at the same time correctly dealing with both nominal and ordinal data. The honeypots were able to collect important information of DoS-attack and the ML models, particularly the Random Forest model, was effective in the identification of threats and their categorisation. This integrated strategy not only improves the threat detection, but it also has significant implications for the development of real-time cybersecurity solutions, providing organisations with powerful means to counter current and emerging cyber threats.

Keywords: Cybersecurity, Honeypot System, SYN-food Attacks, Pentbox Tool, CNN, Random Forest, KNN.

1. Introduction

Need of a strong infrastructure to defend Security has emerged as a key issue in the recent past especially for computer-based systems because of increase in the number as well as the sophistication of threats. This new threat has brought awareness of the various weaknesses of the digital systems and the need to develop measures that will enable the system to be protected from any loss of sensitive data [1]. The requirements of the present world for a robust and efficient security solution are rather apparent for several reasons. First of all, the anonymity of the data is one of the main considerations. A large amount of personal and sensitive information is disclosed online, such as financial statements, bank accounts, business secrets, and other confidential information, therefore a good cybersecurity system

defends this information from attacks, violations, and thefts [2]. Secondly, the growth of the number and the further complication of the threats indicate that the problem of cyber security is urgent. Viruses, Trojans, and other type of threats are known to target personal 333computers, smart phones, and even industrial systems. Strong cyber security framework serves as protection against these threats as such a system is able to detect, reduce and prevent potential security threats [3]. Besides, the issue of digital trust is also relevant in the modern world that is based on the exchange of information. For the consumer, it is important to continue to have confidence in the security of the Digital communication and transaction. Strong cybersecurity infrastructure sustains this trust which is the foundation for safe and continued business.

A security measure, 'honeypot' involves setting up fake networks or systems so that the attackers can be drawn towards them. In this way, these systems imitate real targets that the attackers will have to engage in a communication with, be it servers or IoT devices. With the help of honeypots, which serve as traps, businesses can observe and examine the methods, strategies, and actions taken by attackers. Organizations can learn a lot about potential vulnerabilities, attack methods, and new threats by observing their behaviour in the honeypot environment [4], [5]. Figure 1 below depicts the honeypot method to tackle the vulnerabilities.

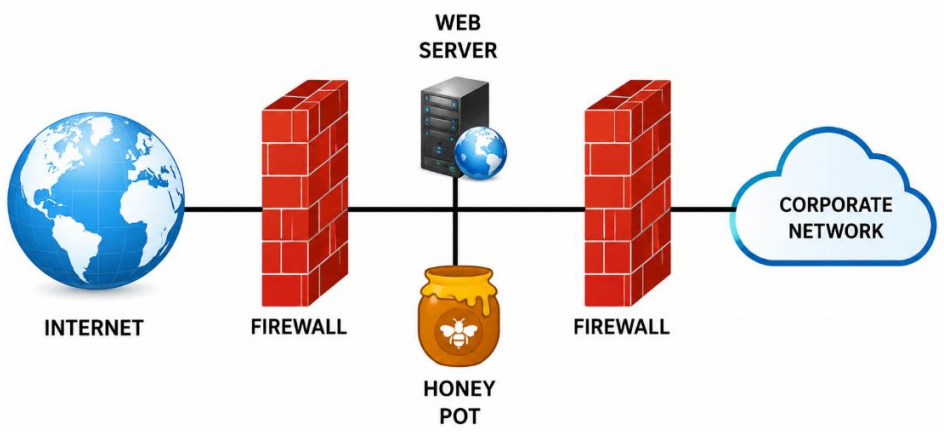

Fig 1. Honeypots Basic Framework.

A two-layer honeypot structure for IoT security against cyberattacks is shown in fig. 2 below.

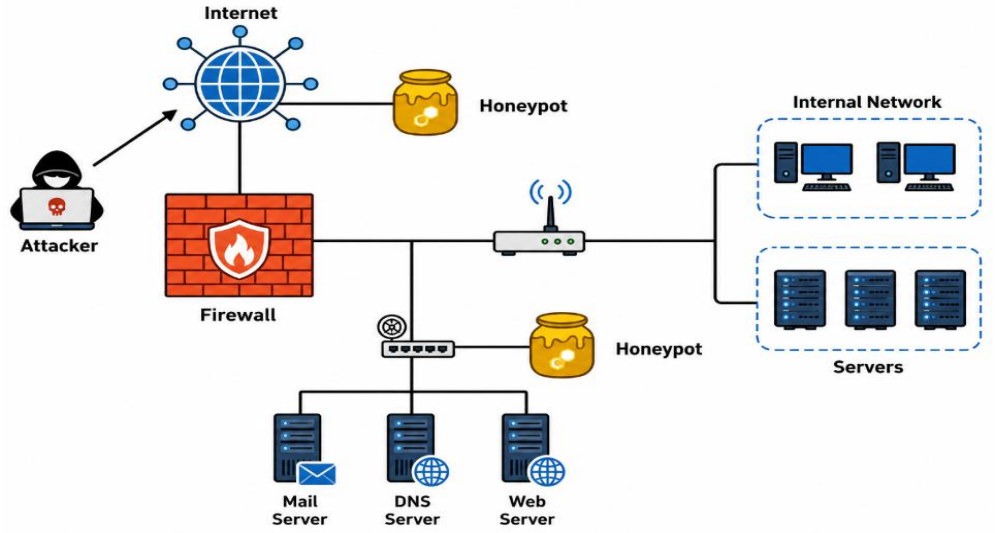

Fig 2. Two-layer honeypot framework.

This research focuses on the implementation of honeypots and evaluation of a machine learning-based honeypot system against zero-day attacks. The scope includes the following aspects:

- Honeypot Implementation: The research will involve designing and developing a honeypot setup, primarily through Python3 scripting and then pentest tool. There are many ways to implement honeypots, but the discussion is limited to above 2 methods only.
- Experimental Evaluation: The research will conduct experiments and simulations to evaluate the performance and efficacy of the honeypot system in real-world scenarios. This can be done through many tools and software, but this work is limited to pentest and python3 and Wireshark.
- Machine Learning implementation: The research will utilize multiple machine learning algorithms like Random Forest, Naive Bayes, Logistic Regression and KNN. Even though there are many algorithms to do so, and then there are also many hybrids possible, but this research is limited to the mentioned 4 models.

With the overall objective of expanding honeypot technology and strengthening network security against new cyber threats, this study intends to thoroughly analyze and respond to the following main research objectives:

Objective 1. Honeypots implementation through python and Wireshark

The first aim is to utilize Python to actually build honeypots. Further, a cyber-attack will be conducted on the honeypot in a controlled environment, and the malicious activity will be collected and logged. A popular network protocol analyzer called Wireshark will be used to make it easier to simulate and examine the attack.

Objective 2. Utilizing machine learning for real-time detection

After a successful attack and logging of the network parameters during the attack, the final aim is to look at how machine learning algorithms may be incorporated into the honeypot architecture for real-time, proactive cyberattack detection. Machine learning models will be trained for the study utilizing previous honeypot-detected attack datasets. To find the best algorithm for real-time cyberattack detection, a detailed analysis of these models will be done using performance measures including accuracy, recall, precision, and F1-score. The goal of this objective is to offer a data-driven strategy to improve honeypot capabilities and support pro-active cybersecurity tactics.

2. Literature Review

2.1 Review of the existing researches

It was discovered through the assessment of existing literature that some scientists [6] used a methodology centred on honeypots to manage hazards within networks. Their strategy comprised employing honeypots to lure cyber-attackers into a specified "sandbox" where their actions could be monitored. Their system was not only dynamically setup as an extremely interactive honeypot, but it was also distinguished by a modular architecture that separated the honeypot design from the principal data processing node. As a result, their honeypot gathered enough data to construct a strong defence against invaders. In another study [7], honeypot system was established in order to enhance Internet network security with the aim of identifying intrusion or intrusions attempts/attacks. This involved establishing the Modern Honey Network (MHN) to ease deployment and management but at the same time enhancing honeypot security. Malevolent users were caught with the help of the concept of social honeypots [8]. They have plans of employing social honeypots for the purposes of collecting information from the fake accounts as well as evaluate the characteristics of the fake accounts. They also employed honeypots in developing classifiers that could assist in the filtration of existing profiles and the monitoring of the new ones. Another worthwhile application of honeypots was mentioned by the [9] who constructed three game-theoretic models that described the interactions between honeypots and other members of a network security system. These models were a simple honeypot choice game, an added honeypot activity option for the attackers, and one where the attacker strategies were depicted by attack graphs. [10] proved that honeypot programs compensated for the shortcomings of the current monitoring methods and enhanced the outcomes of protection programs. A research group [11] also tried to determine or define low-interaction and high-interaction honeypot and their usefulness. To address these constraints, they created a hybrid honeypot architecture that combined the characteristics of both low and high-interaction honeypots. [12] pioneered the notion of "cell-pots" for threat detection and security within cellular networks by introducing a design for next-generation cellular network honeypots. To further this effort, [13] devised a system for collecting network traffic via a honeypot setup that used pattern-matching techniques and protocol conformance checks at different levels of the protocol hierarchy. [14] addressed the problem of thwarting attacks in honeypot-enabled networks by developing a game-theoretic model centred on deception and involving an attacker and a defence. [15] and [16] offered similar review studies, focusing on honeypot use in education, hybrid settings with intrusion detection systems (IDS), and using

signature techniques to analyze honeypot-generated traffic. This collective examination highlights the growing importance of honeypots as critical tools in protecting corporate entities from cyber threats.

2.2 Research Gap

Given the growing significance of using honeypots and leveraging machine learning to networks from zero-day attacks, there is a noticeable gap in the existing scholarly works. While studies on honeypots and their effectiveness in detecting and preventing breaches across multiple scenarios have been carried out, it is evident that is a lack of researches around utilizing honeypot data generated for attack predictions. This is the gap that this research aims to fulfil through combining Honeypots to Machine Learning as a robust solution.

3. Methodology

This section describes the methodology for the implementation of Honeypot attacks. The further chapter is divided into 2 parts - The part 1 explains the methodology of the Honeypot implementation, attack on Honeypot logging the data. The part 2 explains the methodology of using machine learning algorithms to detect the attacks in the data logs.

A. The Rationale behind two phases of implementation.

Honeypot is a very crucial part of network security infrastructure. The main idea behind the implementation of Phase 1, is to demonstrate the utilization of honeypot to capture the unauthorized access. But it is important to ask, if just capturing the data is enough. In real-world applications, only capturing the unauthorized access isn't enough. It is also important that the infrastructure should be capable enough to pre-emptively detect and inform the network experts of the unauthorized access. This is only possible, when there's an algorithm that is "smart" and constantly analyzing the honeypot logs for any cyber-attack. This is where machine learning algorithms find their role. Hence, the phase 2 of the implementation deals with utilizing a readily available dataset, and deploying ML algorithms on it, to detect the cyberattacks. This is how, the phase 2 becomes a logical extension of the phase 1 of the implementation.

B: Phase 1: Honeypot Implementation and Attack Using Wireshark

The honeypot has been implemented using the Pentbox tool then furthering it by python. Further, DoS attack simulation is done using hping3 and captured the traffic using Wireshark.

- Using the Pentbox Tool: The approach to implement a honeypot involved utilizing the Pentbox tool, a penetration testing toolkit. The toolkit contains various tools and scripts for security testing, network scanning, and vulnerability assessment. The honeypot component of this tool has been used for the purpose [16].
- Implementing with Flask: This involved building a custom honeypot using the Flask web framework. Flask enables the creation of a simulated web application that had the logging functionality too with the help of logging library in python.
- Simulated Denial of Service (DoS) Attack: Further, simulation of a DoS attack by SYN-flooding using the hpinng3 tool is done in kali Linux. The comprehensive logs were successfully captured by the already setup honeypot.
- Capturing Traffic Using Wireshark: During the simulated DoS attack, network traffic was captured using Wireshark, a network protocol analyzer. This allows us to analyze the attack traffic, understand its characteristics, and identify potential patterns.

C: Use of machine learning for attack detection

The rationale behind this section 3.3 is that now once that the logs are generated, is there a way that we can leverage the past logs, and leverage machine learning algorithms to detect malicious/attack content in a log. If one is able to do so, this algorithm will find is application as a live-running algorithm, and we would have covered an end-to-end demonstration of honeypot, right from building the code, and simulating an attack in part 1, to analyzing the logs using ML to find real-life applications. The below methodology explains the utilization of multiple machine learning algorithms and relevant flows for attack detection. Fig 3 depicts the ML implementation flow at a broader level.

Step 1: Selection of an appropriate dataset:

Step 2: Analyzing the Dataset Using Different Visualization Mechanisms:

Step 3: Splitting the dataset into '*training_dataset'* and 'test_dataset'

Step 4: Training the ML algorithms

Step 5: Prediction of cyber-attacks on the '*test_dataset'*

Fig 3. ML Implementation Flow

Dataset Description: The issue of malicious websites is a significant matter since it's challenging to individually assess and catalogue each URL on a blacklist. The compilation encompasses a total of 63,191 URLs, sourced from the below link. https://github.com/urcuqui/WhiteHat/tree/master/Research/W eb%20security/datasets Within this dataset, there are annotations denoting URLs as either malicious (M***) or benign (B****). The acquisition of this dataset involved the utilization of diverse credible origins for both benign and malicious URLs. This process occurred within a minimally interactive client honeypot designed to segregate network traffic. To augment the dataset's contents, the creators employed supplementary tools to extract further details, including server countries via the Who is database. The sequence followed by the dataset's authors to finalize the compilation is outlined in fig 4.

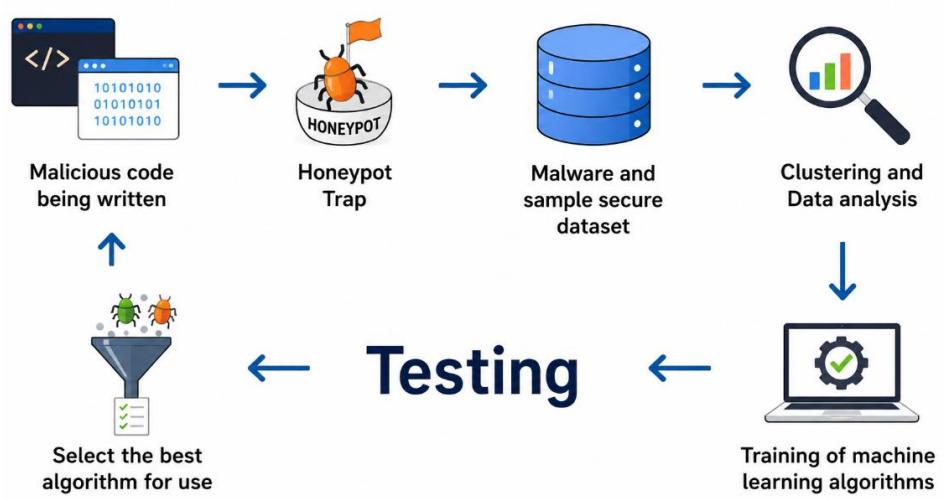

Fig 4: Flow of dataset creation used by creators

Algorithms Utilized: Four algorithms, namely Random Forest, Naïve Bayes, KNN and Logistical Regression were utilized and trained, and finally tested on multiple parameters.

4. Implementation and Results

This section is also divided into 2 parts - The first part in section 4.1 details the results of the honeypot implementation and the second part discusses the results of the machine learning algorithm implementation.

Honeypot Implementation & Results

Step 1: Setup on the Pentest Tool

In order to use this tool, first of all we have to clone it from Github using the url: URL: "*https://github.com/technicaldada/pentbox*". Once extracted, now we will move into the *pentbox* directory and run pentbox.rb file with sudo perms, using the command sudo./pentbox.rb. Fig 5 shows the run window of pentbox for honeypot.

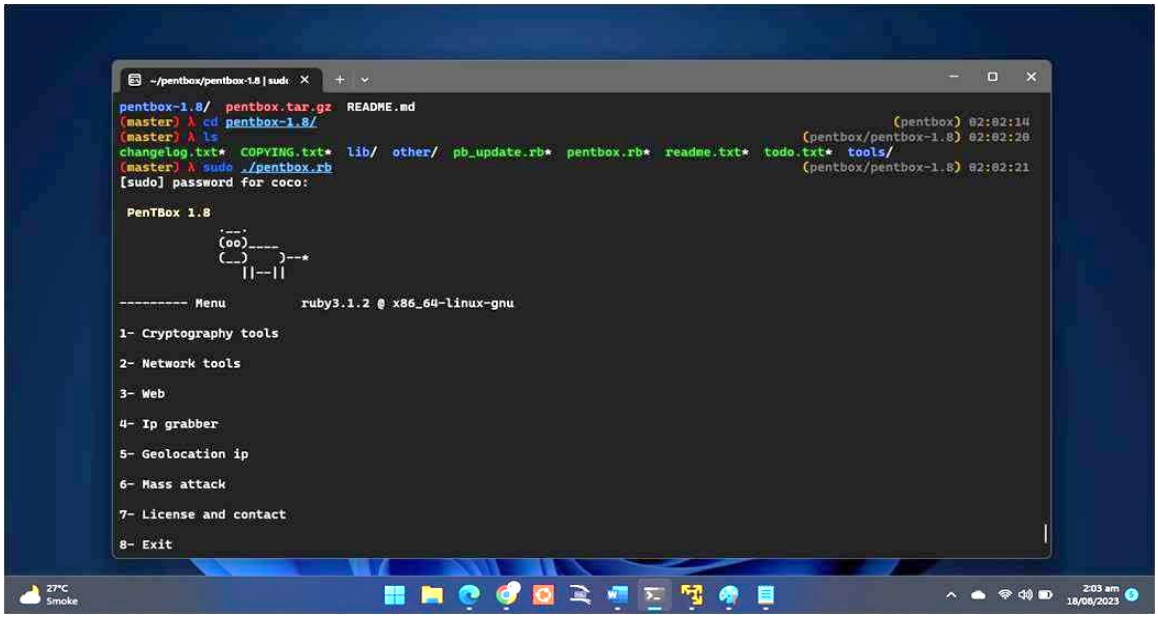

Fig 5. Pentbox Running Window.

Step 2: Setting up the honeypot

The honeypot is located in the Network tools section, so option 2 was chosen, and then option 3 is chosen to access honeypot. It gives two options: automatic and manual. The chosen one is automatic first by option 1 which will setup a honeypot on our port 80. Fig 6 shows the setup process.

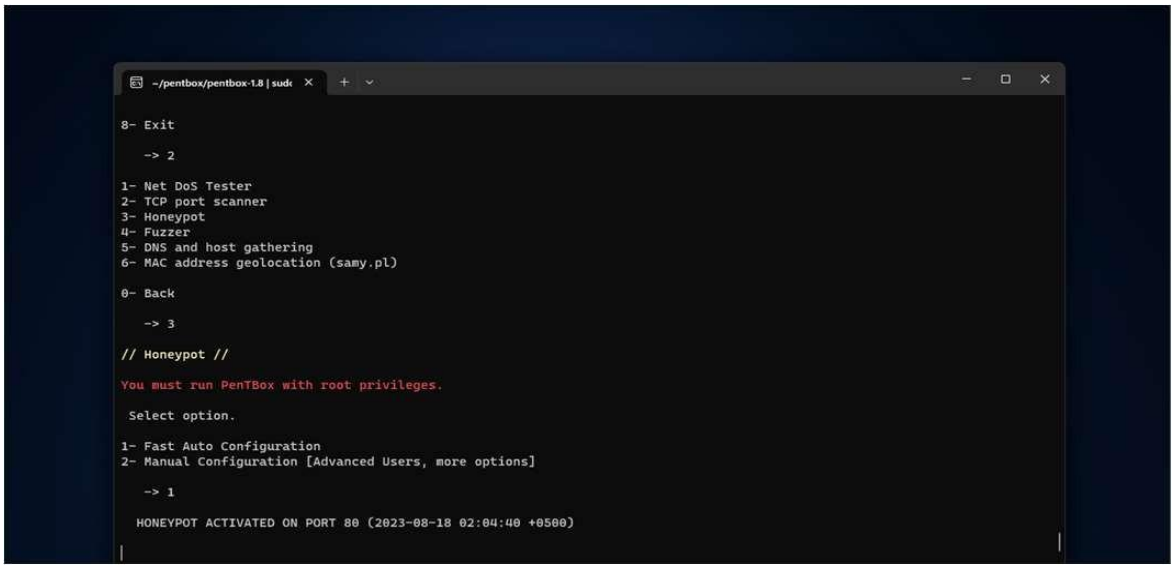

Fig 6. Setting up Honeypot.

To check what's coming up on port 80 we must know our ip address. One can check it using the ifconfig command in a new terminal. In our case, it's 172.21.100.232. Fig 7 depicts the same.

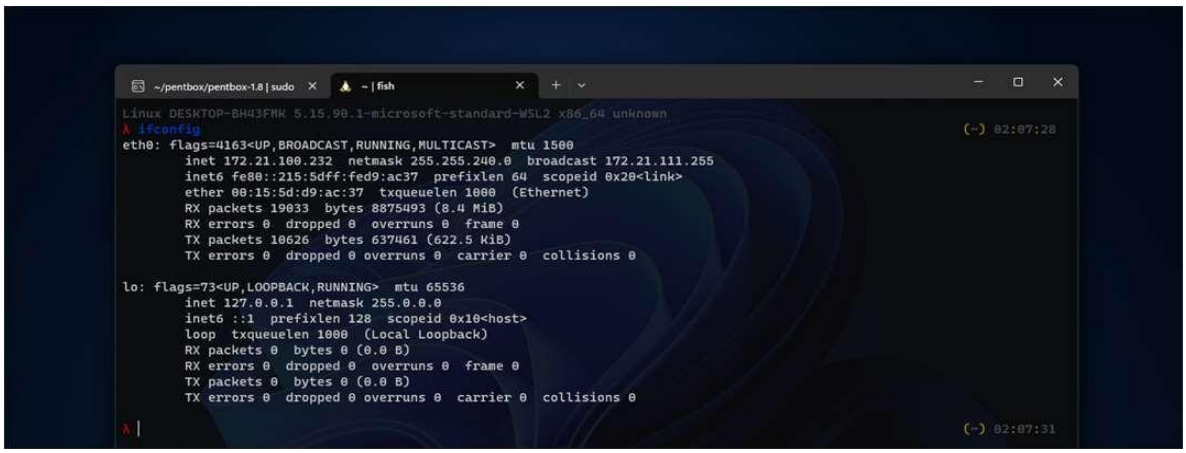

Fig 7. Checking IP.

Step 3: Checking honeypot in-action

If one goes to check this ip, they should get the output a in fig 8.

Fig 8. Access Denied Message.

On the Linux terminal, it is seen that the sample intrusion is being recorded and the headers are also being captured. The output window can be seen in fig 9.

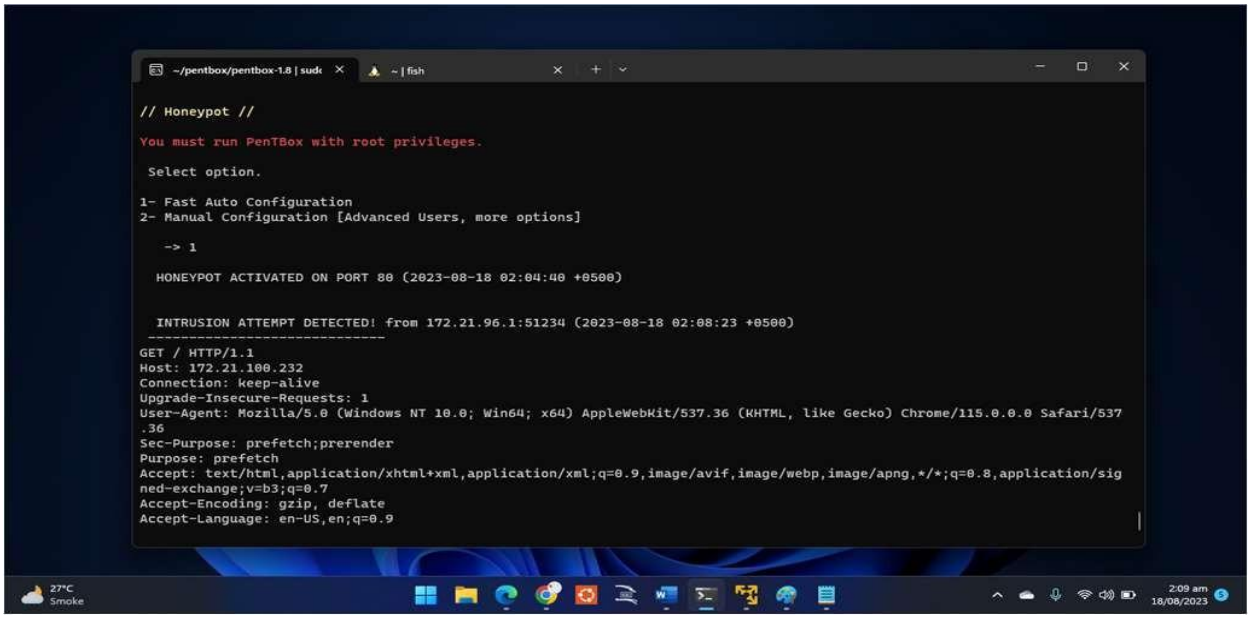

Fig 9: Intrusion Recorded.

One need to press Ctrl+c to stop this honeypot and also run it manually.

Step 4: SYN-Flooding on Honeypot and capturing traffic in Wireshark

The attack can be performed SYN-flood by running the command: “Sudo hping3 -S -p 129 –flood 172.21.100.232”. One can notice that the traffic is being captured by Wireshark and logs are being recorded in the honeypot too.

Step 5: Plotting traffic graph and data logging

In Wireshark, one can also go to statistics > I/O Graphs and see this sudden spike which confirms our DoS attack. The graph thus plotted is captured in fig 10. This confirms the successful DoS attack.

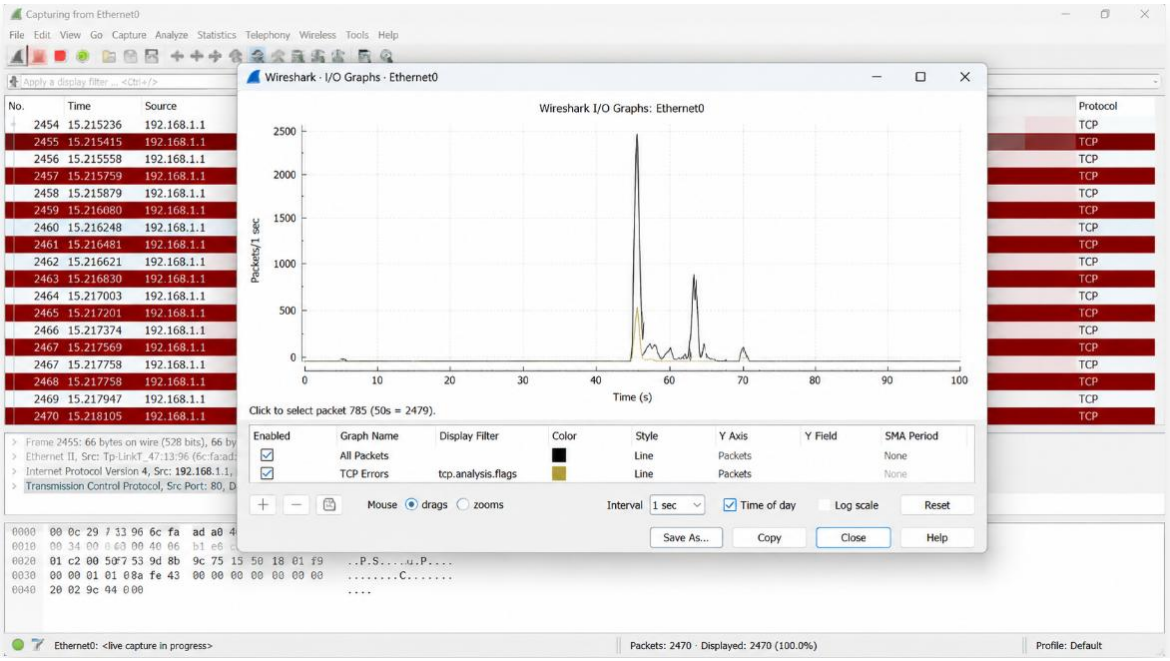

Fig 10. DoS attack confirmation

B: Results of the Machine Learning Algorithm

Step 1: Data Visualization

It involved various graphs and plots to better visualize the data in the dataset. Some of the data visualization plots include following *Distribution of URLs in the Dataset*: malicious vs Benign: This plot shows how much percentage of the dataset is composed of benign URLs and how much is malicious URLs, as in fig 11.

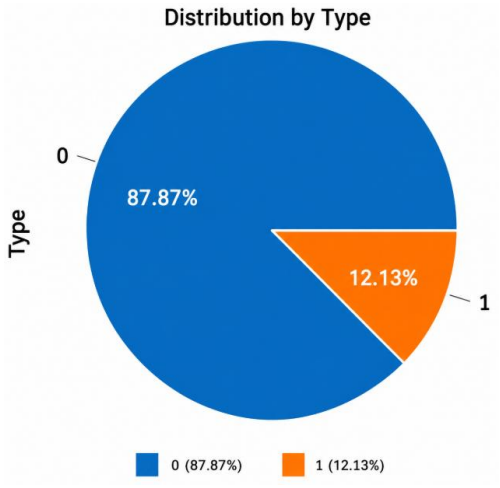

Fig 11. Distribution of URLs in the Dataset: malicious vs Benign (Code Output screenshot)

Step 1: Box Plots

Boxplots are helpful graphs to understand how a certain column is distributed. Box plot correlating number of special characters to type - malicious or benign. Looking at the plot in fig 12, it is evident that a special character word count between 13-20 has higher chances of being malicious.

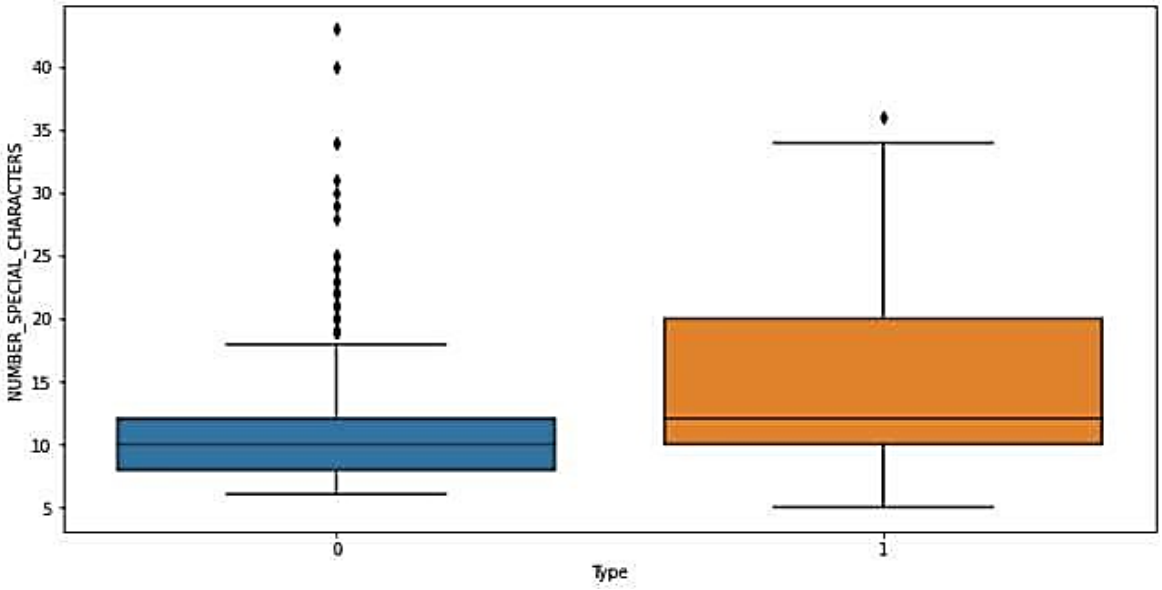

Fig 12. Boxplot - Number of special characters (Code output screenshot).

Step 2: Modelling and Model Evaluation

To automate the operations of model training and assessment, some helper functions are defined and multiple models are trained.

1. CNN with Random Forest hybrid

Post training on CNN + Random Forest hybrid, the confusion matrix and assessment measures like the f1-score, precision, and recall are obtained. Below is the confusion matrix (fig 13a) and performance metrics (fig 13b) for the trained model.

Accuracy	:	95.92%
Precision (Malicious)	:	100.00%
Recall (Malicious)	:	64.29%
F1-Score (Malicious)	:	78.26%
False Positive Rate	:	0.00%
False Negative Rate	:	35.71%

Fig 13(a). CNN + Random Forest Confusion Matrix (Code output screenshot).

	precision	recall	f1-score	support
0	0.95	1.00	0.98	305
1	1.00	0.64	0.78	42
accuracy			0.96	347
macro avg	0.98	0.82	0.88	347
weighted avg	0.96	0.96	0.95	347

Fig 13(b). Random Forest Performance Metrics (Code Output screenshot).

2. CNN with Naive Bayes Hybrid Model

Similarly, after the CNN + Naïve Bayes hybrid algorithm is trained, the confusion matrix and assessment measures like the f1-score, precision, and recall are obtained. The confusion matrix and performance metrics for the trained model are captured in fig 14(a) and fig 14(b) respectively.

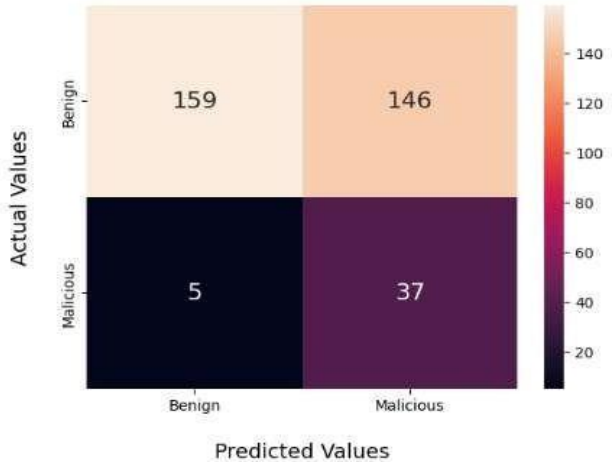

Fig 14(a). CNN + Naive Bayes hybrid confusion matrix (Code output screenshot)

	precision	recall	f1-score	support
0	0.97	0.52	0.68	305
1	0.20	0.88	0.33	42
accuracy			0.56	347
macro avg	0.59	0.70	0.50	347
weighted avg	0.88	0.56	0.64	347

Fig 14(b). CNN + Naive Bayes performance metrics (Code Output screenshot)

3. CNN + Logistic Regression Model

The same process is done for the CNN + Logistic Regression model, and post training, the confusion matrix and assessment measures like the accuracy, f1-score, precision, and recall are obtained. Fig 15(a) shows the confusion matrix and performance metrics in fig 15(b) thus obtained.

Fig 15(a). CNN + Logistic Regression confusion matrix (Code output screenshot).

	precision	recall	f1-score	support
0	0.95	1.00	0.97	305
1	0.96	0.60	0.74	42
accuracy			0.95	347
macro avg	0.95	0.80	0.85	347
weighted avg	0.95	0.95	0.94	347

Fig 15(b). CNN + Logistic Regression performance metrics (Code Output screenshot).

4. CNN with KNN hybrid Model

On applying the same implementation to the CNN + KNN Model, the confusion matrix and performance metrics for the trained model are recorded as in fig 16(a) and fig 16(b) respectively.

Fig 16(a). CNN with KNN confusion matrix (Code output screenshot).

5. CNN with KNN hybrid Model

On applying the same implementation to the CNN + KNN Model, the confusion matrix and performance metrics for the trained model are recorded as in fig 16(a) and fig 16(b) respectively.

Fig 16(b). CNN with KNN confusion matrix (Code output screenshot).

Step 3. Model Comparison

Below table I collates all the models' performance. It is evident that the Random Forest algorithm performs the best amongst all the 4 algorithms under consideration, because it has highest accuracy, precision, recall and F1 score.

Table 1. Models' Performance Comparison

Model	*Accuracy*	*Precision*	*Recall*	*F1-Score*
CNN + Random Forest	0.96	0.98	0.82	0.88
CNN + Naive Bayes	0.88	0.59	0.70	0.50
CNN + Logistic Regression	0.95	0.95	0.80	0.85
CNN + KNN	0.94	0.94	0.81	0.86

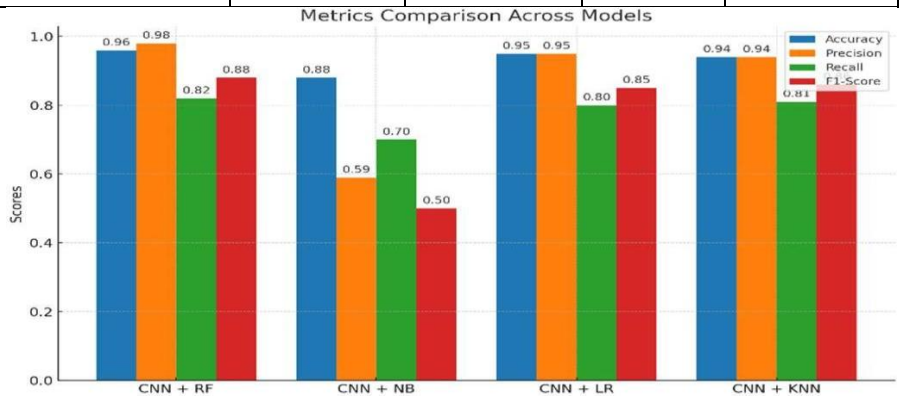

Fig 17. Graphical comparison

5. Discussion

A: Analysis of the findings

The results from the two-phase approach of this study—honeypot implementation followed by machine learning-based attack detection—provide valuable insights into the effectiveness of these techniques in modern

cybersecurity. *Honeypot Implementation Results:* The use of honeypot, which involved the use of Pentbox tool as well as a Flask-based application honeypot, shows that honeypots are a flexible security tool.

- **Effectiveness of Pentbox**: The Pentbox tool which is a rapid deployment honeypot showed how honeypots can easily be deployed in places where there is need to monitor threats on-the-fly. This logging of intrusion attempts with little configuration suggests that such tools are very useful for organizations that may not have the manpower or technical know-how of handling cyber security issues. The logs from the SYN-flood attack show how effectively the honeypot can capture and record specifics of typical attacks such as DoS attacks.
- **Flexibility of Custom Solutions (Flask-based Honeypot):** The use of the custom Flask-based honeypot showed the versatility of the approach of building solutions from scratch for given security requirements. While using the Pentbox tool the user has limited options on what data is captured and how it is analyzed, the custom honeypot gave the ability to capture more data and analyze it in a more detailed manner. This is especially the case when certain classes of attacks or detailed analysis of the systems' state are needed. The capability of capturing HTTP requests and responses and the headers and payloads of these requests and responses provide more insight into the ways attackers are likely to approach a target system and the types of attacks they are likely to launch.
- **The outcome of using the developed hybrid deep learning model in detecting attacks**: The use of deep learning algorithms to identify the malicious activities based on the logs created by the honeypot is an added security measure to network security. The results from the various models tested—Random Forest, Naive Bayes, KNN, and Logistic Regression—provide several key insights.
- **CNN + Random Forest as the Top Performer:** The CNN+ Random Forest model was identified to be the best among all the algorithms with an accuracy of 0.96, precision of 0.98, recall of 0.82 and F1-score of 0.88. This high level of performance can be attributed to the fact that Random Forest is an ensemble method which consists of multiple decision trees to minimize the problem of overfitting. The fact that the algorithm can work with both nominal and ordinal data, as well as with the noise, was also an advantage. This high precision is evident in the low false positive rate, which is very important in cybersecurity because of the problem of alert fatigue.
- **Insight into Feature Importance**: The accuracy of the CNN + Random Forest model also gives information about the significance of different features in identifying the malicious URLs. The decision of the model can be examined in order to identify which of the features (for example, the number of special characters in a URL, the length of the URL or the existence of some patterns) is more indicative of malicious behaviour. This feature importance analysis could help in the future to focus on the further development of these models by optimizing the features and increasing the detection rates.
- **Challenges with CNN + Naive Bayes**: The CNN + Naive Bayes classifier, which gave the lowest accuracy among the models used in the study, shows the weakness of probabilistic models in complex datasets and scenarios as the one considered in this case for cybersecurity. The naive assumption that features are independent of each other in Naive Bayes does not apply in most of the real-life problems as the presence of one feature affects the probability of occurrence of the other. For instance, in URL classification, the number of special symbols and the length of the URL may be related, and this relation is not well handled by Naive Bayes. This result indicates that Naive Bayes might be fine for very basic problems or when the classes are clearly distinguishable, but it is not ideal for complex problems where subtle patterns or associations must be identified.
- **Balanced Performance of CNN + KNN and CNN + Logistic Regression**: The performance of KNN and Logistic Regression was also quite good, though lesser than that of Random Forest. This is because KNN is based on proximity in feature space and therefore is good when the URLs are of two clear types, say malicious and benign while it may not perform well when the features of the malicious and benign URLs are very similar. The algorithm that proved successful was Logistic Regression, which offers probability outputs; its performance was high in cases where data is linearly separable. However, it was slightly slower than Random Forest, indicating that although it is a stable model, it may not be as good at fitting non-linear relationships.
- **Model Interpretability vs. Complexity:** The results also raise the issue of the trade-off between model interpretability and model complexity. Despite the fact that Random Forest gives better accuracy than Logistic Regression, it is more complex and less interpretable. In cases where the decision-making process of the model is important (for example, for compliance or forensic purposes), even though Logistic Regression is slightly worse in terms of performance, it may be used.

B: Real World Applications

The research's discoveries and methodology have resulted in various real-world applications:

1. Honeypot-based Cyberattack Detection

The positive impact of the honeypot system for the identification of cyberattacks in the framework of the project on the effectiveness of the approaches used in the sphere of cybersecurity is also worth noting. By creating honeypots that resemble real systems potential attackers can be enticed and the intentions and goals of these attackers can be analyzed. Such information can be used for enhancing the protection and to act against such threats in the cyber space prior to it.

2. ML Based cybersecurity in Vehicular networks (VANETS)

Applying honeypots in VANETs can track unauthorized attempts to enter the network and employing machine learning algorithms to analyse the traffic and predict possible cyber-attacks to enhance the security of VANET communication systems. The importance of cyber-security in intra-vehicular communication was brought out in [17].

3. Integration of Machine Learning for Pre-emptive reaction

The advantage of employing pre-emptive reaction based on machine learning algorithms is great in practical applications. These algorithms can be used by organizations to analyze the incoming data and identify some patterns that indicate a cyberattack. This makes it possible to take fast actions such as notifying security professionals or implementing preventive measures thus minimizing the effects of cyber disasters. This was especially seen in [18].

4. Using Machine Learning for Enhanced Detection

The use of machine learning algorithms in detection of cyberattacks in real-time has a significance in the cybersecurity field. These algorithms can be used in the systems of businesses and organizations to find out the unlawful actions and the divergence from the norm in the course of carrying out their activities. It is crucial because it strengthens the assessment of threats and their timely response.

6. Conclusion

The present research was able to place a honeypot system and integrate machine learning models to enhance the detection and mitigation of cyber threats. The honeypot was set using the Pen test tools and can capture and record the following evil deeds: It was able to capture a SYN-flood attack with real time data logging as shown in the Wireshark. The successful implementation helped to make the honeypot effectively monitor the network, identify intrusions, as well as gather information for further analysis. The research emphasizes the applicability of honeypots in realistic scenarios of cybersecurity and proves that honeypots can be an effective tool for protection against possible threats. Besides the honeypot implementation, the research used machine learning to categorise URLs as either safe or unsafe. The models that were assessed were CNN hybridized with each of Random Forest, Naive Bayes, Logistic Regression and KNN. The Random Forest model was identified to be the best model with an accuracy of 0.96 and precision of 0.98, and F1-score of 0.88, outperforming other algorithms. This result supports the idea of machine learning as an efficient tool for classifying cyber threats, and provides a direction for the development of automated threat identification systems that can be implemented in real-time cybersecurity systems.

In conclusion, the presented study proves the possibilities of integrating honeypot systems with machine learning algorithms to improve the cybersecurity. That is why the honeypot system is useful because it registers all the malicious activities that occur in the specified period. The machine learning models, especially the Random Forest, provide high accuracy in threat detection, which can be used to prevent cyberattacks. This combined approach does not only enhance the threat detection but also has a vast implication for cybersecurity applications in the real world where organizations can get the necessary tools to counter new emerging threats.

REFERENCES

1. P. Patel, A. Dalvi and I. Sidddavatam, "Exploiting Honeypot for Cryptojacking: The other side of the story of honeypot deployment," 2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA, Pune, India, 2022, pp. 1-5, doi: 10.1109/ICCUBEA54992.2022.10010904.
2. G. A. Jude Saskara, I. K. R. Arthana and P. B. Megawanta, "Simulation and Performance Testing of the Ganesha Honeypot System (GHOST) for SSH Security," 2023 1st International Conference on Advanced Engineering and Technologies (ICONNIC), Kediri, Indonesia, 2023, pp. 55-59, doi: 10.1109/ICONNIC59854.2023.10467574.

3. J. Buzzio-Garcia, "Creation of a High-Interaction Honeypot System based-on Docker containers," 2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4), London, United Kingdom, 2021, pp. 146-151, doi: 10.1109/WorldS451998.2021.9514022.
4. M. A. Lihet and V. Dadarlat, "How to build a honeypot System in the cloud," 2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER), Craiova, Romania, 2015, pp. 190-194, doi: 10.1109/RoEduNet.2015.7311992.
5. Hongxia Li, Junming Chen and Xin Jin, "An outlook on network honeypot," 2011 International Conference on Computer Science and Service System (CSSS), Nanjing, 2011, pp. 1102-1105, doi: 10.1109/CSSS.2011.5972238.
6. M. S. Zemene and P. S. Avadhani, "Implementing high interaction honeypot to study SSH attacks," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 2015, pp. 1898-1903, doi: 10.1109/ICACCI.2015.7275895.
7. N. M. Danchenko, A. O. Prokofiev and D. S. Silnov, "Detecting suspicious activity on remote desktop protocols using Honeypot system," 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2017, pp. 127-128, doi: 10.1109/EIConRus.2017.7910509.
8. V. V. Polyakov and S. A. Lapin, "Architecture of the Honeypot System for Studying Targeted Attacks," 2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE), Novosibirsk, Russia, 2018, pp. 202-205, doi: 10.1109/APEIE.2018.8545323.
9. B. Nagpal, N. Singh, N. Chauhan and P. Sharma, "CATCH: Comparison and analysis of tools covering honeypots," 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, India, 2015, pp. 783-786, doi: 10.1109/ICACEA.2015.7164809.
10. A. Kyriakou and N. Sklavos, "Container-Based Honeypot Deployment for the Analysis of Malicious Activity," 2018 Global Information Infrastructure and Networking Symposium (GIIS), Thessaloniki, Greece, 2018, pp. 1-4, doi: 10.1109/GIIS.2018.8635778.
11. D. Commey, S. Hounsinou and G. V. Crosby, "Strategic Deployment of Honeypots in Blockchain-based IoT Systems," 2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS), Abu Dhabi, United Arab Emirates, 2024, pp. 134-138, doi: 10.1109/AICAS59952.2024.10595866.
12. J. Haseeb, M. Mansoori and I. Welch, "Failure Modes and Effects Analysis (FMEA) of Honeypot-Based Cybersecurity Experiment for IoT," 2021 IEEE 46th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 2021, pp. 645-648, doi: 10.1109/LCN52139.2021.9525010.
13. V. A. Memos and K. E. Psannis, "AI-Powered Honeypots for Enhanced IoT Botnet Detection," 2020 3rd World Symposium on Communication Engineering (WSCE), Thessaloniki, Greece, 2020, pp. 64-68, doi: 10.1109/WSCE51339.2020.9275581.
14. R. Vishwakarma and A. K. Jain, "A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 1019-1024, doi: 10.1109/ICOEI.2019.8862720.
15. J. Z. Guizhou and Z. Liu, "New honeypot system and its application in security of employment network," 2012 IEEE Symposium on Robotics and Applications (ISRA), Kuala Lumpur, Malaysia, 2012, pp. 627-629, doi: 10.1109/ISRA.2012.6219267.
16. A. D. Oza, G. N. Kumar and M. Khorajiya, "Survey of Snaring Cyber Attacks on IoT Devices with Honeypots and Honeynets," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 2018, pp. 1-6, doi: 10.1109/I2CT.2018.8529510.
17. K. Kumarmanas, S. Praveen, V. Neema and S. Devendra, "An innovative device for monitoring and controlling vehicular movement in a Smart city," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, India, 2016, pp. 1-3, doi: 10.1109/CDAN.2016.7570882.
18. S. K. Udayakumar, H. Ragothaman and K. M. Khare, "A Novel Dataset and a Hybrid Ensemble Approach for Anomaly Detection in Enterprise-Access-Logs for Anomaly Detection," 2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), Chennai, India, 2025, pp. 1-6, doi: 10.1109/ICDSAAI65575.2025.11011761.