National College of Ireland

Developing an Advanced and Adaptive Framework of Honeypots for Efficient Deception for ZigBee IOT Environments

MSc Research Project
Master of Science in Cyber Security

Shashank Basavaraju
Student ID: 22241817

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shashank Basavaraju		
Student ID:	22241817		
Programme:	Master Of Science in Cybersecurity	**Year:**	2023-2024
Module:	Practicum Part2		
Supervisor:	Michael Prior		
Submission Due Date:	12th August 2024		
Project Title:	MSc Research Project Part 2		
Word Count:	6875	**Page Count:**	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shashank Basavaraju

Date: 11th August 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 own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

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

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

Developing an Advanced and Adaptive Framework of Honeypots for Efficient Deception for ZigBee IOT Environments

Shashank Basavaraju

Student ID: 22241817

Abstract

Cyber threats continue to be one of the most persistent security concerns in modern digital settings. This study deals with these issues using honeypots, integrated with an intrusion detection system that uses deep learning. Utilizing them to entice potential attackers into the open to demonstrate their methods, captured traffic is analysed using a GRU-based deep learning model to find intrusion attempts. The hardware setup comprises ESP32 microcontrollers connected with DHT11 sensors to achieve the environmental data, XBee modules for communication, OLED displays for data visualization, and a CC2531 ZigBee sniffer for packet analysis. In action, the honeypot captures and preserves the traffic within a controlled environment, where the attackers lead themselves unknowingly to reveal their strategies. Tools like hping3 and Hydra are used to test their attacks to understand the attacker procedure. This framework is coming with lots of advantage on the front of the identification and mitigation of the threat on cyberspace through being proactive. Since the model integrates CNN for feature extraction and GRU for sequence analysis, it is feature-rich to detect any sophisticated pattern of attacks. In addition to that, the tricking nature of the honeypot will help in the early detection of the threats to be attacked and will further enhance the overall cybersecurity defence against the dynamic threat landscapes. The CNN-GRU model is trained using custom datasets created by Wireshark achieved an amazing level of 94% testing accuracy for distinguishing malicious traffic.

1 Introduction

1.1 Background

Cyber threats are those attacks or events that may impair computer systems and gain unauthorized access to them. Targeting critical assets in industries, governments and personal networks, new types of threats are born again every year. These threats continue to change in new and improved breeds, making the task of being able discover them that much harder. One example is The Mirai botnet identified back in 2016, belonging to a family of malware that find deprived IoT devices and employ them to hold out enormous DDoS attacks within any large network. After Mirai, the attackers created its iterations like Persirai to boost their malicious operations while remaining undetectable. Botnet incidents escalated significantly after the introduction of Mirai (Javadpour *et al*., 2024). Two common practices used in cybersecurity to

defend against these threats are intrusion detection and honeypots. Intrusion Detection Systems are tools or applications used to monitor networks as well as systems and they inspect everything that passes from an entity within a different one, in particular the suspicious activities signs like unsanctioned unauthorized access requests of data thefts or even protocol breaches. They are required to detect new categories of attacks (both known and unknown) from internal and external sources. Network Intrusion Detection Systems (NIDS), on the other hand, are unlike traditional firewalls and are installed at a strategic location in your network to actively monitor all traffic as it comes in. Including those that detect malicious network behaviour and any use of a computer in an unauthorized or unwanted manner; IDSs detect active threats continuously and prevent attackers from successfully attacking the network (Wanjau, Wambugu, and Oirere, 2022). IDSs are categorized into host-based IDS (HIDS) and network-based IDS (NIDS), depending on what they monitor. HIDS observes and analyses activities on the system (such as a computer or server) where it's installed. It focuses on monitoring the system's behaviour and current state. In contrast, NIDS monitors network traffic to detect attacks coming through network connections. It is deployed within networks to scrutinize traffic passing through specific devices. NIDS systems can be implemented as hardware or software and typically feature two network connections: one for monitoring network conversations and another for transmitting detection reports (Abdallah, Eleisah, and Otoom, 2022).

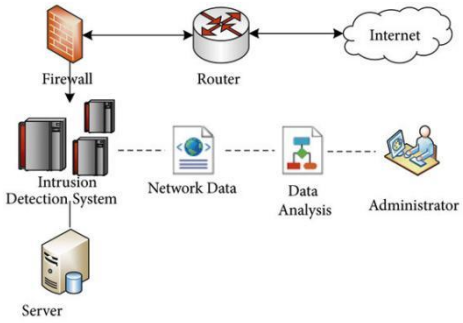


Figure (1): Architecture of an Intrusion Detection System (Wang, Wang and Xie, 2022)

The Figure (1) shows the architecture of an Intrusion Detection System (IDS). Network data flows from the internet through a firewall and router before being monitored by the IDS. The IDS analyses this network data and communicates findings to an administrator, who oversees the process. A server supports data storage and additional processing needs.

Honeypots are great for drawing attackers in, and learning how they operate to help gain an edge defensively. Many organizations have been deploying these kinds of systems In order to improve their security programs and obtain visibility into more frequent types of attacks as a result. Honeypots provide defenders with information on the attacks that are being launched against their system (Zielinski and Kholidy, 2022). Honeypot systems are the main defence to stop or timely detect attacks and malicious activities. By providing specifics regarding both intrusion behaviours as well as system activities, and the techniques employed by malicious individuals, they help to enhance detection speeds in identifying threats along with response times of dealing

with inbound attacks. Therefore, honeypots work by deploying real systems to capture and study an adversarial behaviour efficiently (Titarmare, Hargule and Gupta, 2019).

Figure (2) shows how the honeypot operates. It displays two types of traffic: normal traffic and attack traffic. When traffic reaches the load balancer, it is directed to the Web Application Firewall (WAF). If the traffic is identified as an attack, it is routed to the clone server honeypot. This honeypot mimics a real server but is designed to trap and analyse malicious activities. On the other hand, normal traffic proceeds to the real server for legal interactions.

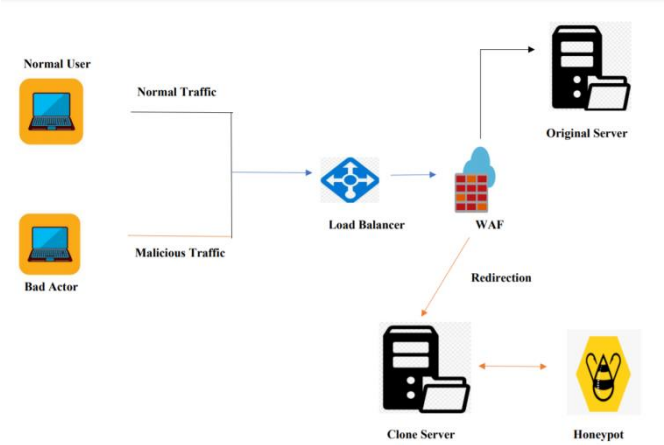


Figure (2): Overview of Honeypot Operation (CyberSRC, 2024)

1.2 Problem Definition

IDS and honeypots have been crucial for minimizing damage from cyber-attacks. However, cyber criminals have evolved their tactics over time. Today's IDS systems often struggle to keep up with these sophisticated attacks (Ozkan-Okay *et al*., 2021). As malicious activities continue to grow in complexity, current security measures are proving inadequate. IDS works on the idea that intruders behave differently from legitimate users, but distinguishing between normal and abnormal behaviour isn't always clear-cut(Ashiku and Dagli, 2021). Over time, attackers have become adept at identifying honeypots systems designed to lure and trap attackers. Once recognized, attackers typically stop wasting effort on them, reducing the defender's ability to gather useful information. Effective honeypot systems must now mimic real systems closely and contain realistic but fake data (M.R. and P., 2022). This approach forces attackers to spend valuable time attempting to exploit the honeypot, looking for signs like valid credentials. Deep Neural Networks(DNN) can enhance IDS by learning to detect both known and new types of network behaviours, including those that haven't been identified before. This capability helps identify intruders and lowers the chances of a system being compromised (Ashiku and Dagli, 2021). In this study proposes an intelligent IDS using DNN for real-time intrusion detection, alongside an advanced honeypot that evades detection by attackers, is proposed. This combined approach aims to enhance cybersecurity by improving detection accuracy and resistance against evolving cyber threats.

The study has the following research question:

How does the security system consisting of the honeypot integrated with deep learning methods to effectively trick and detect attacks in IoT environments?

The aim of the project is to create and evaluate a honeypot framework for ZigBee IoT services. The objectives of the study are:

1. Build a micro controller setup for capturing temperature and humidity.

2. Setting up ZigBee communication between XBee modules and ESP32 and connect a ZigBee sniffer tool to a PC for packet sniffing and analysis.

3. Train a deep learning model using GRU architecture on the custom dataset for real-time intrusion detection.

4. Create a honeypot that captures ZigBee traffic and stores data packet files.

5. Design a landing webpage to display captured data and simulate attacks using tools like hping.

6. Integrate the trained deep learning model into the system to detect and classify attacks.

The novelty of the system proposed here is that it combines a honeypot with real-time intrusion detection using deep learning. This setup not only tricks attackers but also quickly identifies and analyses their actions. By capturing detailed data and using smart models, the system provides a stronger defence against cyber threats in ZigBee IoT environments.

Section 1 of the report introduces the project, explaining IDS, honeypots, and the system being developed. Section 2 covers related work in the field. Section 3 explains the methods used in the study. Section 4 provides the specifications for the different components of the study. Section 5 describes the final implementation of the system. Section 6 discusses the evaluation of the study's results. Section 7 offers the conclusion and suggestions for future improvements.

2 Literature review

This part of the literature review explores past research, and the difficulties faced. It looks at older machine learning and deep learning methods used for spotting attacks, paying attention to the usual approaches used and tools like Snort and Suricata. The main aim of this study is to improve how attacks are detected by using newer, smarter deep learning methods that can detect attacks more accurately and quickly.

2.1 Machine learning and Deep learning methods

Looking at different types of connected intrusion detection systems that use machine learning methods, as discussed in the study by (Abdallah, Eleisah, and Otoom, 2022). It utilizes four prominent datasets: KDD’99, NSL-KDD, CICIDS2017, and UNSW-

NB15. Emphasizing the importance of feature selection, the study highlights how selecting the right features can significantly enhance performance. To address data imbalance, the study employs various sampling approaches to ensure more accurate and representative results. The study employs various supervised learning algorithms, with Random Forest and Deep Neural Networks (DNN) frequently emerging as the most effective. These algorithms achieve high classification performance, with accuracy rates up to 99.7% and low false positive rates (FPR) of 0.005%. This indicates their robustness in identifying and classifying intrusions accurately, making them suitable for practical implementation in real-world scenarios. However, the study also acknowledges certain limitations. One significant challenge is the requirement for substantial computational resources. Large datasets necessitate the use of deep learning techniques for optimal performance, which can be resource-intensive and time-consuming. This limitation highlights the need for more efficient algorithms and hardware to manage extensive data effectively.

A comprehensive analysis of Network Intrusion Detection System techniques utilizing Machine Learning, focusing on signature-based, anomaly-based, and hybrid detection methods, was conducted in the study by (Aziz, Siddiqi, 2021). It uses datasets like KDD99 and UNSW-NB15 to evaluate various ML approaches, including Support Vector Machine (SVM), Decision Trees, Neural Networks, and others. Results indicate that models like J48 and Random Forest outperform others, achieving high accuracy in detecting various types of attacks such as DOS, Probe, U2R, and R2L. These models demonstrate robust performance, with Random Forest often cited for its high accuracy and low false positive rates. However, the study does not provide specific performance metrics, making it challenging to quantify the exact improvements offered by these models. A notable strength of the study is its exploration of multiple machine learning techniques and their application to intrusion detection. By comparing different methods, the study offers valuable insights into which algorithms are most effective under various conditions. The use of well-known datasets like KDD99 and UNSW-NB15 ensures that the findings are grounded in widely recognized benchmarks. However, the study has some limitations. One significant limitation is the need for standardized datasets. The lack of consistency in datasets can hinder the ability to generalize findings across different network environments. Another challenge is adapting to the evolving nature of cyber-attacks. As attackers develop new methods, NIDS must continuously adapt to maintain effectiveness, a task that is inherently complex and resource intensive. Additionally, the study did not explore Recurrent Neural Networks (RNN) such as Gated Recurrent Units (GRU). GRUs could potentially offer advantages in processing sequential data and improving the accuracy of intrusion detection systems. By not considering GRUs, the study misses an opportunity to evaluate a powerful method that could enhance performance, especially in handling time-series data common in network traffic.

Machine learning algorithms and feature selection techniques to analyse network traffic and detect intrusions were discussed in the study by (Malhotra and Sharma, 2019). Utilizing the NSL-KDD dataset, which consists of 125,973 instances with 41 attributes, the study evaluates the performance of 10 popular ML algorithms using WEKA. Additional methods include Wrapper and Filter methods for feature selection and discretization to transform numeric data into nominal data. Results indicate that the Random Forest classifier achieves the highest accuracy of 99.91% with a model

building time of 191 seconds. The Random Tree classifier presents a good balance of accuracy (99.76%) and speed (3.24 seconds), making it a viable option for real-time applications. The study's findings underscore the importance of feature selection in enhancing the performance of ML models in intrusion detection. The use of Wrapper and Filter methods for feature selection highlights the study's methodological rigor. Wrapper methods evaluate the performance of a subset of features based on a specific algorithm, while Filter methods assess features independently of the algorithm. These approaches help in identifying the most relevant features, thereby improving the model's accuracy and efficiency. The study acknowledges some limitations. One notable limitation is the potential performance variability across different datasets. The results obtained using the NSL-KDD dataset may not be directly applicable to other datasets, which could affect the generalizability of the findings. This highlights the need for further research to validate the models across diverse datasets.

Machine learning techniques to analyse and interpret data across various fields, such as computer vision, stroke identification, credit card fraud detection, e-learning systems, cybersecurity, marketing, and more, were discussed in the study by (Kok, S. *et al*,2019). It utilizes data from diverse sources tailored to the specific requirements of each domain, ensuring a comprehensive approach to problem-solving. Advanced analytics, deep learning algorithms, and quantum-inspired techniques are also employed to enhance the predictive capabilities and reliability of the models. The results of the study are significant, with various models achieving notable accuracy rates. However, the study does not specify the numerical values or indicate which model outperforms the others. This omission makes it difficult to determine the relative performance of different models. A common limitation across these studies is the potential for overfitting, where models perform exceptionally well on training data but fail to generalize to new, unseen data.

Machine learning algorithms, specifically Random Forest and Support Vector Machine, focusing on feature selection using Recursive Feature Elimination, were examined in the study by (Patgiri *et al*., 2018). The NSL-KDD dataset, renowned for its cleanliness and 41 features per dataset, was utilized for analysis. Various feature selection techniques, including filter, wrapper, and embedded methods, were applied to identify informative and independent features. Results showed that before feature selection, Random Forest outperformed SVM, but after RFE, SVM demonstrated superior performance for most attack types. The study presented confusion matrices and accuracy percentages for each model and attack type. A notable limitation of the study was the time-consuming nature and performance degradation observed when using all features, underscoring the importance of feature selection in enhancing model efficiency and accuracy. However, the study did not explore neural network models, representing a significant limitation. Neural networks, particularly deep learning models, have shown promising results in intrusion detection systems due to their ability to capture complex patterns in data.

Examining various machine learning algorithms for Intrusion Detection System classification, utilizing the KDD-99 and NSL-KDD datasets to assess performance in accuracy, detection rate, and false alarm rate, was the focus of the study by (Rama Devi and Abualkibash, 2019). The datasets contain network traffic instances with features categorized as Basic, Traffic, and Content, with attacks classified into four

types: DOS, R2L, U2R, and Probe. Supervised learning algorithms like Logistic Regression, K Nearest Neighbour, Decision Tree, Support Vector Machine, Random Forest, AdaBoost, Multi-Layer Perceptron , and Naive Bayes, alongside unsupervised techniques like K-means and the Apriori Algorithm, were employed in the study. Results highlight the Random Forest algorithm's superior performance, achieving an accuracy of 99.7% with a false alarm rate of 3.2% on the NSL-KDD dataset, outperforming other models.

Machine learning and deep learning techniques were employed to develop Intrusion Detection Systems for IoT networks, with a focus on comparing various ML approaches, as discussed in the study by (Baich *et al*., 2022). The NSL-KDD dataset, comprising 148,517 records with 14 features, served as the basis, covering binary and multiclass intrusion scenarios. The study utilized feature selection methods like Pearson correlation and Fisher score, along with feature extraction via Principal Component Analysis (PCA). Notably, the Decision Tree with Fisher score emerged as the top-performing model, boasting a remarkable 99.26% accuracy and swift 0.4 seconds prediction time. However, the study faces limitations, primarily stemming from the NSL-KDD dataset's lack of representation of real-world network environments. The study did not incorporate real-time testing, which is essential for evaluating the performance of IDS in dynamic IoT environments where network conditions and threats continuously evolve. Real-time testing could provide valuable insights into the IDS's responsiveness and effectiveness in detecting emerging threats promptly.

2.2 Intrusion Detection Using Snort And Suricata

The use of Snort for intrusion detection and digital forensics analysis was discussed in the study by (Olutayo, 2022). Data for this study came from various academic sources focusing on cyber forensics, intrusion detection, and network analysis. Using a mix of different methods, the study linked security events to find and examine unusual activities in the network. The results showed that a model combining data mining and forensic techniques worked the best. However, the study has a drawback as it didn't do its own practical testing and depended on data from other scholarly sources.

The analysis of how well Snort and Wireshark work for analysing network traffic was discussed in the study by (Jain & Anubha, 2021).They used Snort to capture live data packets, which then checks these packets against known signatures and sends an alert if it doesn't find a match. The captured data is then analysed in detail using Wireshark. The findings show that Snort and Wireshark are effective tools for network traffic analysis. Snort helps by detecting and alerting users about potential attacks, while Wireshark provides a detailed look at the captured packets. However, a limitation is that there might be times when an attack happens and no alert is generated, which could lead to missed detection and response opportunities.

Examining how well Suricata can detect various Indicators of Compromise was discussed in the study by (Raharjo and Salman, 2023). The research is split into two stages: design and testing. It describes the testing environment, how the dataset is created, the scenarios used, and the performance measures. They created a dataset of one million IoC parameters, including IPReps, HTTP, DNS, SSL JA3, and MD5

Hash, by changing network traffic data packets. Suricata was then used to detect these IoC parameters in five different scenarios. The results show that Suricata can identify rules for all IoC types, performing best with HTTP and SSL JA3. The study also notes that there isn't a straightforward link between the number of rules and the detection probability. However, the research acknowledges some limitations, such as issues with the testing environment and scenarios, no alerting system, and not using other tools like Snort or Wireshark.

2.3 Summary

The below tables summarize the studies analysed:

Study	Dataset	Model And Approach	Conclusions	Drawbacks
Abdallah, Eleisah, and Otoom, (2022)	KDD'99, NSL-KDD, CICIDS2017, and UNSW-NB15	Random Forest (RF) and Deep Neural Networks (DNN), emphasizes feature selection and sampling approaches to address data imbalance	Achieves high performance with accuracy rates up to 99.7% and low false positive rates (FPR) of 0.005%. Demonstrates robustness in identifying and classifying intrusions accurately.	Requires substantial computational resources and time consuming
Aziz, Siddiqi, (2021)	KDD99 and UNSW-NB15	Support Vector Machine (SVM), Decision Trees, and Neural Networks. Focused on Signature -based, anomaly-based and hybrid-based detection methods	Models J48 and RF outperforms others, achieving high accuracy in detecting attacks such as DOS, Probe, U2R, and R2L, with high accuracy and low false positive rates.	Since the study deals with sequential data could have explored into Recurrent Neural Networks like GRU which is good ay handling sequential data
Malhotra and Sharma, (2019)	NSL-KDD dataset	Employed 10 Machine Learning Algorithms using WEKA. Uswd Wrapper and Filter Techniques for feature selection and discretization.	RF achieved classified achieved 99.91% accuracy with a model building time of 191 seconds, hence making it suitable for real-time application.	The study's findings may not generalize across different datasets, necessitating further research for validation.
Patgiri *et al*., (2018)	NSL-KDD dataset	RF and SVM, Focusses on feature selection using Recursive Feature Elimination(RFE) along with filter, wrapper, and other embedded methods	Before feature selection, Random Forest outperformed SVM, but after applying RFE, SVM demonstrated superior performance for most attack types.	The study highlights the time-consuming nature and performance degradation when using all features.
Rama Devi and Abualkibash,	KDD-99 and NSL-KDD	K-Nearest Neighbour (KNN),	The Random Forest algorithm	Limitation of the study is its reliance

(2019)		Decision Tree, Support Vector Machine (SVM), Random Forest, AdaBoost, Multi-Layer Perceptron (MLP), and Naive Bayes	demonstrates superior performance, achieving an accuracy of 99.7% with a false alarm rate of 3.2% on the NSL-KDD dataset, outperforming other models.	on the KDD-99 and NSL-KDD datasets, which, while commonly used, may not fully represent the diversity of real-world network environments and evolving attack types.
Baich *et al*., (2022)	NSL-KDD dataset	Decision Tree and other Machine learning algorithms. Applied feature selection methods like Pearson correlation, Fisher score and feature extraction through PCA	The Decision Tree model achieved a 99.26% accuracy and a prediction time of 0.4 seconds. These results highlight the model's potential effectiveness for accurate and efficient intrusion detection in IoT networks.	Could have explored the options of deep learning models which would have helped achieve better results and faster computation.
Olutayo, (2022)	Used data from various academic sources and cyber forensics , IDS and network analysis.	The study employed a mix of different methods to link security events and identify unusual activities in the network, focusing on digital forensics for analysis	The study found that a model combining data mining and forensic techniques yielded the best results for identifying and examining unusual network activities. This approach showed promise in enhancing security event detection and analysis	A drawback of the study was its reliance on data from other scholarly sources without conducting its own practical testing. This limitation may affect the direct applicability and validation of the findings in real-world network environments.

Table (1): Summary table of the Machine Learning literature studies

Study	Method	Analysis	Drawbacks
Jain & Anubha, (2021)	Examined effectiveness of Snort and Wireshark in terms of analysing them against known signatures to detect potential attacks. Wireshark used for deep packet analysis.	Snort effectively detected and alerted users about potential attacks. Wireshark offered comprehensive insights into captured packets, aiding in detailed network traffic analysis.	Possibility of Snort failing to generate alerts for certain attacks, potentially leading to missed detection and response opportunities in some scenarios
Raharjo & Salman, (2023)	The study designed a testing environment, created a dataset of one million IoC parameters including IPReps, HTTP, DNS, SSL JA3, and MD5 Hash, and tested Suricata's performance in detecting these parameters	The study demonstrated that Suricata successfully identified rules for all IoC types, with particularly strong performance in detecting HTTP and SSL JA3 parameters. However, the study noted that the number of rules did not	Limitations of the study included issues with the testing environment and scenarios used, the absence of an alerting system to notify of detected IoCs in real-time, and the study's focus solely on Suricata without

	across different scenarios.	straightforwardly correlate with detection probability	comparing its performance against other tools like Snort or Wireshark.

Table (2): Summary table of the reference literature studies

From the above tables it can be understood that none of the studies proposed a system that uses honeypots and machine learning together for real-time intrusion detection. Instead, they relied on existing datasets for training and prediction. In contrast, this study introduces a system that integrates real-time detection using deep learning and honeypot techniques, consolidated within a web application for administrators to easily access detailed information.

3 Research Methodology

This research project introduces an innovative approach to enhancing security in ZigBee IoT environments through the development of an advanced and adaptive framework of honeypots. Using ESP32 as the main microcontroller, the framework integrates various hardware components such as DHT11 sensors, XBee modules, OLED displays, and the CC2531 Zigbee sniffing tool for robust packet analysis and data capture. The software part involves training a deep learning model on the IoT20 dataset to recognize and classify attack patterns, crucial for the honeypot's detection capabilities.

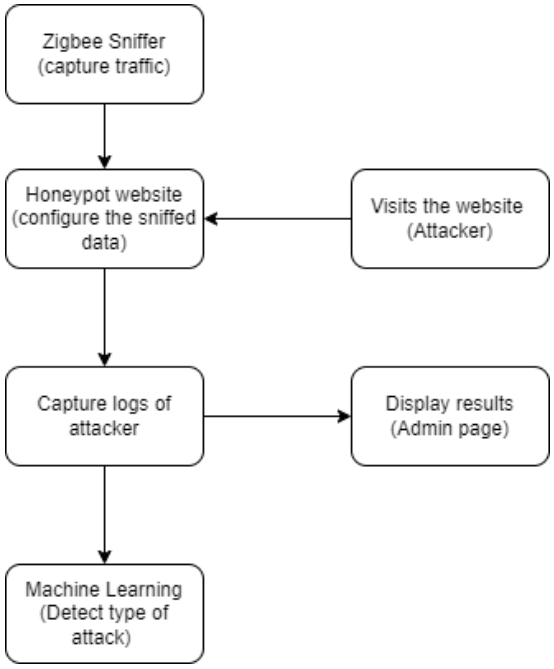


Figure (3): System Workflow for Honeypot-based Zigbee Network Intrusion Detection

The workflow shown in Figure (3) represents a Zigbee network intrusion detection system using a honeypot and machine learning. It begins with a Zigbee sniffer capturing network traffic, which is then configured into a honeypot website designed to attract attackers. When attackers visit this decoy website, their activities are logged. These logs are subsequently displayed on an admin page, providing administrators

with detailed insights into the attackers' behaviours and methods. The final step involves analysing these logs using machine learning algorithms to identify and classify the types of attacks. This systematic approach enables continuous monitoring and analysis, improving the security and resilience of IoT systems using Zigbee networks.

3.1 Data Fetching from DHT11

The DHT11 sensor is popular for measuring temperature and humidity because it is simple, affordable, and easy to use (Ztt1 *et al*., 2022). It can measure temperatures ranging from 0°C to 50°C with an accuracy of ±2°C and humidity levels from 20% to 90% relative humidity with an accuracy of ±5%. The sensor provides a digital output, which simplifies its integration with microcontrollers and other digital systems. Operating with low power, it is suitable for battery-powered applications. In this study data fetching from the DHT11 sensor involved connecting it to the ESP32 microcontroller to retrieve temperature and humidity readings. During the development phase, challenges related to library compatibility and integration were encountered, requiring troubleshooting to ensure reliable data acquisition. The ESP32, developed by Espressif Systems, is a powerful, versatile, and cost-effective microcontroller that has gained popularity among hobbyists, developers, and engineers, particularly for Internet of Things applications (Hübschmann, 2020).

3.2 Data Transmission via Zigbee Using XBee Modules

The XBee S2C 802.15.4 module is a robust and versatile wireless communication device designed for low-power, low-data-rate applications. It operates on the IEEE 802.ex15.4 standard, which forms the basis for Zigbee networks, offering reliable and secure communication over short distances (digi.com, 2024). The XCTU software is used to configure the XBee module. During the configuration process, there were some difficulties in setting up the module properly, requiring troubleshooting to ensure the correct settings for the application. Once configured, the data from the ESP32 module is transmitted to the XBee module serially. The ESP32 handles the data processing and then sends it to the XBee for wireless transmission. This ensures that the data is sent efficiently and accurately. On the receiving end, another XBee module receives the transmitted data, which is then sent serially to another ESP32 module. The ESP32 on the receiving side processes the data and displays it on an OLED screen. The communication is serial, ensuring a straightforward and reliable transfer of data. By overcoming the initial configuration challenges, the system is now capable of efficiently transmitting and displaying sensor data. This makes it a practical solution for various applications that require low-power, short-range wireless communication.

3.3 Packet Sniffing Using Zigbee USB Dongle Plus CC2531

Packet sniffing is performed between the communication of the ESP32 modules using the Zigbee USB Dongle Plus CC2531. The packet sniffer, provided by Texas Instruments, is used to analyse the data being transmitted. During this stage, a key challenge was finding the proper matching frequency to accurately capture the packets. Once this issue was resolved, the packet sniffer was able to intercept and log

the data exchanged between the XBee modules. The sniffed data is then exported as a log file for further analysis. This log file provides detailed insights into the data packets, helping to understand the communication flow and diagnose any potential issues. By addressing the frequency matching challenge, the system can now effectively monitor and analyse the wireless communication between the ESP32 modules. This step is crucial for verifying the correct operation and performance of the Zigbee network.

3.4 Training Deep Learning Model

3.4.1 Dataset Collection

The dataset for this project was custom-created using Wireshark to capture network traffic. Key features extracted from Wireshark include time, source, destination, protocol, length, and type (DDoS or normal). This process involved significant effort and time, especially in accurately labelling the data as either normal or indicative of a DDoS attack. The main difficulty faced during this phase was the time-consuming nature of data extraction and ensuring the accuracy of the captured features to create a reliable dataset for training the deep learning model.

3.4.2 Dataset Pre-processing

Getting the data ready is a key step before training a deep learning model. This means cleaning up the raw data and changing it into a format that the model can easily understand and learn from (Roy *et al*., 2019). In this project, preprocessing began with the removal of unwanted features to streamline the dataset. The selected features for analysis are Time, Protocol, Length, and type (DDoS or normal). Features such as Source, Destination, and Info were removed because they could introduce noise and redundancy, potentially leading to overfitting without contributing significantly to the model's ability to detect network anomalies. During preprocessing, label encoding was performed on the Protocol and type columns. This step converts categorical data into numerical format, making it compatible with the deep learning model. Label encoding ensures that the model can interpret these features correctly, enhancing its ability to learn from the data and improve its predictive accuracy. One of the challenges faced during preprocessing was ensuring that the encoding was consistent and correctly applied, as any discrepancies could result in misinterpretation by the model and affect its performance.

3.4.3 Model Creation, Compiling And Training

Deep learning works really well for tasks like image recognition, NLP, and running self-driving systems. By learning from large amounts of data, deep learning models can identify intricate patterns and make accurate predictions, making them powerful tools in modern AI applications (Santosh, Das and Ghosh, 2022). In this study, a Convolutional Neural Network combined with a Gated Recurrent Unit model is used. The CNN is adept at extracting features from input data, making it ideal for recognizing patterns within network traffic. The GRU, a type of recurrent neural network, is effective in handling sequential data and learning long-term dependencies, making it well-suited for time-series analysis in network traffic. The advantages of using a CNN-GRU model include its ability to automatically detect significant

features in the data and its proficiency in learning from sequential patterns, which are common in network traffic. This combination enhances the model's ability to detect and classify network anomalies accurately (Adryan Fitra Azyus, Sastra Kusuma Wijaya and Mohd Naved, 2023). The model in the study was created using the TensorFlow library, which provides a flexible and comprehensive framework for building and training deep learning models. The CNN layers identify patterns in the spatial aspects of the input data, while the GRU layers capture temporal dependencies. After defining the model architecture, it is compiled with the Adam optimizer, known for its efficiency and capability to handle sparse gradients. This compilation prepares the model for training, setting the stage for optimizing its parameters to achieve the best performance on the given dataset. Finally, the model was trained with five epochs and then saved for future use. One of the challenges faced during compilation was managing shape issues, which required careful adjustment of the model's input and output dimensions to ensure compatibility across all layers.

3.5 Honeypot Creation

A honeypot is a security tool created to identify, divert, or in some way prevent unauthorized attempts to use information systems (Javadpour *et al*., 2024). In this study, a honeypot was used to create a decoy environment that attracts attackers and captures details of their activities. This helps in understanding attack patterns and enhancing security measures. The honeypot was created using the Django library, specifically a library called admin honeypot. A middleware script was developed to capture and record attack details, which are then displayed in a table for analysis. In addition to capturing and recording attack details, the middleware script plays a crucial role in the honeypot setup. Implemented as a Django middleware using the MiddlewareMixin, it intercepts incoming requests to the web application. The script tracks client IP addresses and ports from request metadata, using a background thread to monitor and detect potential distributed denial-of-service attacks. This functionality helps in identifying patterns of suspicious activity, such as multiple requests from the same IP within a short timeframe. Detected attacks are logged in real-time to a file, providing security administrators with immediate visibility and enabling proactive responses to mitigate threats. One challenge encountered during the setup was dealing with version issues between Django and the admin honeypot library, which required careful troubleshooting and adjustments to ensure compatibility.

4 Design Specification

4.1 DHT11

The DHT11 sensor plays a crucial role in this study by gathering environmental data. It is often utilized for monitoring temperature and humidity levels because it is cost-effective and reliable. This sensor outputs a digital signal, which makes it compatible with a wide range of microcontrollers and IoT platforms (Islam *et al*., 2023).

4.2 ESP32

The ESP32 is a versatile and affordable platform ideal for creating IoT applications. Developed by Espressif Systems in Shanghai, China, it combines robust features

tailored for IoT use. Programming the ESP32 is flexible, supporting multiple development frameworks and languages. The popular choice is C++, and developers commonly use tools like the Arduino IDE or PlatformIO to program it (Hercog *et al*., 2023).

4.3 XBee

Xbee modules are radio frequency devices that operate based on the 802.15.4 standard, with some being fully ZigBee compliant. Initially developed for military use due to their vulnerability to interference, these modules now offer open access for public use. They use Direct Sequence Spread Spectrum technology, which provides robust wireless communication but requires more bandwidth. Xbee modules enable easy and dependable wireless serial communication between microcontrollers, computers, and systems equipped with serial ports (Hercog *et al*., 2023).

4.4 Zigbee

Zigbee is a wireless technology used for smart devices to communicate within a Personal Area Network (PAN), facilitating cost-effective, low-power connections for machine-to-machine (M2M) and Internet of Things (IoT) networks (Linda Rosencrance, 2017). Task Group 4 oversees ZigBee as part of the IEEE 802.15.4 standard, developed by the Zigbee Alliance. This standardizes the physical and Medium Access Control layers of the technology's architecture, with ZigBee focusing on enhancing the upper layers for improved functionality (Ahmed,2023).

4.5 CNN-GRU

A hybrid Convolutional-Recurrent Model combines the strengths of both networks: the feature extraction ability of convolutional layers and the sequential learning capability of recurrent computations. This integration enhances its effectiveness compared to other hybrid models. Specifically, the CNN GRU model leverages convolutional layers to capture spatial features from input data, while GRU handles sequential data to learn long-term dependencies. This dual approach enables the model to effectively analyse complex patterns and make accurate predictions, making it a robust choice for various tasks in machine learning and artificial intelligence applications (Jaiswal and Singh, 2022).

5 Implementation

In the implementation phase of this project, the focus is on creating a sophisticated honeypot framework tailored for ZigBee IoT environments. The main objective is to develop a robust system that effectively deceives potential attackers while providing valuable insights into network security threats. Hardware components include the integration of essential devices like the DHT11 sensor for monitoring temperature and humidity via ESP32 microcontrollers, XBee modules for ZigBee communication, an OLED display for visual data representation, and a Zigbee sniffer tool interfacing with a PC for traffic analysis. On the software side, custom datasets are generated using Wireshark to simulate network intrusions, which are then pre-processed and analysed to prepare for training deep learning models. These models, utilizing a GRU

architecture, are trained to detect and classify attacks based on the collected data. The honeypot framework, implemented using Django, consists of two key modules: an admin interface for detailed traffic analysis presented in tables, and a user interface displaying real-time sensor data from the DHT11. The honeypot captures and records network traffic, presenting this data on a landing webpage designed to analyse potential attackers. Tools such as hping from Kali Linux are employed to simulate attacks, triggering scripts that gather attacker details. Behind the scenes, machine learning models continuously analyse captured data, automatically identifying and reporting potential threats within the web application. This integrated approach ensures proactive cybersecurity measures within ZigBee IoT networks, leveraging both hardware and software components to enhance security monitoring and threat detection capabilities.

6 Result And Evaluation

6.1 Results

Figure (4): Sniffed data

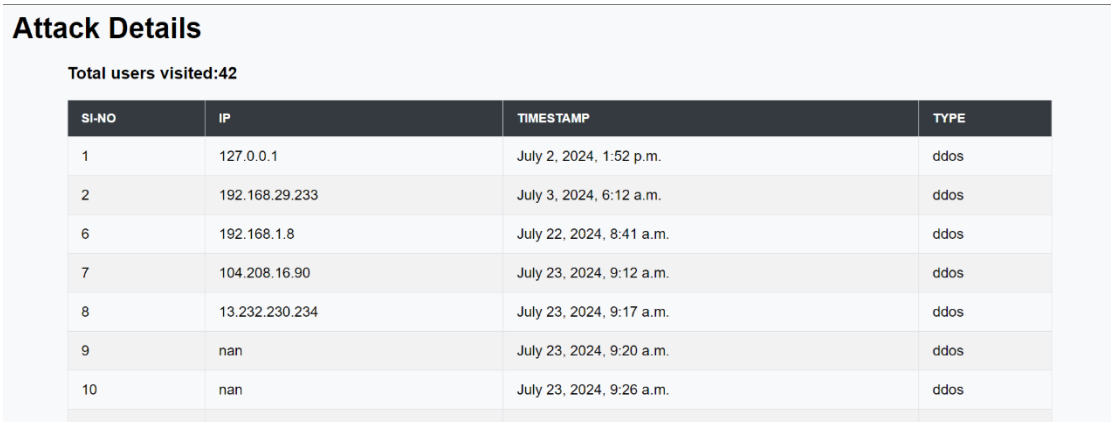
Figure (4) displays the data details captured by the Zigbee USB Dongle Plus CC2531 sniffer tool. It effectively captures and matches frequencies, demonstrating its capability to monitor and analyse ZigBee network traffic with accuracy. This tool plays a important role in the project by gathering valuable insights from frequency data, which are essential for project success.

```
Epoch 1/5
10/10 [==============================] - 6s 115ms/step - loss: 0.5707 - accuracy: 0.7745 - val_loss: 0.6971 - val_accuracy: 0.7657
Epoch 2/5
10/10 [==============================] - 0s 21ms/step - loss: 0.3396 - accuracy: 0.9003 - val_loss: 0.6461 - val_accuracy: 0.8347
Epoch 3/5
10/10 [==============================] - 0s 23ms/step - loss: 0.2611 - accuracy: 0.9218 - val_loss: 0.5989 - val_accuracy: 0.9121
Epoch 4/5
10/10 [==============================] - 0s 25ms/step - loss: 0.2265 - accuracy: 0.9398 - val_loss: 0.5601 - val_accuracy: 0.9351
Epoch 5/5
10/10 [==============================] - 0s 19ms/step - loss: 0.2144 - accuracy: 0.9506 - val_loss: 0.5336 - val_accuracy: 0.9456
```

Figure (5): Training epochs

Figure (5) shows the progress over 5 training epochs The model achieved 95% accuracy in training and 94% in validation, showing it learned well from the data. The training loss, which indicates how much predictions deviate from actual values during training, was 0.2144. Validation loss, which measures performance on new data, stood

at 0.5336. These results demonstrate the model's ability to effectively identify intrusion attempts in real-time in ZigBee IoT settings.

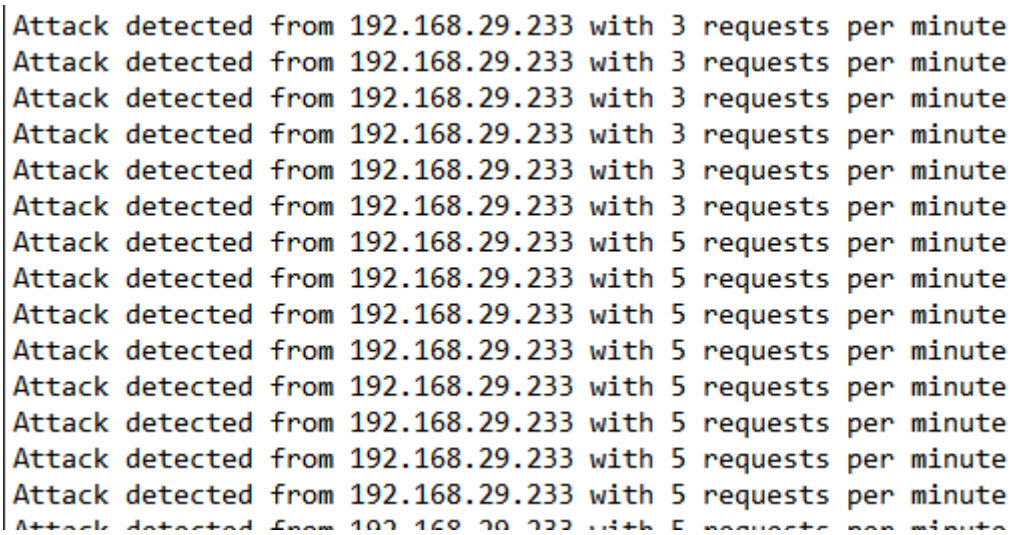


Attack Details

Total users visited:42

SI-NO	IP	TIMESTAMP	TYPE
1	127.0.0.1	July 2, 2024, 1:52 p.m.	ddos
2	192.168.29.233	July 3, 2024, 6:12 a.m.	ddos
6	192.168.1.8	July 22, 2024, 8:41 a.m.	ddos
7	104.208.16.90	July 23, 2024, 9:12 a.m.	ddos
8	13.232.230.234	July 23, 2024, 9:17 a.m.	ddos
9	nan	July 23, 2024, 9:20 a.m.	ddos
10	nan	July 23, 2024, 9:26 a.m.	ddos

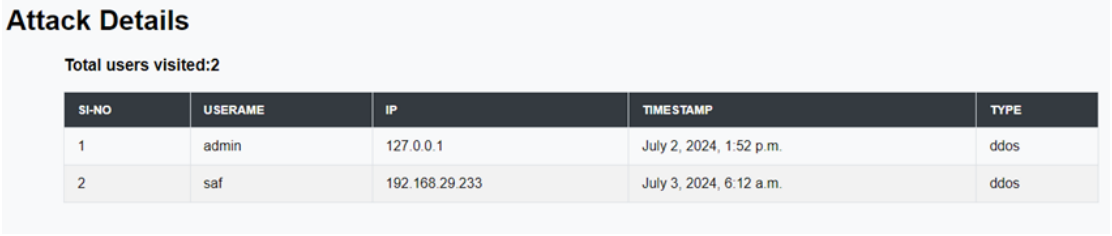
Figure (6): Detection logs on website



```
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 3 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
Attack detected from 192.168.29.233 with 5 requests per minute
```

Figure (7): Detection logs

Figure (6) shows the detected logs, indicating the IP addresses from which attacks originate. This visual representation helps to understand where potential threats are coming from and provides valuable insights into their types and frequency. Identifying these IP addresses clarifies who is trying to breach the system's security measures, allowing for proactive responses to protect against such threats effectively.



Attack Details

Total users visited:2

SI-NO	USERAME	IP	TIMESTAMP	TYPE
1	admin	127.0.0.1	July 2, 2024, 1:52 p.m.	ddos
2	saf	192.168.29.233	July 3, 2024, 6:12 a.m.	ddos

Figure (8): Admin view of network attack traffic

Figure (7) displays the admin view of the traffic details page, presenting data in a clear table format. It includes key information such as IP addresses, type of the traffic, and timestamps. This setup, integrated into a Django web application, enables administrators to monitor network activities effortlessly. By reviewing the recorded data and timestamps, they can identify both routine traffic patterns and suspicious activities like potential DDoS attacks, which is essential for effective network security management.

6.2 Evaluation

The evaluation of the proposed system demonstrates its effectiveness in combining advanced honeypot technology with deep learning for real-time intrusion detection. The results indicate that the system successfully enhances cybersecurity by accurately detecting evolving cyber threats. The objectives set for the study were all successfully achieved: Objective 1, involving setting up a microcontroller system for environmental data capture, was successfully accomplished. Objective 2 was also met, which aimed to set up ZigBee communication between XBee modules and ESP32 and included using a ZigBee sniffer tool to analyse traffic. Objective 3 aimed at training a deep learning model using GRU architecture for intrusion detection, and Objective 4 encompassed developing a functional honeypot to capture and store ZigBee traffic. Objective 5 integrated this with a web interface for real-time attack simulation and detection, while Objective 6 successfully integrated the trained deep learning model into the system to detect and classify attacks. This method uses honeypots along with advanced intrusion detection to trick attackers and quickly analyse and respond to their actions, making ZigBee IoT environments more secure.

The research question about “how well the security system combines honeypots with deep learning to trick and spot attacks in IoT settings?” has been answered. As seen in Figure (8), the system detects attacks accurately. This visual evidence demonstrates its ability to quickly recognize potential threats within IoT networks, showcasing its effectiveness in enhancing cybersecurity. One area where this study falls short is in the detection of various types of attacks by the deep learning models. While the system effectively spots some threats, it doesn't catch all possible attack scenarios.

7 Conclusion and future enhancements

The study developed an advanced framework of honeypots designed for ZigBee IoT environments, aimed at effectively deceiving potential attackers. The hardware setup included connecting sensors like the DHT11 and XBee modules to the ESP32 microcontroller, alongside an OLED display and a Zigbee sniffer tool linked to a PC. In the software realm, custom data was generated from Wireshark for intrusion detection. The project focused on training a deep learning model using GRU architecture on this dataset, thereby improving real-time intrusion detection capabilities. The deep learning model achieved a validation accuracy of 94%, highlighting its effectiveness in identifying potential threats. A honeypot was created to capture ZigBee traffic, storing data packets for analysis on a dedicated landing webpage. The honeypot's effectiveness was tested against simulated attacks using tools like hping on Kali Linux.

Future enhancements could include transitioning the system from local deployment to cloud deployment for enhanced scalability, ensuring it can handle increased data loads and expand its reach effectively. Also, a future enhancement in the model should focus on enhancing its detection capabilities to encompass a broader spectrum of security threats in IoT environments.

References

Abdallah, E.E., Eleisah, W. and Otoom, A.F. (2022) 'Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey', *Procedia Computer Science*, 201, pp. 205–212. Available at: https://doi.org/10.1016/j.procs.2022.03.029.

Ahmed, A. (2023) 'Introduction to Zigbee Technology'. Available at: https://www.researchgate.net/publication/370208966_Introduction_to_Zigbee_Technology

Alqahtani, H. *et al*. (2020) 'Cyber Intrusion Detection Using Machine Learning Classification Techniques', in N. Chaubey, S. Parikh, and K. Amin (eds) *Computing Science, Communication and Security*. Singapore: Springer, pp. 121–131. Available at: https://doi.org/10.1007/978-981-15-6648-6_10.

Amal, M.R. and Venkadesh, P. (2022) 'Review of Cyber Attack Detection: Honeypot System', *Webology*, Volume 19(No. 1), pp. 5497–5514. Available at: https://doi.org/10.14704/WEB/V19I1/WEB19370.

Ashiku, L. and Dagli, C. (2021) 'Network Intrusion Detection System using Deep Learning', *Procedia Computer Science*, 185, pp. 239–247. Available at: https://doi.org/10.1016/j.procs.2021.05.025.

Aziz, A. and Siddiqi, M. (2021) *Network Intrusion Detection Techniques using Machine Learning*. Available at: https://doi.org/10.13140/RG.2.2.23174.50248.

Azyus, A.F., Wijaya, S.K. and Naved, M. (2023) 'Prediction of remaining useful life using the CNN-GRU network: A study on maintenance management', *Software Impacts*, 17. Available at: https://doi.org/10.1016/j.simpa.2023.100535.

Baich, M. *et al*. (2022) 'Machine Learning for IoT based networks intrusion detection: a comparative study', *Procedia Computer Science*, 215, pp. 742–751. Available at: https://doi.org/10.1016/j.procs.2022.12.076.

Digi XBee® 802.15.4. Available at: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-802-15-4 (Accessed: 10 July 2024).

Hercog, D. *et al*. (2023) 'Design and Implementation of ESP32-Based IoT Devices', *Sensors,* 23(15), p. 6739. Available at: https://doi.org/10.3390/s23156739.

'Honeypot As Service' *CyberSRC*. Available at: https://cybersrcc.uk/cybersrc-honeypot-service/ (Accessed: 29 June 2024).

Hübschmann, I. (2020) ‘ESP32 for IoT: A Complete Guide’, *Nabto*, 28 August. Available at: https://www.nabto.com/guide-to-iot-esp-32/ (Accessed: 14 June 2024).

Islam, M.R. *et al*. (2023) ‘Machine learning enabled IoT system for soil nutrients monitoring and crop recommendation’, *Journal of Agriculture and Food Research*, 14, p. 100880. Available at: https://doi.org/10.1016/j.jafr.2023.100880.

Jain, G. and Anubha (2021) ‘Application of SNORT and Wireshark in Network Traffic Analysis’, *IOP Conference Series: Materials Science and Engineering*, 1119(1), p. 012007. Available at: https://doi.org/10.1088/1757-899X/1119/1/012007.

Javadpour, A., Ja’fari, F., Taleb, T., Shojafar, M. and Benzaïd, C. *A comprehensive survey on cyber deception techniques to improve honeypot performance - ScienceDirect*. Available at: https://www.sciencedirect.com/science/article/pii/S0167404824000932?via%3Dihub (Accessed: 18 July 2024).

Kok, S. *et al*. (2019) ‘A review of intrusion detection system using machine learning approach’, *International Journal of Engineering Research and Technology*, 12, pp. 8–15. Available at: https://www.researchgate.net/publication/332260496_A_review_of_intrusion_detection_system_using_machine_learning_approach

Mebawondu, J. Olamantanmi *et al*. (2020) ‘Network intrusion detection system using supervised learning paradigm’, *Scientific African*, 9, p. e00497. Available at: https://doi.org/10.1016/j.sciaf.2020.e00497.

Olutayo, V. (2022) ‘Analysis of Digital Forensics in the Implementation of Intrusion Detection using Snort’, 07, pp. 100–107. Available at: https://www.researchgate.net/publication/360808614_Analysis_of_Digital_Forensics_in_the_Implementation_of_Intrusion_Detection_using_Snort

Ozkan-Okay, M. *et al*. (2021) ‘A Comprehensive Systematic Literature Review on Intrusion Detection Systems’, *IEEE Access*, 9, pp. 157727–157760. Available at: https://doi.org/10.1109/ACCESS.2021.3129336.

Patgiri, R. *et al*. (2018) ‘An Investigation on Intrusion Detection System Using Machine Learning’, in 2018 *IEEE Symposium Series on Computational Intelligence (SSCI). 2018 IEEE Symposium Series on Computational Intelligence (SSCI*), pp. 1684–1691. Available at: https://doi.org/10.1109/SSCI.2018.8628676.

Prachi, Malhotra, H. and Sharma, P. (no date) ‘Intrusion Detection using Machine Learning and Feature Selection’, *International Journal of Computer Network and Information Security*, 11(4), p. 43. Available at: https://doi.org/10.5815/ijcnis.2019.04.06.

Raharjo, D.H.K. and Salman, M. (2023) ‘ANALYZING SURICATA ALERT DETECTION PERFORMANCE ISSUES BASED ON ACTIVE INDICATOR OF COMPROMISE RULES’, *Jurnal Teknik Informatika (Jutif),* 4(3), pp. 601–610. Available at: https://doi.org/10.52436/1.jutif.2023.4.3.1013.

Rama Devi, R. and Abualkibash, M. (2019) ‘Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - A Review Paper’, *International Journal of Computer Science and Information Technology*, 11(03), pp. 65–80. Available at: https://doi.org/10.5121/ijcsit.2019.11306.

Roy, S. *et al*. (2019) ‘Pre-Processing: A Data Preparation Step’, in S. Ranganathan *et al*. (eds) *Encyclopedia of Bioinformatics and Computational Biology*. Oxford: Academic Press, pp. 463–471. Available at: https://doi.org/10.1016/B978-0-12-809633-8.20457-3.

Santosh, K., Das, N. and Ghosh, S. (2022) ‘Chapter 3 - Deep learning models’, in K. Santosh, N. Das, and S. Ghosh (eds) *Deep Learning Models for Medical Imaging*. Academic Press (Primers in Biomedical Imaging Devices and Systems), pp. 65–97. Available at: https://doi.org/10.1016/B978-0-12-823504-1.00013-1.

Titarmare, N., Hargule, N. and Gupta, A. (2019*). International Journal of Computer Sciences and Engineering ijcseonline.org*. Available at: http://www.ijcseonline.org/ (Accessed: 15 July 2024).

Wang, H., Wei, Q. and Xie, Y. (2022) ‘A Novel Method for Network Intrusion Detection’, *Scientific Programming*, 2022(1), p. 1357182. Available at: https://doi.org/10.1155/2022/1357182.

Wanjau, S.K., Wambugu, G.M. and Oirere, A.M. (2022). Network Intrusion Detection Systems: A Systematic Literature Review o f Hybrid Deep Learning Approaches. *International Journal of Emerging Science and Engineering (IJESE).* Available at: https://www.ijese.org/portfolio-item/f25300510622/ (Accessed: 9 June 2024).

Zielinski, D. and Kholidy, H.A. (2022) ‘An Analysis of Honeypots and their Impact as a Cyber Deception Tactic’. arXiv. Available at: https://doi.org/10.48550/arXiv.2301.00045.

Ztt1, F. *et al*. (2022) ‘DHT11 Based Temperature and Humidity Measuring System’, *Journal of Electrical Engineering and Electronic Technology*, 2022. Available at: https://www.scitechnol.com/abstract/dht11-based-temperature-andrnhumidity-measuring-system-20039.html (Accessed: 19 June 2024).