ML- Based IoT Threat Detection

Nirmit Patel, Pratik Paul, Purushothama Rajanna
Syracuse University, npatel13@syr.edu, ppaul01@syr.edu, prajanna@syr.edu

Abstract - **In the rapidly evolving landscape of the Internet of Things (IoT), where connected devices permeate various facets of our lives, ensuring robust security and privacy measures is paramount. This project addresses the escalating challenges posed by the proliferation of IoT devices by adopting a multifaceted research approach. The primary objective is to enhance IoT security and privacy through an in-depth understanding of network behaviors under diverse attack scenarios. Leveraging advanced network simulation tools, including Cisco Packet Tracer and NetSim, we construct a highly realistic and scalable IoT environment to emulate device behaviors, network topologies, and communication protocols. A spectrum of attack scenarios, including Distributed Dinal of Service (DDoS), Mirai, SQL Injection, Command Injection and BHijacking, is orchestrated to comprehensively evaluate IoT security. Programmatic endpoints within the simulated environment facilitate extensive data collection, encompassing network traffic, device interactions, timestamps, and behavioral patterns during normal operations and attacks. Rigorous data cleaning and preparation, including deduplication and noise reduction, precede the application of diverse machine learning algorithms, like Logistic Regression, Random Forrest Classification and XGBoost, and deep learning techniques like Neural Networks. These algorithms scrutinize preprocessed data to identify patterns and anomalies associated with attacks. By evaluating the efficacy of various machine learning algorithms and deep learning algorithms, the project aims to contribute not only valuable insights into the security and privacy of IoT networks but also proactive, data-driven defense mechanisms. The significance of this endeavor lies in its potential to uncover hidden vulnerabilities, fortify network defenses, and mitigate the evolving threat landscape, ensuring the integrity and confidentiality of IoT data and services.**

Index Terms – Attack Scenario, Deep Learning Techniques, Machine Learning Algorithms, Network Simulation.

Introduction

In today's digital world, the widespread use of Internet of Things (IoT) devices has changed how we engage with our environment. Whether it is smart homes, city living, or systems in healthcare and industry, IoT has become a crucial part of our daily lives. [1] However, the quick increase in connected devices has brought about new challenges in terms of security and privacy.

This project is born out of a dedication to tackle the complex issues arising from the use of IoT devices. Our main goal is to strengthen the security and privacy of IoT setups through a detailed research approach. By thoroughly studying how IoT networks behave under different attack situations, we want to use machine learning and deep learning to not just anticipate, detect, and respond to threats effectively but also to contribute to a broader understanding of securing IoT environments.

As the world of IoT expands with various devices, sensors, and communication methods, it's crucial to thoroughly explore this complex system. With IoT devices becoming a part of critical infrastructures and personal spaces, potential security breaches go beyond just compromising data; they can lead to disruptions in services and even physical harm. This urgency emphasizes the need to protect the integrity, availability, and confidentiality of IoT data and services.

This project's importance goes beyond academic boundaries, directly affecting our daily lives and the security of essential infrastructures. The widespread use of IoT devices in sectors like healthcare, transportation, agriculture, and manufacturing increases the risks to finances and threats to public safety and privacy due to vulnerabilities in IoT networks. By carefully examining the security and privacy aspects of these networks, our aim is to contribute to the creation of strong defense mechanisms capable of adapting to the ever-changing threat landscape.

The following sections of this report will delve into the detailed methodology used, which includes advanced network simulations, various attack scenarios, thorough data collection and cleaning, and the use of a range of machine learning algorithms. Together, these methods aim not just to provide valuable insights into the security and privacy of IoT networks but also to pave the way for developing proactive, data-driven defense mechanisms that ensure the continuous functioning of IoT systems.

Problem Statement

[2] In today's world of distributed computing and widespread remote access to services through the Internet, ensuring the security of data transmission is a pressing concern. The emergence of different attacks and malicious codes poses a significant threat to the integrity of distributed computing platforms. Beyond the well-known distributed denial of service (DDoS) attacks, there's a range of malicious activities such as Mirai, SQL Injection, Command Injection and BHijacking. Consequently, the need for a robust mechanism to detect, analyze, and remove botnets is Important. Existing studies on threat detection techniques are varied but often limited in scope, lacking discussions on the latest

advancements. This study aims to fill this gap by developing advanced machine learning models and deep learning techniques for threat detection, incorporating the newest techniques, and analyzing both current and historical research trends. By providing a thematic taxonomy for classifying threat detection techniques, the study aims to shed light on their implications and critical elements in a way that is accessible and relevant to a broader audience.

Significance of Our Study

The popularity of Internet of Things (IoT) services and applications has surged owing to their functionality and user-friendly nature. Companies are actively creating a diverse array of IoT-based products, ranging from everyday personal gadgets like smartwatches to expansive networks like smart grids, smart mining, smart manufacturing, and autonomous driverless vehicles. However, the widespread availability and sheer quantity of IoT devices have attracted potential hackers seeking to exploit vulnerabilities for data theft and cyberattacks. Security emerges as a paramount concern within the realm of IoT. This study aims to address this concern by proposing an innovative machine learning algorithm-based model designed to detect and thwart threats on IoT networks.

Project Objectives and Methodology

I. Objectives

The objectives of the project are as follows:

- Simulate a complex network of IoT devices, introduce different types of network attacks and extract the raw network traffic data.
- Transform this network traffic data into data that we can use in our machine learning models using different preprocessing techniques.
- Develop different machine learning and deep learning modes to classify and predict different types of threat.

II. Methodology

The cornerstone of this project lies in acquiring high-quality data, as its absence would hinder the effective utilization of machine learning models. To procure this essential data, our strategy involves employing network simulation tools. After thorough deliberation, we opted for the NetSim tool due to its flexibility, user-friendly interface, and its capacity to seamlessly incorporate attacks into our simulated network.

First, [3]a survey was conducted to identify over 30 Wireless Sensor Network (WSN) simulators suitable for IoT research. We started by creating a complex network of IoT devices, after that we used NetSim's inbuilt tools to introduce attacks into the network. Here we used five different types of attacks- Distributed Denial of Service (DDoS), BHijacking, Command Injection, SQL Injection and Mirai. We simulated multiple instances of these attacks with multiple different factors and parameters so that we could get a large sample data. NetSim provides functionality to export the network traffic as a PCAP file that we can use with wireshark tool to do a primary network analysis, we took advantage of this feature and exported network traffic data for all the different types of attacks as PCAP files. With this attack data we required normal network traffic data without any attacks as a baseline. During our development our NetSim license got expired and the company did not provide us with an extension. So, to get the benign data we used Cisco Packet Tracer. We repeated the same steps that we used for NetSim to extract the benign data. After running all the simulations, we ended up with six different PCAP files, one for each type of attack and one for benign data.

Secondly, The data we have is in PCAP format, to use this data with machine learning models we converted combined this data into a single CSV file. We used this CSV file to perform various data cleaning techniques like removing NaN and duplicates values, encoding IP address, transforming all the columns into Int or Float dtypes and using get dummies for categorical variables. We used this cleaned data and performed various analytical tests to determine the final set of features that we are going to use for our final predictions.

For the last step of our project, we set up multiple machine learning and deep learning models that includes Logistic Regression, Random Forest Classification, Support Vector Machines, K-Nearest Neighbor Algorithm, XGBoost and Deep Learning Neural Network. This is a good spread models that gives us a good idea of which supervised learning algorithm performs the best. We split the data in training and testing data in 80-20 ratio. We used the training data to train all the machine learning models and used the testing data to predict the outputs.

This meticulous methodology ensures a robust evaluation of our machine learning models, validating their effectiveness in detecting and mitigating threats within IoT networks.

Data Preprocessing

I. Data Extraction

After running simulations for all the different types of attacks and benign traffic data we generated six different PCAP files. Since we cannot use the PCAP file for our model analysis we had to extract all necessary data and create a CSV file. We used this file to do all our data analysis. We used scapy library to conduct feature extraction from PCAP files, extracting a myriad of features from network packets. Each feature is meticulously selected or computed to offer a comprehensive understanding of the network activity or potential attacks. The 'Label' attribute, for instance, categorizes each packet into distinct types, such as 'BHijacking,' 'CommandInjection,' 'DDos,' 'Mirai,' 'SqlInjection,' or 'Benign,' drawing information from the predefined labels associated with each pcap file.

The 'Time' feature, derived from the timestamp of each packet at the IP layer, provides temporal insights into when each network event occurred. Moving to the 'Source IP' and 'Destination IP,' these attributes unveil the origin and destination of the network communication, gleaned from the IP layer of the packet. 'Source Port' and 'Destination Port' enter the scene if the packet contains TCP or UDP data, revealing the specific application or service linked with the network communication.

The 'Protocol' feature, obtained from the IP layer, clarifies whether the packet utilizes TCP, UDP, or another protocol. 'Length,' on the other hand, quantifies the packet size in terms of bytes. 'Source IP Bytes' and 'Destination IP Bytes' further enrich the dataset by calculating the byte length of the source and destination IP addresses using UTF-8 encoding.

The dynamic duo of 'Srate' (Source Rate) and 'Drate' (Destination Rate) then steps in to quantify the data rates for the source and destination, measured in bits per second. This calculation considers the size of the IP addresses and the time difference between consecutive packets, offering valuable insights into the data transfer rates of the network communication.

In the broader context, these features collectively form a robust dataset poised for analysis and subsequent application in machine learning for network security and IoT device monitoring.

II. Feature Engineering

The ensemble of custom transformers presented here serves as a key component in the realm of machine learning feature engineering. Each class, designed to seamlessly integrate with scikit-learn's pipeline and transformation tools, contributes to the augmentation of the dataset with refined features, enhancing the overall efficacy of machine learning models.
The IPAddressCategoryTransformer enriches the dataset with SourceIPCategory and DestIPCategory columns, categorizing IP addresses into private, public, multicast, or other types, with a distinct category for invalid IPs.

CommonPortFlagTransformer extends the dataset with SourcePortCommon and DestPortCommon columns, indicating whether SourcePort and DestPort correspond to common ports, such as 80 for HTTP or 443 for HTTPS.
The TemporalFeaturesTransformer provides temporal insights by extracting HourOfDay and DayOfWeek from the Time column, offering valuable information about the timing of network events.

TrafficFlowFeaturesTransformer focuses on network traffic flow, introducing BytesRatio and LengthToPortRatio features. BytesRatio signifies the ratio of bytes sent from source to destination, while LengthToPortRatio represents the ratio of the data packet length to the sum of source and destination ports.

The RateFeaturesTransformer emphasizes data transfer rates, incorporating CombinedRate and RateDifference columns. CombinedRate represents the sum of rates from source to destination and vice versa (Srate + Drate), while RateDifference captures the difference between the source and destination rates (Srate - Drate).

In the context of machine learning, each transformer significantly contributes to refining the dataset and introducing novel features, thereby enhancing model accuracy and overall performance. This seamless integration into the data transformation process ensures a robust foundation for subsequent machine learning applications in network security and IoT device monitoring.

III. Data Cleaning and Data Preparation

In the process of data cleaning, the code adeptly identifies columns containing missing values (NaNs) by utilizing the isna() function and filtering out columns with any NaN values using any(). Specifically, the columns 'SourcePort' and 'DstPort' emerge as the ones harboring missing data.

To address this, a strategic approach involves imputing the missing values in the 'SourcePort' and 'DstPort' columns with their respective mode values. The mode, representing the most frequently occurring value in a column, is employed as a robust choice for imputing categorical data, such as port numbers. This imputation technique, facilitated by the fillna() function with inplace=True, ensures that the most common values seamlessly replace the missing entries, enhancing the completeness of the dataset.

Beyond NaN values, the broader scope of data cleaning encompasses the removal of duplicate and repetitive values. By undertaking these measures, the dataset undergoes comprehensive refinement, ensuring its integrity and paving the way for more robust analyses and model building in subsequent stages of the project.

In the realm of data preparation, several transformative steps are undertaken to enhance the dataset's utility and compatibility with machine learning algorithms. Commencing with the 'Time' column, a pivotal temporal element, a conversion to a datetime object using pd.to_datetime unfolds. Subsequently, the code extends its scope by extracting diverse time components, including the year, month, day, hour, minute, and second, thus enriching the dataset with a nuanced temporal context.

Beyond the basic temporal features, the code delves deeper, extracting additional temporal attributes. The day of the week and week of the year are discerned, providing valuable insights into cyclic patterns. Furthermore, a determination of whether the time corresponds to a weekend occurrence is made. Notably, the code calculates a 'time_of_day' feature, encapsulating the temporal information in a decimal format, and creates a Unix timestamp, contributing to a more comprehensive temporal representation.

Shifting focus to the encoding of IP addresses, the code introduces a function, 'ip_to_int,' facilitating the transformation of 'SourceIP' and 'DestIP' columns into their numerical counterparts. Subsequently, the 'Time,' 'SourceIP,' and 'DestIP' columns are judiciously dropped, marking the conclusion of the transformation process, as their information has been effectively assimilated into more meaningful features.

Transitioning to the dataset division, the StratifiedShuffleSplit technique is employed to segregate the dataset into training and test sets. This technique ensures that both sets maintain a representative distribution of classes, a crucial consideration in scenarios where class imbalances exist. By preserving class proportions, the stratification process fortifies the reliability of subsequent evaluation metrics, especially significant for models susceptible to overfitting on majority classes and underperforming on minority ones.

In the realm of feature scaling, both standardization and normalization techniques are employed to align the dataset for optimal model performance. The StandardScaler transforms attribute distributions to a standard normal distribution with a mean of 0 and a standard deviation of 1, catering to algorithms assuming Gaussian data. Simultaneously, the MinMaxScaler ensures a fixed value range, typically between 0 and 1, beneficial for algorithms reliant on scaled inputs, such as neural networks and k-nearest neighbors (k-NN).

Lastly, the Class Weight Calculation technique is harnessed to address imbalances in class representation during model training. By assigning higher weights to underrepresented classes, the model is incentivized to pay more attention to minority classes, preventing biases towards the majority class. This nuanced approach is crucial for handling imbalanced datasets, fostering a more equitable learning process for machine learning models.

Data Analysis

The visualizations provided encompass a series of insightful plots, presumably derived from a comprehensive data analysis of network traffic. [4] There are potential benefits of analyzing the data collected from Internet of Things (IoT) devices. Each plot offers distinctive conclusions:

- **Protocol Distribution Bar Chart:** This chart vividly highlights the prevalence of one protocol (possibly labeled "6") with a significantly higher count than others. This dominance suggests that this protocol is the most frequently encountered or utilized in the dataset, overshadowing the counts of the remaining protocols ("17" and "1").

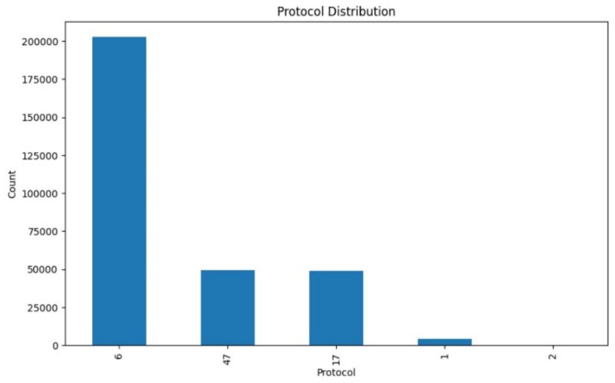

FIGURE I
Protocol Distribution Bar Chart

- **Traffic Type Distribution Pie Chart:** The pie chart provides a nuanced perspective on the distribution of various traffic types. Categories include "Benign," "DDos," "Mirai," "SqlInjection," "CommandInjection," and "Hijacking." "Benign" traffic occupies the largest slice, closely followed by "Hijacking" and "DDos." This composition implies a dataset exhibiting a blend of normal and malicious traffic, with benign traffic prevailing but accompanied by notable instances of diverse attack types.

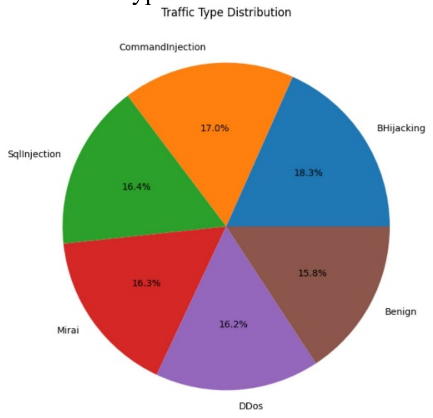

FIGURE II
Traffic Type Distribution Pie Chart

- **Source vs Destination IP Bytes Scatter Plot**: In this scatter plot comparing source and destination IP bytes across different traffic types, each type is represented by distinct colors. While no distinct patterns emerge solely based on byte count, the plot reveals a diverse distribution of source and destination bytes for various traffic types.

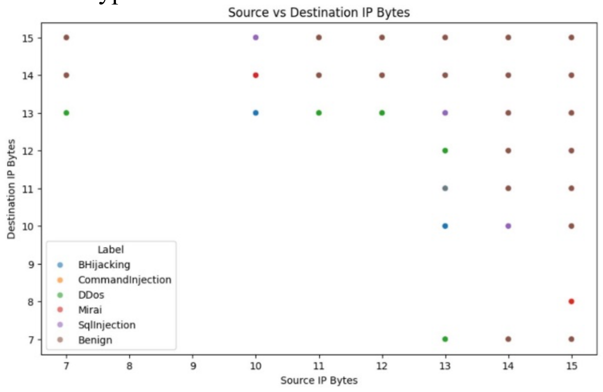

FIGURE III
Source vs Destination IP Bytes Scatter Plot

- **Packet Length Distribution Box Plot**: The box plot sheds light on the distribution of packet lengths for different traffic types. Notably, "Benign" traffic tends to exhibit shorter packet lengths, while "DDos" traffic showcases a broader range with potential outliers indicating unusually long packets. Other categories also exhibit variations in packet length, with potential outliers signifying abnormal sizes possibly indicative of malicious activity.

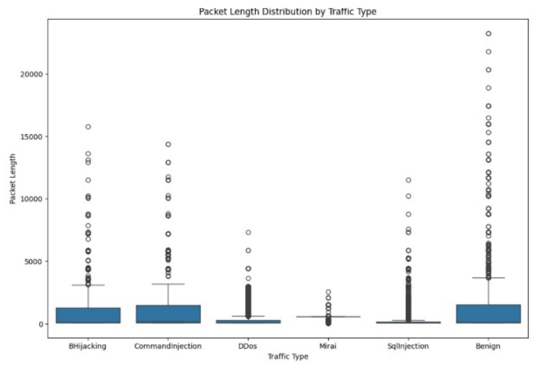

FIGURE IV
Packet Length Distribution Box Plot

- **Feature Correlation Matrix Heatmap**: The heatmap delves into the correlation between various features encompassing packet lengths, port numbers, byte sizes, and other technical aspects of network traffic. High positive or negative correlation values indicate strong relationships, crucial for feature selection in machine learning models. Understanding these relationships aids in discerning network behavior and holds potential for developing predictive models in the realm of cybersecurity.

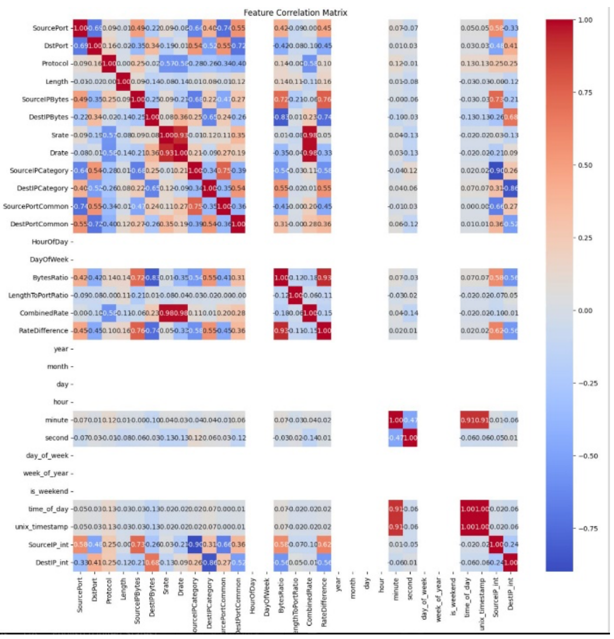

FIGURE V
Feature Correlation Matrix Heatmap

- Overall Conclusion: These visualizations stem from a meticulous network traffic analysis, aiming to discern and comprehend the characteristics of diverse traffic types, ranging from benign to potentially malicious activities. The protocol distribution highlights a dominant protocol, the traffic type distribution portrays a mixture of benign and malicious activities, the scatter plot offers insights into IP bytes without a clear distinction based on byte count, and the packet length distribution unveils the nature of traffic types. The correlation matrix further contributes to the understanding of feature relationships, valuable for both diagnosing network behavior and advancing predictive models in the realm of cybersecurity.

Model Analysis

[5] In the pursuit of constructing a robust and effective model for the cybersecurity domain, two distinct categories are explored: [6] Machine Learning Models and Deep Learning Models.

I. Machine Learning Models

- **Random Forest:** The RandomForestClassifier is employed and meticulously tuned for optimal performance. Initially, without class weights, the model is fine-tuned using RandomizedSearchCV, exploring parameters such as the number of estimators, maximum depth, and minimum samples split. Subsequently, class weights are incorporated into the model by creating a pipeline with SMOTE for oversampling. Again, RandomizedSearchCV is employed to fine-tune hyperparameters. Both scenarios are evaluated on the test set using metrics like accuracy and F1 score.
- **Logistic Regression:** Logistic Regression, a reliable choice for multi-class classification, is implemented with the 'lbfgs' solver and a maximum iteration setting of 1000. After training on the scaled training set, predictions are made on the scaled test set. Performance metrics including accuracy, precision, recall, and F1 score are calculated and presented in a comprehensive classification report.
- **XGBoost:** XGBoost, renowned for its efficiency in classification tasks, is leveraged for identifying cyberattacks. Labels are encoded, and the dataset is split into training and testing sets. The classifier is initialized with 'multi:softprob' for multiclass classification. Post-training, predictions are made on the test set, and the model's performance is evaluated using a classification report and accuracy score.
- **Support Vector Machine (SVM):** SVM, known for its effectiveness in high-dimensional spaces, is implemented with a linear kernel for classifying different types of network traffic. The process involves encoding labels, fitting the SVM model on scaled training data, predicting labels on the test data, and calculating performance metrics. A detailed classification report offers insights into the SVM model's efficacy.
- **Ensemble (Random Forest and XGBoost):** An ensemble of RandomForest and XGBoost models is explored to harness the collective predictive power. Both classifiers are trained on a labeled dataset, and the ensemble prediction is created by averaging probabilities. The final prediction is derived from the class with the highest average probability. Ensemble

performance is assessed through accuracy and a detailed classification report.

II. Deep Learning Models

- **Neural Network:** Neural Networks, revered for their ability to capture complex patterns and non-linear relationships, are embraced for classifying network traffic and attacks. The TensorFlow library is employed to construct a Neural Network model with Dense layers, ReLU activation, and a softmax output layer. L1 and L2 regularization, along with dropout, are incorporated for preventing overfitting. The model is compiled with the Adam optimizer and categorical crossentropy loss function, subsequently trained with validation data to monitor performance. The evaluation involves assessing the model's predictions against the test set, presenting metrics such as accuracy and a classification report to gauge its effectiveness.

III. Result

[7] The evaluation results shed light on the performance of the various models deployed in our cybersecurity project. Each model underwent meticulous tuning and training to comprehend and classify different types of network traffic, ranging from benign to potentially malicious activities.

- **Random Forest:** With an accuracy of 32.2%, the Random Forest model demonstrated a modest capability in correctly classifying instances. However, precision, recall, and F-1 score values of 0.12, 0.33, and 0.17, respectively, suggest challenges in achieving a balanced trade-off between true positives, false positives, and false negatives.
- **Logistic Regression:** The Logistic Regression model exhibited a similar level of accuracy at 30.7%, with precision, recall, and F-1 score values of 0.12, 0.30, and 0.17, mirroring a comparable performance to the Random Forest model.
- **XGBoost:** Surprisingly, the XGBoost model outperformed the others with a perfect accuracy of 100%. While achieving impeccable precision, recall, and F-1 scores of 1.0 across all categories, the results warrant a closer examination to ensure the absence of overfitting or data leakage.
- **SVM:** The Support Vector Machine achieved an accuracy of 32.7%, aligning closely with the Random Forest and Logistic Regression models. The precision, recall, and F-1 score values of 0.17, 0.32, and 0.20 indicate a comparable performance level.
- **Ensemble (Random Forest and XGBoost):** The ensemble model, combining the strengths of both Random Forest and XGBoost, demonstrated an accuracy of 32%. The precision, recall, and F-1 score values mirrored those of the individual models, suggesting that the ensemble approach might not have significantly enhanced performance in this context.
- **Neural Network:** The Neural Network model emerged as a strong contender, boasting an accuracy of 85%. Precision, recall, and F-1 score values of 0.92, 0.87, and 0.84 underline its proficiency in correctly classifying instances, especially when compared to the traditional machine learning models.

In conclusion, while traditional machine learning models such as Random Forest, Logistic Regression, and SVM showcased comparable performances, the standout performer was the Neural Network, showcasing superior accuracy and balanced precision, recall, and F-1 scores. The unexpected perfection from XGBoost warrants further investigation to ensure the model's robustness. The ensemble model, while not yielding a significant improvement, underscores the nuanced nature of combining different model architectures. The detailed evaluation results provide valuable insights for refining and enhancing our approach in future iterations of this cybersecurity project.

TABLE I
Result Table

	Accuracy	Precision	Recall	F-1 Score
Random Forest	32.0%	0.12	0.33	0.17
Logistic Regression	30.7%	0.12	0.30	0.17
XGBboost	100%	1.00	1.00	1.00
SVM	32.7%	0.17	0.32	0.20
Ensemble	32.0%	0.12	0.33	0.17
Neural Network	85.5%	0.92	0.87	0.84

Conclusion and Future Work

In this IoT: Security and Privacy project, our aim was to develop an effective system for detecting threats in IoT networks, considering the rising vulnerability of these networks to cyberattacks. The widespread adoption of IoT devices in various sectors necessitates robust security measures. We selected NetSim as our network simulation tool for its flexibility, user-friendly interface, and attack simulation capabilities. Despite a sudden NetSim license expiration, we swiftly transitioned to Cisco Packet Tracer for benign data extraction.

Simulating attacks like DDoS, BHijacking, Command Injection, SQL Injection, and Mirai, we generated a diverse dataset. The data, presented in PCAP format, underwent thorough preparation for machine learning analysis. Using Logistic Regression, Random Forest Classification, SVM, XGBoost, ensemble and a Neural Network, we rigorously tested and evaluated our model against various simulated attacks.

The analysis revealed varying performances, with the Neural Network emerging as the top performer, showcasing its ability to handle complex patterns and non-linear relationships. Acknowledging the strengths and limitations of each model, our project provides a comprehensive view of their applicability in IoT network security.

Looking ahead, future work could involve refining model architecture, exploring ensemble strategies, and incorporating additional features. Real-time implementation and collaboration with cybersecurity experts may further enhance the model's effectiveness.

In conclusion, this project marks a significant step in fortifying IoT network security through innovative machine learning approaches. Insights gained contribute to advancing threat detection models, fostering proactive cybersecurity measures in the interconnected digital world.

References

[1] H. Suo, J. Wan, C. Zou and J. Liu, "Security in the Internet of Things: A Review," 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 2012, pp. 648-651, doi: 10.1109/ICCSEE.2012.373.

[2] HaddadPajouh, Hamed (06/01/2021). "A survey on internet of things security: Requirements, challenges, and solutions". Internet of Things (2542-6605), 14.

[3] M. Chernyshev, Z. Baig, O. Bello and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds," in IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1637-1647, June 2018, doi: 10.1109/JIOT.2017.2786639.

[4] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, Amit P. Sheth, Machine learning for internet of things data analysis: a survey, Digital Communications and Networks, Volume 4, Issue 3, 2018.

[5] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems," 2nd ed. O'Reilly Media, 2019.

[6] Khalid Alissa, Tahir Alyas, Kashif Zafar, Qaiser Abbas, Nadia Tabassum, Shadman Sakib, "Botnet Attack Detection in IoT Using Machine Learning", Computational Intelligence and Neuroscience, vol. 2022, Article ID 4515642, 14 pages, 2022.

[7] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A. A. Ghorbani. "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensor (2023) – (submitted to Journal of Sensors).