Module Title

Second Year Project

Assessment Weightage & Type
Second Year Interim Report (25%)

Year and Semester
2022/23 Autumn

Student Name: Aayush Wanem Limbu
College ID: np05cp4a220010
London Met ID: 22072043

Assignment Due Date: 2023-11-01

Assignment Submission Date: 2023-11-01

Supervisor Name: Mr. Romi Khatri, Mr. Nikesh Regmi

Title: HIDS a SYP project

I confirm that I understand my coursework needs to be submitted online via My Second Teacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

Table of Contents

1. Introduction 1

1.1. Introduction to the Topic 2

1.2. Current Scenario 3

1.3. Project Scope 4

1.4. End User 5

1.5. Aims and Objectives 5

2. Project Background 7

2.1. Tools and technology Used 8

2.2. Project Development Methodology 14

2.3. Review of the Similar Project 15

2.3.1 snort: 15

2.3.2 Comparison beteen snort and therat guard 16

3. Project Development 16

3.1. Requirement Gathering 16

3.2. Use Case 17

3.3. SRS Document 18

3.4. Wireframes 18

3.5. ERD 18

4. Analysis of progress 18

4.1. Progress Table 19

4.2. Progress Timeline (Gantt Chart) 20

5. Future work 20

6. References 21

7. Appendix 21

7.1. SRS Document 21

1. Introduction.....21
1.1 Purpose.....21
1.2 Scope.....22
1.3 Definitions, Acronyms, and Abbreviations.....22
1.4 References.....22
2. Overall Description.....22
2.1 Product Perspective.....22
2.2 Product Features.....22
2.3 User Classes and Characteristics.....23
2.4 Operating Environment.....23
3. Specific Requirements.....23
3.1 Functional Requirements.....23
3.2 Non-Functional Requirements.....24
4. Appendix.....25
4.1 Use Case Diagram.....25
4.2 Entity Relationship Diagram.....26
4.3 Glossary.....26
7.2. Wireframes.....27
7.3. Entity Relationship Diagram.....27
7.4. Project as a solution.....31
7.5. Project Gantt Chart.....32
7.6. Selected Methodology.....33

Table of Figures

Figure 1: Vs Code Logo 8
Figure 2:Sublime Text 8
Figure 3: Jupyter Lab 9
Figure 4:Jupyter Notebook 9
Figure 5: GitHub Logo 10
Figure 6:HTML CSS JS Logo 10
Figure 7:Django Logo 11
Figure 8:SQLite 11
Figure 9: Draw.io Logo 12
Figure 10: Team Grantt Logo 12
Figure 11: Figma 12
Figure 12Python 13
Figure 13:Random Forest 13
Figure 14 Wireshark 14
Figure 15: Snort 15
Figure 16: Comparision between snort and ThreatGuard 16
Figure 17: Use Case Diagram 17
Figure 18: Initial ERD 18
Figure 19: Progress Table 19
Figure 20: Gantt Chart 20

1. Introduction

The rapid evolution of cyber threats has underscored the critical need for robust Intrusion Detection Systems (IDS) capable of detecting and mitigating network attacks. Traditional IDS face challenges in accurately identifying modern, sophisticated threats, prompting the exploration of advanced technologies such as machine learning. This interim report presents a study that aims to evaluate the effectiveness of machine learning algorithms, including decision trees, KNN, and the random forest classifier, in predicting network attacks and enhancing IDS accuracy.

The research focuses on leveraging the random forest algorithm from scikit-learn, a popular machine learning library in Python known for its efficiency and effectiveness in handling complex datasets. The study utilizes the KDDCUP99 dataset, a benchmark for training and testing IDS, to analyze and classify diverse intrusion patterns. By enhancing the accuracy and efficiency of intrusion detection, this research seeks to bolster network security in the face of evolving cyber threats. This project not only aims to improve the detection capabilities of IDS but also to provide insights into the practical application of machine learning in enhancing cybersecurity measures."

This expanded version provides more detail about the specific algorithm (random forest) and dataset (KDDCUP99) you used, as well as the potential implications of your research for enhancing network security.

1.1. Introduction to the Topic

"Treat Guard" stands at the forefront of cybersecurity as a comprehensive Django web application tailored for administrators overseeing host device security. This innovative platform seamlessly integrates with a Host-based Intrusion Detection System (HIDS), enabling real-time email alerts to administrators in response to potential cyber threats. Leveraging Django's powerful framework, Treat Guard offers a sophisticated user interface, empowering administrators to monitor, manage, and administer host devices with ease.

Treat Guard's advanced features include detailed device profiles, allowing administrators to view and modify device settings, network configurations, and security protocols. The application's integration with the HIDS provides administrators with critical insights into network activity, alerting them to potential threats such as unauthorized access attempts, malware activity, or suspicious network traffic. Administrators can take immediate action, such as isolating devices or blocking IP addresses, to prevent further security breaches.

Additionally, Treat Guard's email alert system ensures that administrators are promptly informed of any security incidents, enabling them to respond swiftly and effectively. This proactive approach to cybersecurity not only enhances the overall security posture of the organization but also ensures business continuity by minimizing the impact of potential cyber-attacks. Treat Guard sets a new standard in host device security, offering administrators a powerful tool to protect their network infrastructure and data assets.

1.2. Current Scenario

Intrusion Detection System(IDS) and Machine Learning

In today's rapidly evolving digital landscape, the proliferation of network connectivity has brought about a corresponding increase in cybersecurity threats. Intrusion Detection Systems (IDS) play a pivotal role in safeguarding networks by monitoring and analyzing network traffic to detect and respond to malicious activities. However, traditional IDS face significant challenges in effectively identifying and mitigating modern, sophisticated attacks.

Traditional IDS, such as Snort and Suricata, typically rely on predefined rules or signatures to detect known threats. For example, Snort uses a signature-based approach to compare network traffic against a database of known attack patterns. Similarly, Suricata utilizes signature-based detection along with support for protocol analysis and anomaly-based detection to identify suspicious activity. However, these signature-based approaches have limitations when faced with unknown or evolving attack vectors. For instance, a zero-day exploit, which exploits a vulnerability that is unknown to the software vendor, may bypass signature-based detection, leading to a successful attack. Additionally, signature-based IDS can generate a high number of false positives if the signatures are too broad or if they match legitimate traffic patterns. Moreover, traditional IDS often struggle to distinguish between legitimate and malicious activity, leading to false positives. For example, legitimate activities such as network scans or software updates may trigger alerts in an IDS if they are not properly configured to differentiate between normal and suspicious behavior.

The increasing complexity and volume of network traffic pose scalability challenges for traditional IDS. As network traffic grows, IDS may struggle to keep pace with the volume of data, leading to performance issues and potential missed detections. To address these limitations, there has been a growing interest in leveraging machine learning algorithms for intrusion detection. Machine learning offers the potential to enhance IDS capabilities by enabling systems to learn from and adapt to new threats in real-time. By analyzing patterns and anomalies in network traffic data, machine learning algorithms can detect subtle signs of malicious activity that may go unnoticed by traditional IDS.

However, the deployment of machine learning-based IDS presents its own set of challenges. These include the need for large, labeled datasets for training, the complexity of algorithm selection and tuning, and the risk of adversarial attacks designed to deceive machine learning models.

Considering these challenges, there is a pressing need for research to evaluate the effectiveness of machine learning algorithms in enhancing IDS accuracy and efficiency. By exploring the capabilities of algorithms such as decision trees, KNN, and the random forest classifier, researchers seek to improve the detection capabilities of IDS and strengthen network security in the face of evolving cyber threats.

1.3. Project Scope

In addition to evaluating the effectiveness of the Random Forest Classifier (RFC) algorithm in enhancing the accuracy and efficiency of Intrusion Detection Systems (IDS), this research project will explore the feasibility of integrating a web interface for interacting and administering the IDS.

The web interface will provide a user-friendly way for administrators to monitor the IDS alerts, view network traffic patterns, and adjust IDS settings. This approach aims to enhance the usability and accessibility of the IDS, allowing administrators to manage network security more effectively.

The web interface will be designed to be intuitive and responsive, providing real-time updates and notifications for IDS alerts. It will also include features for data visualization, such as charts and graphs, to help administrators analyze network traffic patterns and identify potential security threats.

The feasibility of integrating a web interface will be evaluated based on factors such as technical complexity, resource requirements, and compatibility with existing IDS infrastructure. The goal is to demonstrate that the addition of a web interface can enhance the overall effectiveness and usability of the IDS, making it a more valuable tool for network security.

1.4. End User

The web interface of the Intrusion Detection System (IDS) is designed to meet the needs of IT officers, SOC analysts, network administrators, and system administrators, providing them with a comprehensive set of tools and features to effectively monitor and manage network security.

Key features of the web interface for end users include:

1. **Dashboard Overview**: A centralized dashboard provides a real-time overview of network activity, including alerts, traffic patterns, and system status, allowing users to quickly assess the security posture of the network.
2. **Alert Management**: Users can view and manage alerts generated by the IDS, including investigating alerts, acknowledging them, and taking appropriate actions to mitigate potential security threats.
3. **Configuration and Settings**: IT officers and administrators can customize IDS settings and configurations through the web interface, tailoring the system to meet specific security requirements and network conditions.
4. **Reporting and Analysis**: The web interface offers detailed reporting and analysis tools, allowing users to generate reports on network activity, alerts, and security incidents for further analysis and auditing purposes.
5. **Collaboration and Communication**: The web interface facilitates collaboration and communication among security team members, enabling them to share information, coordinate responses to security incidents, and work together efficiently.

Overall, the web interface enhances the user experience for IT officers, SOC analysts, and network and system administrators, providing them with the tools and insights they need to effectively monitor and manage network security.

1.5. Aims and Objectives

Aims:

The primary aim of this project is to develop a robust and efficient Intrusion Detection System (IDS) capable of identifying and mitigating various network attacks. The project aims to enhance cybersecurity measures by implementing advanced machine learning algorithms and integrating them into a comprehensive IDS.

Objectives:

1. Develop an Intrusion Detection System: Create a functional IDS that can monitor network activities, detect suspicious patterns, and identify potential intrusions.
2. Implement Machine Learning Algorithms: Utilize machine learning algorithms, specifically the Random Forest Classifier, to enhance the IDS's accuracy in identifying network attacks and distinguishing them from normal network traffic.
3. Integration of Frontend and Backend Technologies: Integrate frontend technologies (HTML, CSS, JavaScript) to create an intuitive user interface and backend technologies (Django) to handle data processing and algorithm implementation.
4. Utilize KDDCUP99 Dataset: Utilize the KDDCUP99 dataset, a benchmark dataset for intrusion detection, to train and test the machine learning algorithms, ensuring the system's effectiveness against a wide range of intrusion scenarios.
5. Implement Port Status Check and Control: Develop functionality to enable administrators to check the status of individual ports, including whether they are enabled or disabled, and to control ports by enabling or disabling them as needed.
6. Optimize System Performance: Optimize the system's performance by selecting appropriate hardware components, including processors, memory, and storage, ensuring seamless execution of algorithms and efficient data processing.
7. Implement Virtualization: Implement virtualization techniques to create a scalable and flexible environment, allowing for the deployment and testing of the IDS in various configurations.
8. User-Friendly Interface: Develop an intuitive user interface that allows administrators to visualize detected intrusions, generate reports, configure the IDS settings, and perform individual port status checks, enabling or disabling ports and checking the services running on ports.
9. Testing and Validation: Conduct rigorous testing, including unit testing, integration testing, and performance testing, to validate the IDS's accuracy, efficiency, and reliability under different network conditions and attack scenarios.

By achieving these objectives, the project aims to deliver an advanced and user-friendly Intrusion Detection System, contributing significantly to the realm of cybersecurity and network protection.

2. Project Background

Intrusion Detection Systems (IDS) are crucial components of network security, tasked with monitoring and analyzing network traffic to detect and respond to potential security threats. Traditional IDS face significant challenges, including high false positive rates and an inability to effectively detect unknown or evolving threats. These limitations highlight the need for more advanced and efficient IDS solutions.

Machine learning algorithms have emerged as promising tools for enhancing IDS capabilities. By analyzing patterns and anomalies in network traffic data, machine learning algorithms can improve detection accuracy and adaptability to new threats. The Random Forest Classifier (RFC) algorithm has shown effectiveness in distinguishing between normal network traffic and malicious activities.

This project aims to develop a robust and efficient IDS by integrating machine learning algorithms, specifically the RFC algorithm, into a comprehensive IDS framework. The project also includes the integration of a web interface to provide administrators with an intuitive tool for monitoring and managing network security.

The motivation behind this project stems from the increasing complexity and sophistication of cyber threats, which require more advanced and adaptive security measures. By developing an IDS that leverages machine learning and integrates a user-friendly web interface, this project seeks to address the limitations of traditional IDS and enhance network security in the face of evolving cyber threats.

2.1. Tools and technology Used

1. Integrated Development Environments (IDEs):

- Visual Studio Code (VS Code): VS Code is a popular code editor developed by Microsoft. It provides features such as syntax highlighting, code completion, and debugging support for various programming languages.

Figure 1: Vs Code Logo

- Sublime Text 4: Sublime Text is a lightweight, yet powerful text editor known for its speed and simplicity. It offers a wide range of plugins and customization options for efficient coding.

Figure 2:Sublime Text

2. Jupyter-lab and Jupyter-notebook:

- Jupyter-lab: JupyterLab is an interactive development environment for working with notebooks, code, and data. It provides a flexible and extensible interface for data science and machine learning tasks.

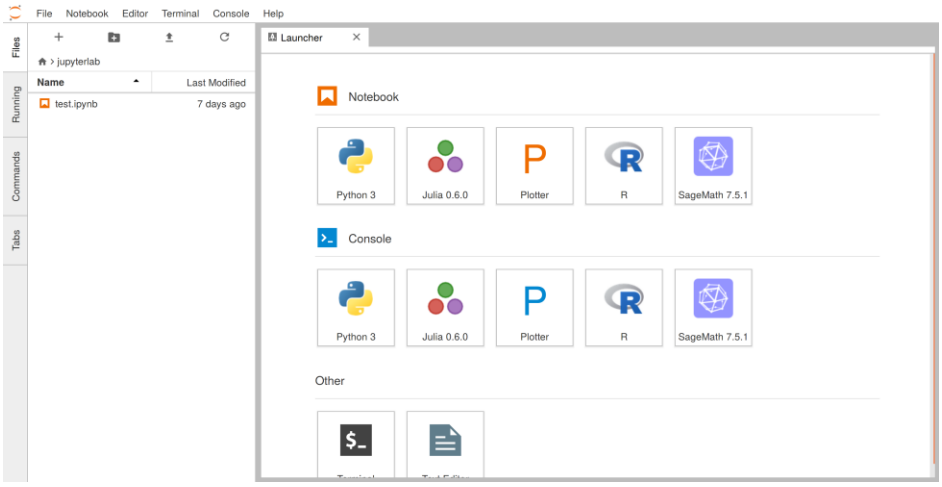


Figure 3: Jupyter Lab

- Jupyter-notebook: Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used for prototyping and experimenting with code.

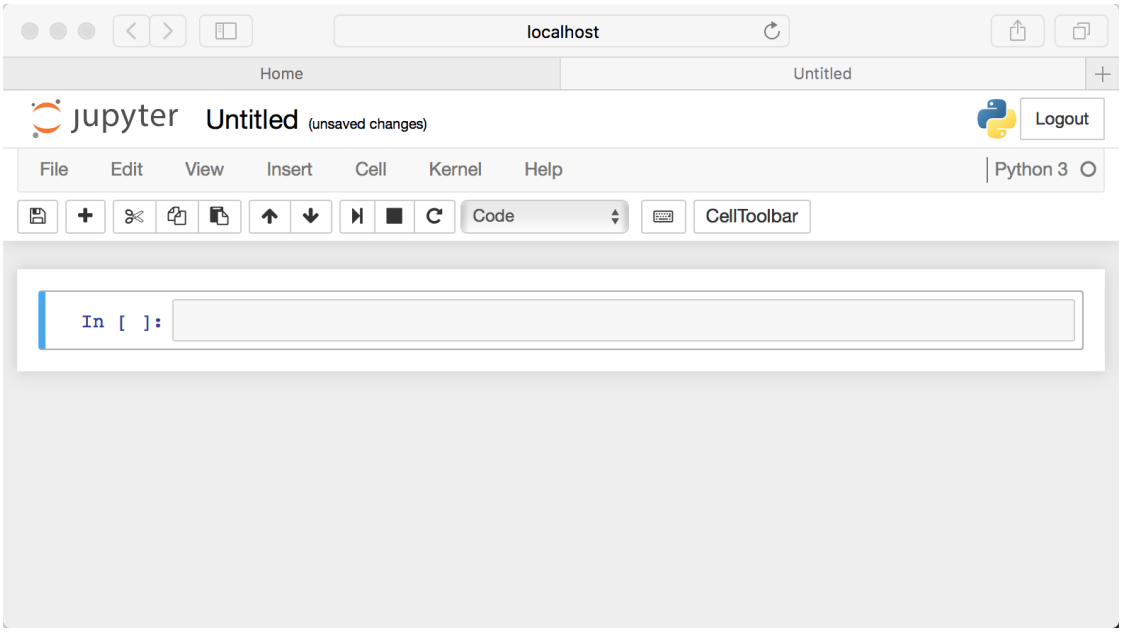


Figure 4:Jupyter Notebook

3. Version Control:

GitHub: GitHub is a web-based platform for version control and collaboration. It allows you to host and review code, manage projects, and build software alongside millions of other developers.

Figure 5: GitHub Logo

4. Frontend Languages:

HTML, CSS, and JavaScript: HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript are the core technologies used for building the frontend of web applications. HTML is used for creating the structure of web pages, CSS is used for styling and layout, and JavaScript is used for adding interactivity and dynamic behavior to web pages.

Figure 6:HTML CSS JS Logo

5. Backend Language and Framework:

Django: Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides built-in features for authentication, URL routing, templating, and database management, making it suitable for building complex web applications.

Figure 7:Django Logo

6. Database Management System:

SQLite3: SQLite is a lightweight, serverless, and self-contained SQL database engine. It is the default database engine used by Django for development and testing purposes.

Figure 8:SQLite

7. Diagramming Tools:

Draw.io: Draw.io is a free online diagramming tool that allows you to create flowcharts, UML diagrams, network diagrams, and more. It provides a user-friendly interface and a wide range of shapes and templates for diagram creation.

Figure 9: Draw.io Logo

8. Project Management Tools:

Team Gantt: TeamGantt is a project management tool that helps teams plan, organize, and track their projects using Gantt charts. It allows you to create and share project timelines, set dependencies, and allocate resources efficiently.

Figure 10: Team Grantt Logo

9. UI Design and Wireframing Tools:

Figma: Figma is a collaborative interface design tool that enables teams to create, prototype, and collaborate on designs in real-time. It provides a cloud-based platform for designing user interfaces, wireframes, and interactive prototypes.

Figure 11: Figma

10. Python and Machine Learning:

- Python: Python is a versatile programming language widely used for various purposes, including web development, data analysis, and machine learning. It offers a rich ecosystem of libraries and frameworks for developing and deploying machine learning models.

Figure 12Python

- Random Forest Classifier: Random Forest is a machine learning algorithm that uses an ensemble of decision trees to make predictions. It is commonly used for classification tasks and is known for its robustness and high accuracy.

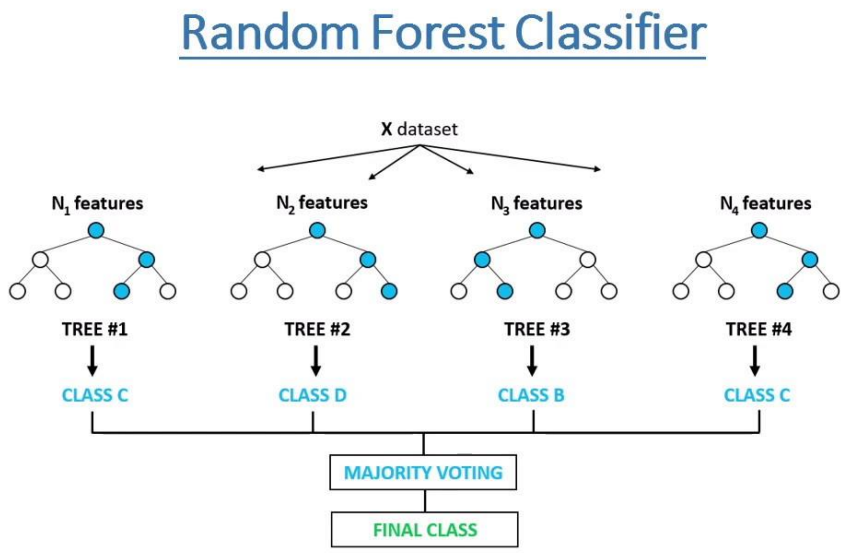


Figure 13:Random Forest

- kddcup99 Dataset: The KDD Cup 1999 dataset is a widely used benchmark dataset for intrusion detection research. It contains network traffic data captured in a simulated environment, including normal traffic and various types of attacks.

11. Packet Capture Tools:

Wireshark, TCPdump, Scapy, pcapy: These are packet capture tools used for capturing and analyzing network traffic. Wireshark is a popular network protocol analyzer, while TCPdump is a command-line packet sniffer. Scapy is a Python library for crafting and dissecting network packets, and pcapy is a Python interface to libpcap, the packet capture library used by TCPdump.

Figure 14 Wireshark

By leveraging these tools and technologies, I was able to effectively develop, manage, and analyze your project, from coding and version control to data visualization and machine learning model development.

2.2. Project Development Methodology

From different types of methodology for SDLC (Software Development Life Cycle), I have decided to follow RUP (Rational Unified Process) methodology. Rational Unified Process (RUP) is an iterative software development methodology that provides a disciplined approach to assigning tasks and responsibilities within a software development organization. It emphasizes an iterative development cycle, use-case driven techniques, architecture-centric approach, and component-based development. RUP divides the software development process into four phases: Inception, Elaboration, Construction, and Transition, each with specific goals and activities.

RUP promotes the use of best practices in software engineering and project management, encouraging collaboration among team members and allowing flexibility to adapt to changing requirements. It is based on industry-standard practices, focusing on iterative development, continuous risk management, and the importance of defining and maintaining the system architecture throughout the development process. RUP guides software development teams through a structured approach, ensuring the delivery of high-quality software that meets user needs (Study, 2022).

2.3. Review of the Similar Project

2.3.1 snort:

Snort is an open-source network intrusion detection system (NIDS) that is widely used for detecting and preventing network attacks. It was created by Martin Roesch in 1998 and is now maintained by Cisco. Snort is known for its flexibility, speed, and effectiveness in detecting a wide range of network threats.

Figure 15: Snort

Key features of Snort include:

1. Signature-Based Detection: Snort uses a set of predefined rules or signatures to detect known threats and malicious activity in network traffic. These rules can be customized and updated regularly to adapt to new threats.
2. Protocol Analysis: Snort can analyze network protocols at a deep level, allowing it to detect anomalies and suspicious patterns that may indicate a network attack.
3. Real-Time Alerting: Snort can generate real-time alerts when it detects a potential threat, allowing network administrators to respond quickly to mitigate the impact of an attack.
4. Packet Logging: Snort can log network traffic that matches its detection rules, providing a detailed record of network activity for analysis and forensics purposes.
5. Integration with Other Security Tools: Snort can be integrated with other security tools and systems, such as firewalls and intrusion prevention systems (IPS), to enhance overall network security.

2.3.2 Comparison beteen snort and therat guard

Feature	Treat Guard	Snort
Type	Host-based Intrusion Detection System (HIDS)	Network Intrusion Detection System (NIDS)
Focus	Monitoring and securing individual host devices	Monitoring and securing network traffic
Port Security	Provides individual port status check and control	Focuses on network traffic analysis and detection
Deployment	Deployed on individual host devices	Deployed on network infrastructure
Detection Approach	Monitors host device logs and activity	Analyzes network traffic for suspicious activity
Detection Mechanism	Uses machine learning for anomaly detection	Uses signature-based detection
Alerting	Sends alerts to admin via email for cyber attacks	Sends alerts for detected network threats
Customization	Can be customized for specific host configurations	Can be customized with user-defined rules
Scalability	Suitable for small to medium-scale environments	Suitable for large-scale network environments
Integration	Integrates with web interface for admin control	Integrates with other security tools and systems
Maintenance	Requires regular updates and monitoring	Requires regular rule updates and tuning
Use Case	Ideal for securing individual devices	Ideal for securing network infrastructure

Figure 16: Comparision between snort and ThreatGuard

3. Project Development

3.1. Requirement Gathering

During the requirement gathering phase, I conducted extensive research on existing projects and Proof of Concepts (POCs) related to Host-based Intrusion Detection Systems (HIDS) using machine learning (ML). This research aimed to gain insights into the best practices, challenges, and potential approaches for developing an effective HIDS.

Based on this research, I documented a comprehensive set of requirements for the HIDS project. This included both functional and non-functional requirements. Functional requirements outlined the specific features and capabilities the HIDS should have, such as monitoring host devices, detecting intrusions, and providing alerts to administrators. Non-functional requirements focused on quality attributes like scalability, reliability, and ease of use.

I also detailed the ML requirements for the HIDS, specifying the algorithms to be used, sources of training data, and criteria for evaluating the performance of the ML models. Additionally, I outlined the data collection and analysis methods, user interface design,

security measures, integration requirements with existing systems, and testing and validation procedures.

By documenting these requirements, I aimed to ensure that the HIDS project is well-defined and aligned with the goals of enhancing network security through ML-based intrusion detection.

3.2. Use Case

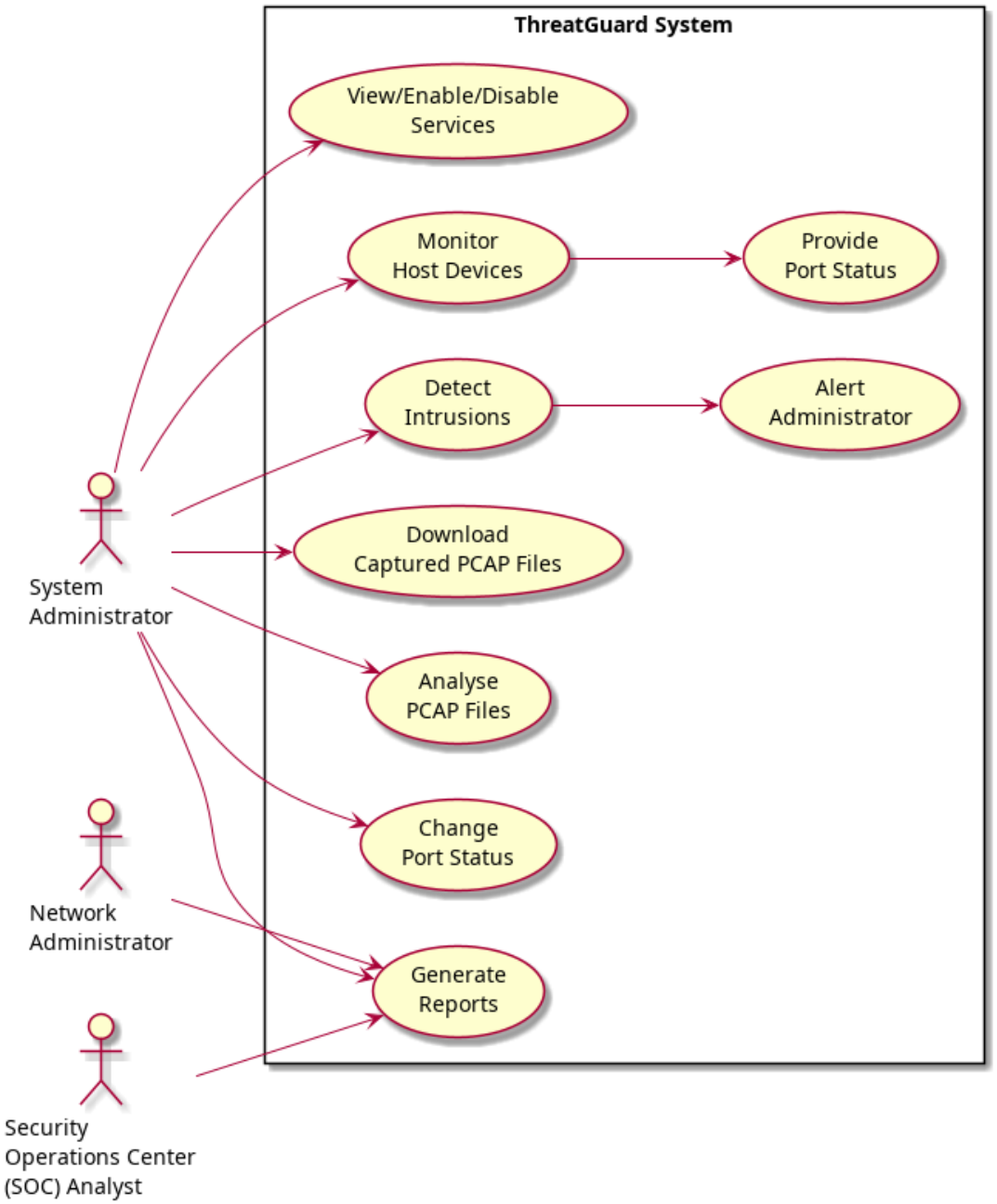


Figure 17: Use Case Diagram

3.3. SRS Document

The full description of SRS document is kept in Appendix section i.e. SRS Document.

3.4. Wireframes

The images of wireframes are kept in Appendix section i.e. Wireframes with its full description.

3.5. ERD

Note: Identification process of entities, Relation between entities are kept in appendix section i.e. Entity Relationship Diagram with full description.

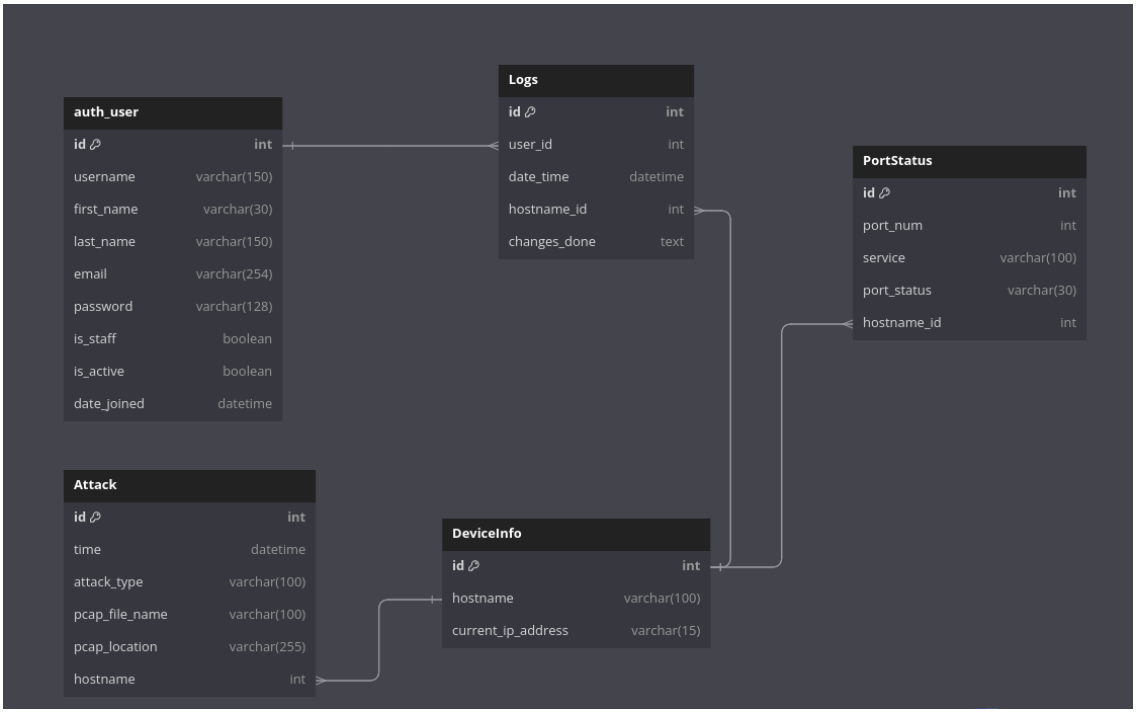


Figure 18: Initial ERD

4. Analysis of progress

Analysis of progress sections includes the progress of second year project and how it is developing? This section helps to determine the actual progress of project by comparing with Gantt Chart.

4.1. Progress Table

Phase	Tasks	Progress
Inception	Resource Gathering	100%
	Project Planning	100%
	Risk Management	100%
	Inception Phase Review and Approval	100%
Elaboration	Problem Domain Analysis	100%
	ERD Development	100%
	Use Case Diagram Development	100%
	Wireframe Design	85%
	Elaboration Phase Review and Approval	100%
Construction	Dataset Cleaning	100%
	Machine Learning Model Development	100%
	Real-time Data Analysis Module Development	0%
	Frontend Development	20%
	Backend Development	10%
	Database Integration	45%
	Construction Phase Review and Approval	0%
Transition	Integration and System Testing	0%
	Real World Tests and Attacks Prediction	0%
	Documentation	0%
	Ongoing System Monitoring and Perf. Opt.	0%
	Deployment	0%
	Transition Phase Review and Approval	0%

Figure 19: Progress Table

4.2. Progress Timeline (Gantt Chart)

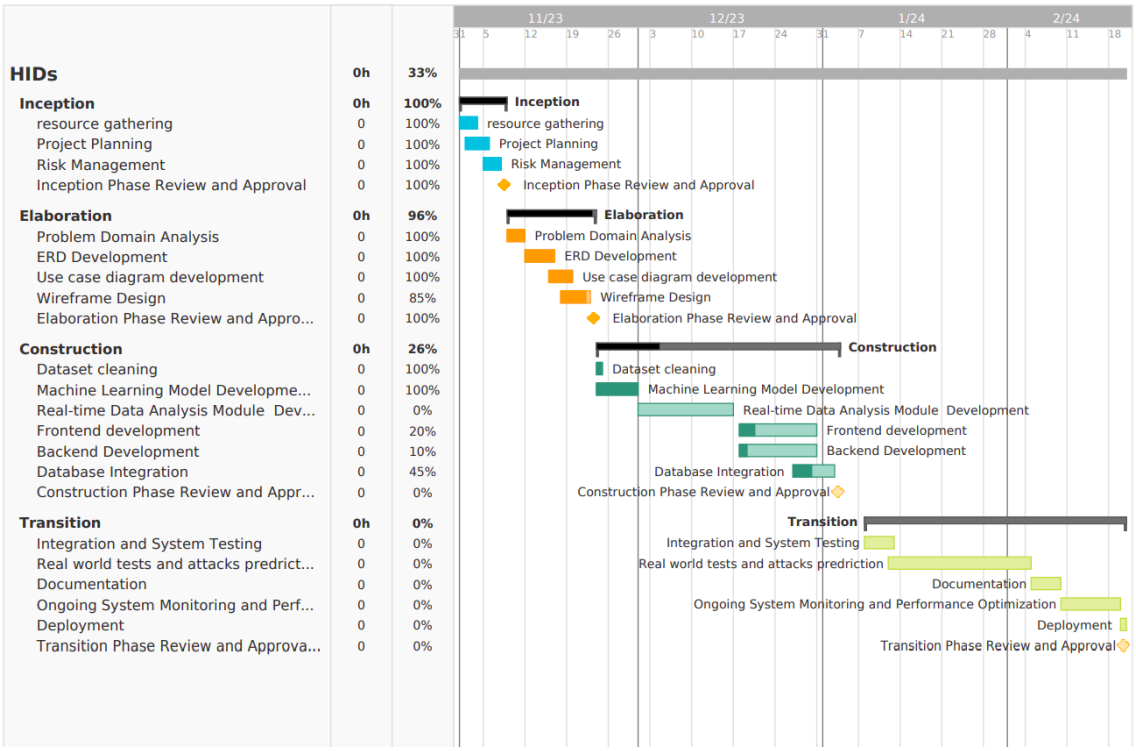


Figure 20: Gantt Chart

5. Future work

1. Advanced Machine Learning Algorithms: Explore the implementation of advanced machine learning algorithms, such as deep learning models, to improve the accuracy and effectiveness of intrusion detection.
2. Real-time Data Analysis: Enhance the system to perform real-time data analysis for quicker detection and response to intrusions.
3. Behavioral Analysis: Introduce behavioral analysis techniques to detect anomalies in user behavior and network traffic patterns, enabling proactive threat detection.
4. Integration with Threat Intelligence Feeds: Integrate the system with external threat intelligence feeds to enhance detection of known threats and vulnerabilities, and improve response strategies.
5. Automated Response Mechanisms: Implement automated response mechanisms to mitigate detected threats, such as isolating compromised devices or blocking malicious traffic, reducing manual intervention.

6. Scalability and Performance Optimization: Optimize the system for scalability and performance to handle larger networks and higher traffic volumes, ensuring efficient operation in dynamic environments.
7. User Interface Enhancements: Improve the user interface to provide more detailed insights and visualization of network activity and detected threats, enhancing usability and decision-making.
8. Cross-platform Compatibility: Ensure the system is compatible with a wide range of operating systems and network environments for broader deployment and increased accessibility.
9. Continuous Monitoring and Updates: Implement a system for continuous monitoring of network activity and regular updates to the intrusion detection algorithms to adapt to new threats and attack vectors.
10. Integration with Security Information and Event Management (SIEM) Systems: Integrate the HIDS with SIEM systems for centralized monitoring and management of security events, streamlining security operations.

These future works aim to enhance the capabilities of the Host-based Intrusion Detection System (HIDS) and ensure its effectiveness against evolving cyber threats.

6. References

7. Appendix

7.1. SRS Document

1. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements for the development of a Host-based Intrusion Detection System (HIDS) using machine learning (ML). This system, named ThreatGuard, aims to enhance network security by monitoring and detecting intrusions on host devices.

1.2 Scope

The scope of the project includes the development of a HIDS that can monitor host devices, detect intrusions using ML algorithms, and provide alerts to administrators. The system will also allow administrators to view and administer host devices, including changing port status and managing services.

1.3 Definitions, Acronyms, and Abbreviations

- HIDS: Host-based Intrusion Detection System
- ML: Machine Learning
- SOC: Security Operations Center

1.4 References

- KDDCUP99 dataset: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

2. Overall Description

2.1 Product Perspective

ThreatGuard will operate as a standalone system, interacting with host devices to monitor and detect intrusions. It will also provide a web interface for administrators to view and administer host devices.

2.2 Product Features

- Monitor Host Devices: ThreatGuard will monitor host devices for suspicious activity and potential intrusions.
- Detect Intrusions: The system will use ML algorithms to detect intrusions based on network traffic patterns.
- Alert Administrator: ThreatGuard will provide alerts to administrators when intrusions are detected.
- View and Administer Host Devices: Administrators will be able to view and

administer host devices, including changing port status and managing services.

2.3 User Classes and Characteristics

- System Administrator: Responsible for managing and configuring ThreatGuard.
- Network Administrator: Responsible for network operations and security.
- SOC Analyst: Responsible for monitoring and analyzing security incidents.

2.4 Operating Environment

ThreatGuard will be developed using Django for the backend and HTML, CSS, and JavaScript for the frontend. It will be deployed on a server running Python and SQLite3 for the database.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Monitor Host Devices

- The system shall monitor network traffic on host devices.
- The system shall analyze network traffic patterns for suspicious activity.

3.1.2 Detect Intrusions

- The system shall use ML algorithms to detect intrusions.
- The system shall classify detected intrusions based on severity.

3.1.3 Alert Administrator

- The system shall provide alerts to administrators when intrusions are detected.
- The system shall send alerts via email.

3.1.4 View and Administer Host Devices

- Administrators shall be able to view a list of host devices.
- Administrators shall be able to change the status of ports on host devices.
- Administrators shall be able to view and manage services running on host devices.

3.2 Non-Functional Requirements

3.2.1 Security

- The system shall require authentication for access.
- The system shall encrypt sensitive data.

3.2.2 Performance

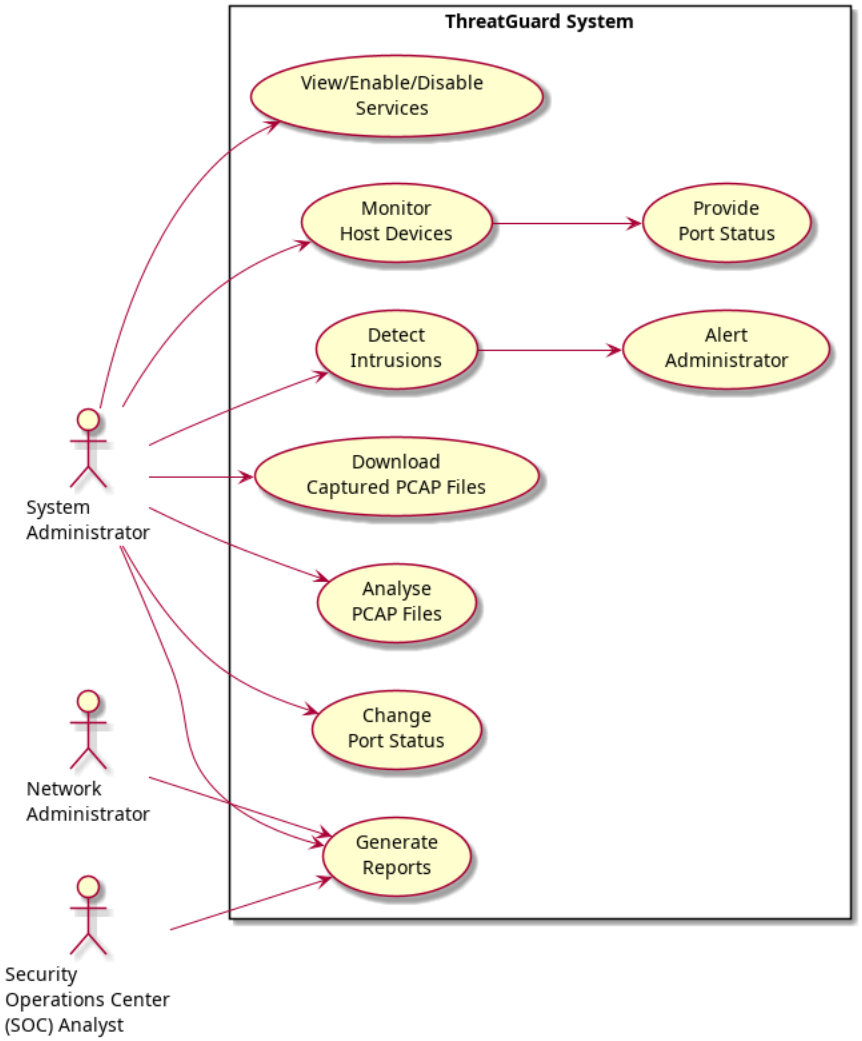
- The system shall be able to process network traffic in real-time.
- The system shall be able to handle a large number of host devices.

3.2.3 Usability

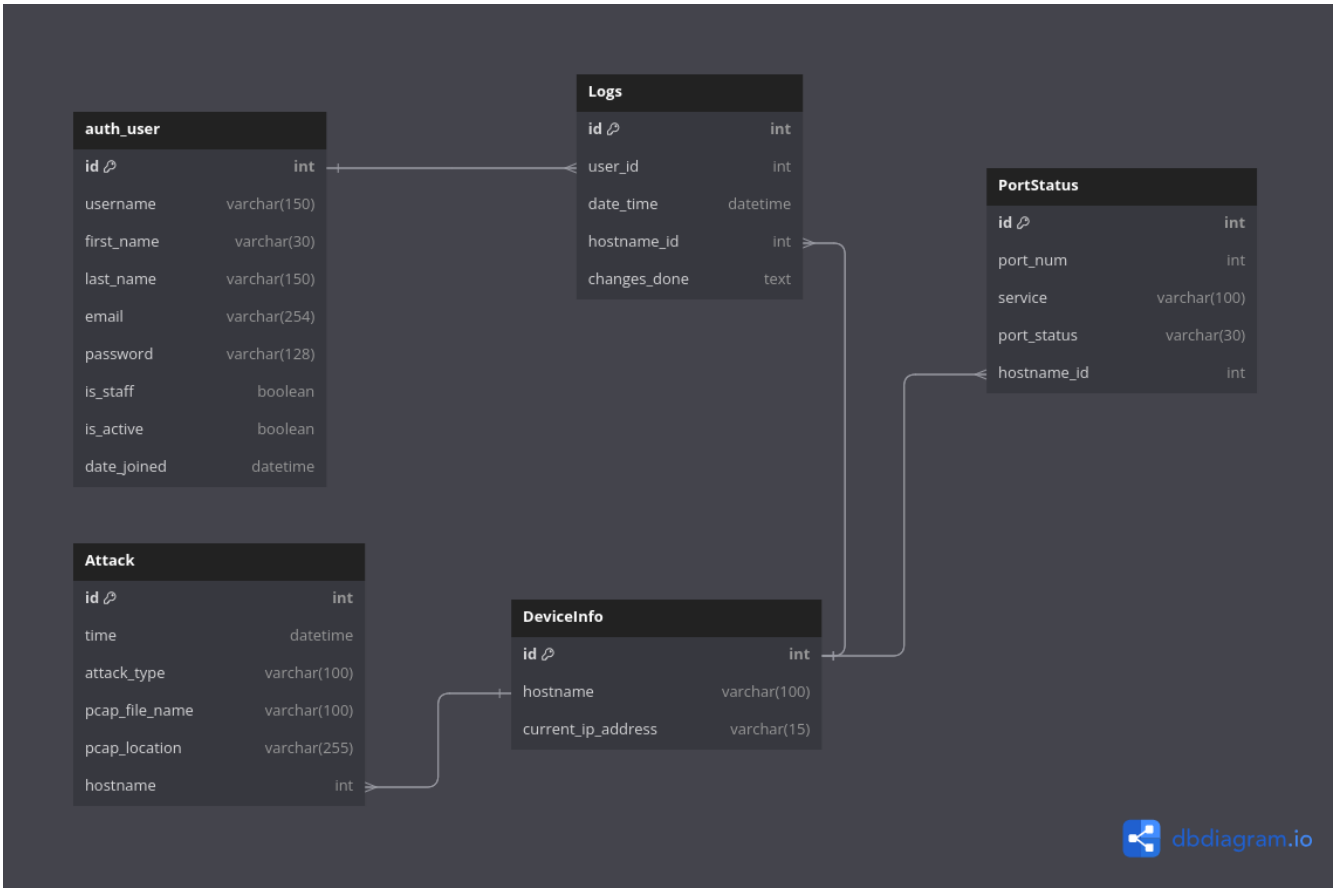
- The system shall have a user-friendly interface.
- The system shall provide clear and informative alerts.

4. Appendix

4.1 Use Case Diagram



4.2 Entity Relationship Diagram

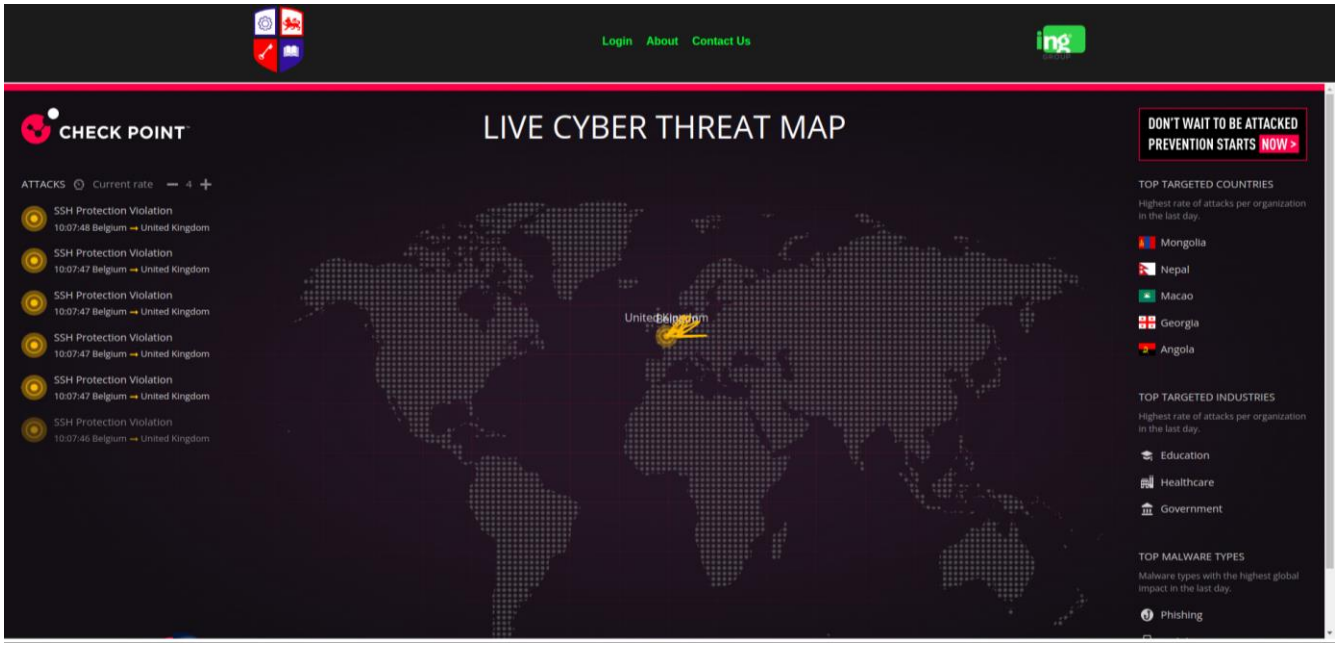


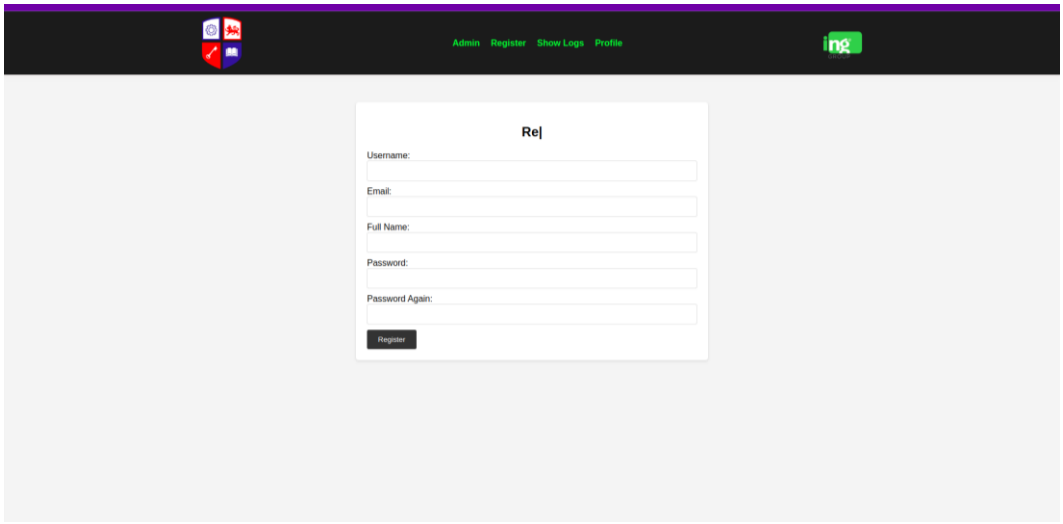
4.3 Glossary

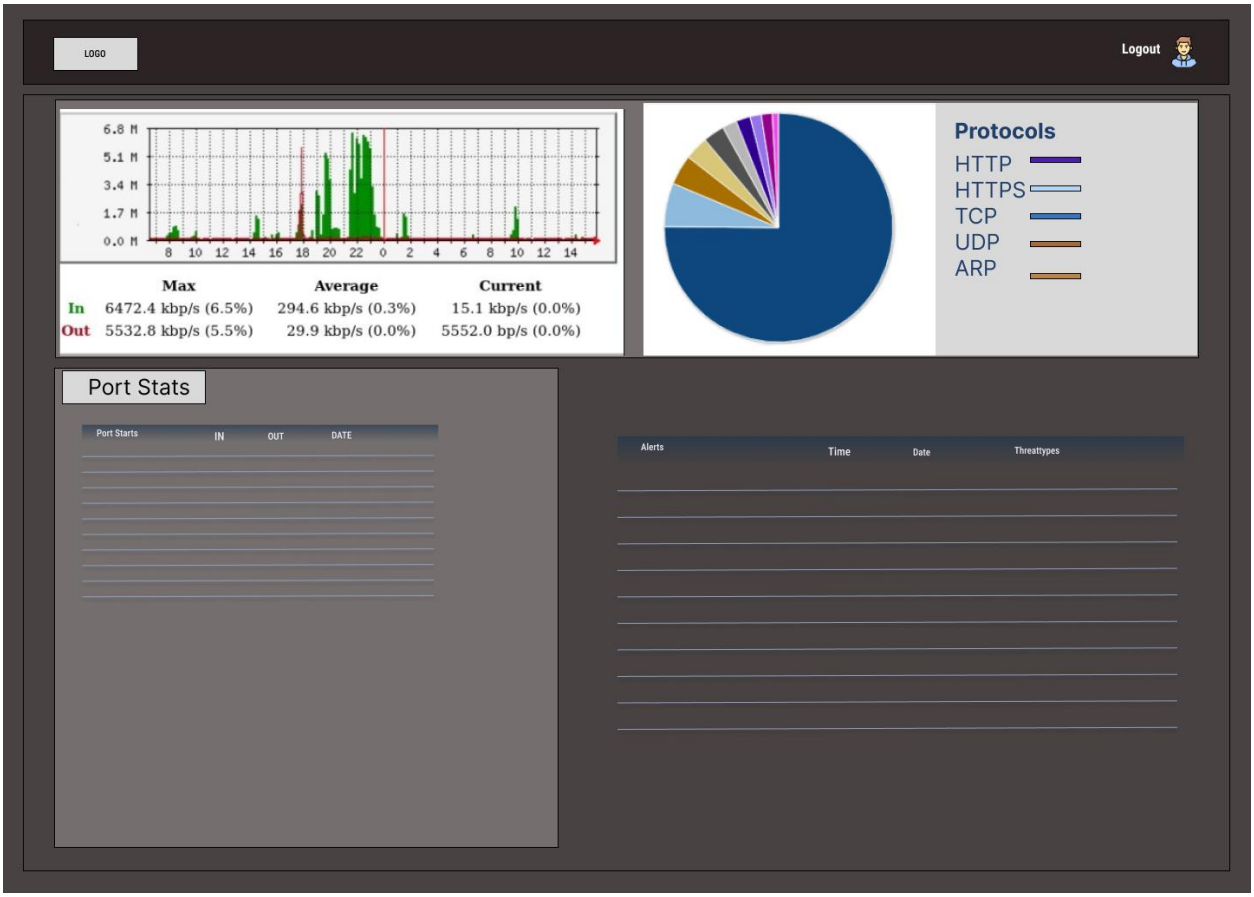
- IDS: Intrusion Detection System
- RFC: Random Forest Classifier

This concludes the Software Requirements Specification for ThreatGuard.

7.2. Wireframes







7.3. Entity Relationship Diagram

Identification of entity and their attributes:

1. **auth_user**:
    - Entity: User
    - Attributes:
        - id (primary key)
        - username

- first_name
- last_name
- email
- password
- is_staff
- is_active
- date_joined

2. **DeviceInfo**:

- Entity: Device
- Attributes:
  - id (primary key)
  - hostname
  - current_ip_address

3. **Logs**:

- Entity: Log
- Attributes:
  - id (primary key)
  - user_id (foreign key referencing auth_user.id)
  - date_time
  - hostname_id (foreign key referencing DeviceInfo.id)
  - changes_done

4. **PortStatus**:

- Entity: Port
- Attributes:
  - id (primary key)
  - port_num
  - service
  - port_status

- hostname_id (foreign key referencing DeviceInfo.id)

5. **Attack**:

- Entity: Attack
- Attributes:
  - id (primary key)
  - time
  - attack_type
  - pcap_file_name
  - pcap_location
  - hostname_id (foreign key referencing DeviceInfo.id)

These entities represent the main components of the system and their associated attributes.

Here's a brief explanation of each entities:

1. **auth_user**: Stores user authentication information, such as username, first name, last name, email, password hash, and metadata like whether the user is a staff member or active, and the date they joined.
2. **DeviceInfo**: Contains information about devices, including a unique identifier (`id`), hostname, and current IP address.
3. **Logs**: Records logs of changes or actions performed by users, referencing the `auth_user` table (`user_id`) and the `DeviceInfo` table (`hostname_id`). It also includes a timestamp (`date_time`) and details of the changes (`changes_done`).
4. **PortStatus**: Tracks the status of ports on devices, including the port number, the service running on the port, and the current status (defaulting to 'Inactive'). It references the `DeviceInfo` table (`hostname_id`).
5. **Attack**: Stores information about detected attacks, including the timestamp (`time`), the type of attack (`attack_type`), the name of the pcap file associated with the attack (`pcap_file_name`), the location of the pcap file

(`pcap_location`), and the affected device (`hostname_id`).

Relationship between entities:

1. **auth_user** and **Logs**:
    - One-to-Many Relationship: Each user can have multiple logs (changes_done), but each log belongs to only one user. This is represented by the foreign key `user_id` in the Logs table referencing the `id` primary key in the auth_user table.
2. **DeviceInfo** and **Logs**:
    - One-to-Many Relationship: Each device can have multiple logs (changes_done), but each log is associated with only one device. This is represented by the foreign key `hostname_id` in the Logs table referencing the `id` primary key in the DeviceInfo table.
3. **DeviceInfo** and **PortStatus**:
    - One-to-Many Relationship: Each device can have multiple port statuses, but each port status is associated with only one device. This is represented by the foreign key `hostname_id` in the PortStatus table referencing the `id` primary key in the DeviceInfo table.
4. **DeviceInfo** and **Attack**:
    - One-to-Many Relationship: Each device can be associated with multiple attacks, but each attack is associated with only one device. This is represented by the foreign key `hostname_id` in the Attack table referencing the `id` primary key in the DeviceInfo table.

7.4. Project as a solution

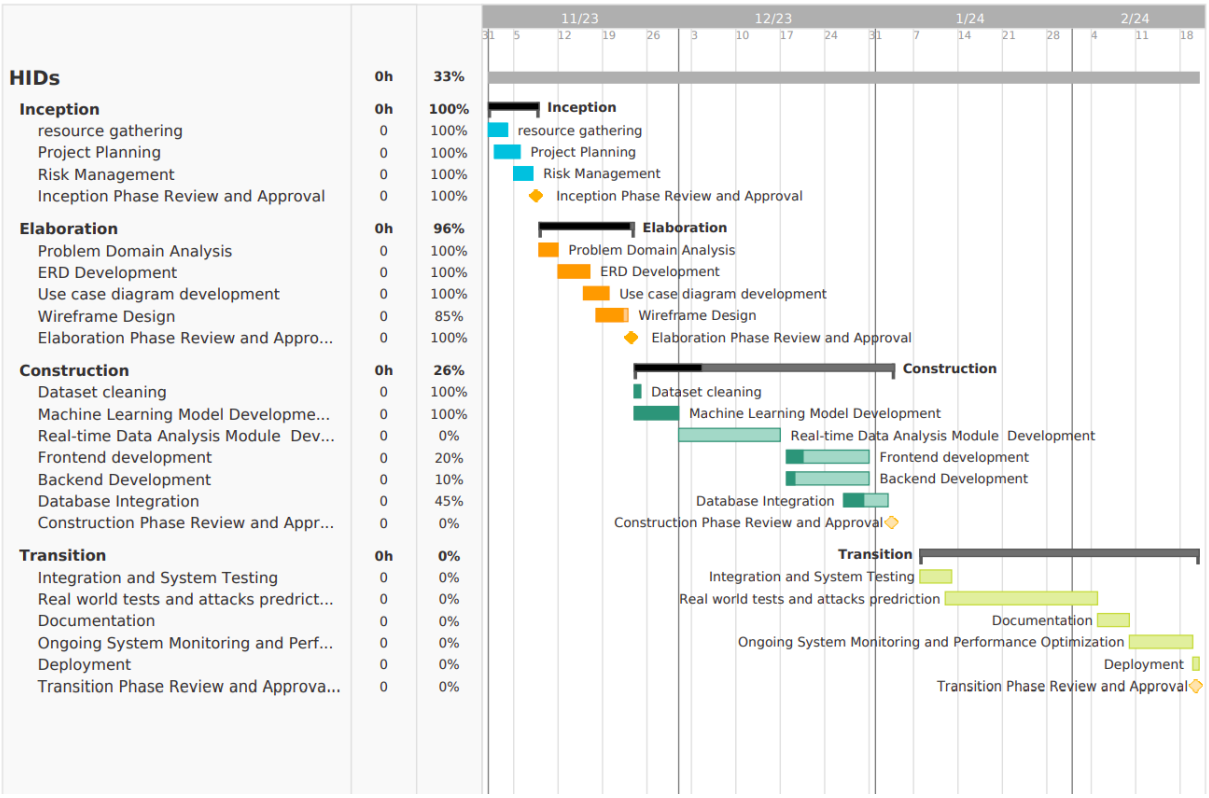
The "ThreatGuard" project aims to provide a comprehensive solution for network security by developing a sophisticated Host-based Intrusion Detection System (HIDS) using machine learning. By leveraging advanced machine learning algorithms, particularly the Random Forest Classifier, the system can effectively detect and mitigate various network attacks, including known and unknown threats.

The key components of the solution include:

1. **Intrusion Detection**: The HIDS monitors network activities in real-time, analyzing patterns and anomalies to detect suspicious behavior indicative of an intrusion.
2. **Machine Learning**: By training on the KDDCUP99 dataset, the system can classify network traffic and identify potential threats with high accuracy, minimizing false positives and negatives.
3. **Alerting Mechanism**: Upon detecting an intrusion, the system alerts administrators via email, enabling swift response and mitigation of the threat.
4. **Port Status Check**: The system provides functionality to check and control individual port statuses, allowing administrators to enable or disable ports and manage services running on them.
5. **User Interface**: An intuitive web interface enables administrators to visualize network activity, view intrusion alerts, and manage system settings, enhancing usability and accessibility.
6. **Scalability and Performance**: The system is designed to be scalable and performant, capable of handling large networks and high traffic volumes without compromising on detection accuracy.
7. **Future Enhancements**: Potential future enhancements include the integration of advanced machine learning algorithms, real-time data analysis, and automated response mechanisms to further enhance the system's capabilities.

Overall, the "ThreatGuard" project serves as a robust and efficient solution for network security, providing organizations with the tools and technologies needed to protect their networks against evolving cyber threats.

7.5. Project Gantt Chart



7.6. Selected Methodology

From different types of methodology for SDLC (Software Development Life Cycle), I have decided to follow RUP (Rational Unified Process) methodology. Rational Unified Process (RUP) is an iterative software development methodology that provides a disciplined approach to assigning tasks and responsibilities within a software development organization. It emphasizes an iterative development cycle, use-case driven techniques, architecture-centric approach, and component-based development. RUP divides the software development process into four phases: Inception, Elaboration, Construction, and Transition, each with specific goals and activities.

RUP promotes the use of best practices in software engineering and project management, encouraging collaboration among team members and allowing flexibility to adapt to changing requirements. It is based on industry-standard practices, focusing on iterative development, continuous risk management, and the importance of defining and maintaining the system architecture throughout the development process. RUP guides software development teams through a structured approach, ensuring the delivery of

high-quality software that meets user needs. (study.com, 2022)

My project plan according to this methodology is described below:

1. Inception

The Inception phase is the first phase of the RUP methodology, where the project's feasibility and scope are assessed. During this phase, the project's objectives, initial requirements, stakeholders, and potential risks are identified. The goal is to establish a clear understanding of the project's purpose and to justify its initiation. In this step of my project, I will be focusing on the following things which are mentioned below:

- Resource gathering
- Project planning
- Risk management

2. Elaboration

The Elaboration phase is the second phase of the RUP methodology, focusing on detailed planning and risk mitigation. In this phase, detailed requirements are gathered, architectural foundations are established, and major risks are addressed. The goal is to refine the project vision, establish a stable architecture, and create a solid foundation for development. In this step for my project I will be doing the following:

- Analyzing the problem domain
- Development of the use case diagram and ERD
- Analyzing the functional and nonfunctional developments
- Wireframe design and development

3. Construction

The Construction phase is the third phase of the RUP methodology, where the actual development of the software takes place. During this phase, the software is incrementally built, features are implemented, and components are integrated. The focus is on iterative development, ensuring that the software is functional, reliable, and feature complete. In

this construction step I will be doing the following:

- Dataset Cleaning
- Machine learning model development
- Real time packet analysis module development
- Frontend development
- Backend development
- Database integration

4. Transition

The Transition phase is the final phase of the RUP methodology, involving the deployment of the software to end-users. In this phase, the software undergoes user acceptance testing, deployment planning is finalized, end-users are trained, and ongoing support mechanisms are established. The goal is to transition the software from development to production, ensuring a smooth deployment and operational continuity.