Module Code & Module Title

CS6PO5NT-Final Year Project

Assessment Weightage & Type

Final Year Project Proposal (5%)

Year and Semester

2023-24 Autumn

Student Name: Aayush Wanem Limbu

London Met ID: 22072043

College ID: NP05CP4A220010

Assignment Submission Date: December 4, 2024

Internal Supervisor: Amit Shrestha

External Supervisor: Prakash Basnet

Title: FYP-NIDS

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline 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

Contents

Table of Figures

1. Introduction

In today's digital age, cyber threats are becoming increasingly sophisticated and frequent. To stay ahead of these threats, organizations need robust security solutions. Network Intrusion Detection Systems (NIDS) are a vital tool for safeguarding networks by monitoring network traffic for malicious activity.

A NIDS (Network Intrusion Detection System) is a security technology that monitors and analyzes network traffic for signs of malicious activity, unauthorized access, or security policy violations. The primary function of a NIDS is to detect and alert network administrators of any potential or ongoing attacks on the network.

NIDS examines data packets for specific patterns and behaviors that indicate the presence of an attack. It can detect and alert network administrators of attacks such as DoS (Denial of Service), port scanning, virus and malware infections, and unauthorized access attempts. NIDS is an essential component of a comprehensive network security strategy. It helps to identify and respond to threats quickly before they can cause significant damage or compromise sensitive data. (Jacky, 2023)

Rule-based NIDS, such as Snort, rely on predefined sets of rules to identify malicious activities by matching patterns in network traffic against known signatures of attacks. This approach offers high accuracy in detecting well-documented threats and allows for customization based on specific organizational requirements. However, rule-based systems face challenges, including the need for regular rule updates, the inability to detect zero-day attacks, and the potential for performance bottlenecks in high-traffic environments.

As part of a comprehensive security strategy, NIDS plays a vital role in identifying and mitigating threats before they can cause significant damage. This project focuses on the design and implementation of a rule-based NIDS, similar to Snort, aiming to provide a customizable and efficient detection mechanism for known threats. The system will emphasize the creation and testing of effective detection rules to address modern cybersecurity challenges while maintaining scalability and ease of use.

1.1 Introduction to Topic

The rapid advancement of technology and the increasing reliance on interconnected systems have made cybersecurity a critical concern for organizations worldwide. With cyber threats becoming more sophisticated and frequent, traditional security measures are no longer sufficient to protect sensitive information and critical infrastructure. Malicious activities such as Denial of Service (DoS) attacks, port scanning, and malware infections pose significant risks, requiring innovative solutions to detect and mitigate these threats effectively.

Network Intrusion Detection Systems (NIDS) play a vital role in modern security frameworks by monitoring and analyzing network traffic for signs of malicious activity. These systems provide an additional layer of defense, alerting administrators to potential security breaches before significant damage occurs. NIDS have evolved over time, with approaches ranging from signature-based methods, which rely on predefined rules, to anomaly-based techniques that detect deviations from normal behavior.

This project focuses on a rule-based NIDS, similar to Snort, which leverages predefined rules to identify malicious patterns in network traffic. By integrating a web-based admin dashboard, the system offers a user-friendly platform for managing detection rules, viewing alerts, and analyzing network activity, ensuring an efficient and scalable approach to network security.

1.2 Problem Scenario

In today's interconnected world, networks are increasingly targeted by malicious actors looking to exploit vulnerabilities for unauthorized access, data theft, or disruption of services. As businesses and organizations depend heavily on their network infrastructure to operate, the need for robust security systems to monitor and protect network traffic has never been greater. However, traditional security mechanisms often struggle to keep up with the evolving threat landscape, leading to gaps in defense and missed detection of potential attacks.

One critical challenge is the detection of known threats, which relies on predefined signature-based methods. These systems, such as Snort, use a set of rules to identify specific attack patterns. While effective against well-documented threats, signature-

based systems have notable limitations. They often fail to detect new, unknown threats (zero-day attacks), as they require pre-existing knowledge of attack patterns. Furthermore, these systems can become overwhelmed in high-traffic environments, resulting in performance bottlenecks and slower detection times. Additionally, managing and updating the rules is an ongoing challenge, requiring continuous maintenance to ensure the system remains effective.

Organizations also face difficulties in customizing these rule-based systems to match their unique network environments, making it hard to tailor detection to specific use cases. The lack of real-time and user-friendly interfaces for network monitoring further exacerbates the issue, making it difficult for administrators to react quickly to emerging threats.

This project aims to address these challenges by developing a Network Intrusion Detection System (NIDS) that combines rule-based detection with an easy-to-use web dashboard for real-time monitoring. The system will be designed to efficiently handle network traffic while offering scalability, customization, and improved detection capabilities for known threats. By providing administrators with a powerful tool to identify and mitigate potential security risks, this NIDS will contribute to enhancing overall network security.

1.3 Project as a Solution

In response to the growing challenges faced by organizations in securing their networks, this project proposes the development of a **Snort-compatible, rule-based Network Intrusion Detection System (NIDS)**, paired with a **web-based admin dashboard** for simplified monitoring and management and also a pcap file analysis with report generation. This solution aims to address the limitations of traditional NIDS by combining reliable, signature-based detection with a user-friendly interface and the scalability required for modern networks.

The system will leverage **Snort's well-established rule set**, which provides a robust framework for detecting known attack patterns such as denial-of-service (DoS) attacks, malware infections, and unauthorized access attempts. By using Snort-compatible rules, the NIDS ensures that the system can effectively identify and respond to threats based on predefined attack signatures, offering reliable protection for a wide range of known

threats. At the same time, the flexibility of Snort rules allows organizations to tailor the system to their specific needs, ensuring that it can be customized to match the unique characteristics of their network.

The **web-based admin dashboard** will be a key component of the solution, providing an intuitive and accessible interface for real-time monitoring, alerts, and incident response. Traditionally, many NIDS require complex command-line configurations, making them difficult for non-technical staff to manage effectively. By offering a user-friendly dashboard, this project aims to bridge the gap between technical and non-technical users, ensuring that security teams of all sizes can easily navigate the system, identify potential threats, and take timely action. This dashboard will provide a clear view of network traffic, alert status, and ongoing attacks, allowing administrators to respond more quickly and efficiently to security incidents.

To address scalability concerns, the solution will be designed to handle large volumes of network traffic without overwhelming the system. It will include features to prioritize critical alerts and filter out false positives, reducing the risk of alert fatigue and ensuring that security teams can focus on the most important incidents. Furthermore, the system will be built with scalability in mind, allowing it to grow alongside an organization’s network infrastructure. This will ensure that as networks expand, the NIDS remains effective at detecting and responding to threats in a timely manner.

2. Aims and Objectives

2.1 Aims

Below are the major aims that this project is going to accomplish.

- **Develop a Rule-Based Intrusion Detection System (NIDS)**:

To design and implement a Network Intrusion Detection System that utilizes a rule-based approach, similar to Snort, to identify and respond to known network threats such as Denial of Service (DoS), port scanning, malware infections, and unauthorized access attempts.

- **Real-Time Network Traffic Monitoring**:

To enable real-time monitoring of network traffic, leveraging libraries like Scapy, and implement efficient packet capture techniques to analyze and detect potential threats as they occur on the network.

- **Create a Customizable Rule Engine**:

To provide a flexible and scalable rule management system that allows network administrators to customize and update detection rules based on their specific network environments and security requirements.

- **Enable PCAP File Analysis**:

To implement functionality for uploading and analyzing PCAP files, allowing for historical traffic analysis and enabling administrators to investigate past network activity for potential threats or security incidents.

- **Alerting and Reporting System**:

To create an alerting and reporting system that categorizes detected threats by severity and provides clear and actionable information to network administrators, ensuring timely responses to incidents.

- **Visualization of Network Devices and Topology**:

To implement visualizations on the dashboard that represent network topology and display devices on the network, highlighting any unusual or unauthorized devices or behavior.

- **Scalability and Performance Optimization**:

To optimize the performance of the NIDS, ensuring that it can handle high traffic volumes without significant performance degradation, and develop strategies for scaling the system to cover larger networks if needed.

- **Contribute to Enhanced Network Security**:

To provide a comprehensive, reliable, and scalable solution for enhancing network security, helping organizations detect and mitigate cyber threats before they can cause significant damage.

- **Build a User-Friendly Web Dashboard**:

To develop an intuitive and interactive web-based dashboard using Django that will provide real-time statistics, alerts, and detailed insights into the captured network traffic, helping administrators make informed decisions quickly.

2.2 Objectives

- **Implement Real-Time Packet Capture and Analysis**:

To develop functionality that continuously captures and analyzes network packets in real-time using tools like Scapy, enabling the detection of suspicious activity and intrusions as they happen on the network.

- **Design and Integrate Rule-Based Detection Mechanism**:

To create a rule-based detection engine that matches network traffic against predefined attack signatures, enabling the system to identify known threats such as DoS attacks, malware, port scanning, and unauthorized access attempts.

- **Provide Customizable Detection Rules**:

To build a system that allows administrators to easily customize, add, or remove detection rules based on specific organizational needs or emerging threats, ensuring flexibility and adaptability in the system.

- **Develop a Web-Based Dashboard for Monitoring and Reporting**:

To design a web-based user interface using Django that allows network administrators to view real-time traffic statistics, alerts, and network activity, facilitating better decision-making and timely responses to detected threats.

- **Implement PCAP File Upload and Analysis**:

To enable the system to process and analyze uploaded PCAP files, providing a comprehensive historical view of network traffic to identify previously undetected threats or to analyze past security incidents.

- **Design an Alerting System with Severity Categorization**:

To create an alerting system that categorizes detected threats by their severity (e.g., critical, high, medium, low) and notifies administrators, accordingly, ensuring that urgent issues are prioritized.

- **Visualize Network Topology and Device Activity**:

To develop visualizations on the dashboard that represent network topology, showing the relationships between devices and identifying any unusual or unauthorized devices or connections.

- **Ensure System Scalability and Performance Optimization**:

To optimize the system's performance to handle high volumes of network traffic without significant degradation, and to consider scalable architecture options for future expansion to larger network environments.

- **Test and Validate System Effectiveness**:

To rigorously test the system with various types of network traffic and security scenarios, ensuring its ability to accurately detect threats and provide meaningful insights into network behavior.

- **Provide Export and Reporting Features**:

To implement the ability to export analysis reports, alerts, and traffic data into multiple formats (e.g., CSV, PDF, HTML), enabling easy sharing and further analysis of the findings.

- **Enhance Network Security Posture**:

To ensure that the NIDS contributes effectively to an organization’s overall cybersecurity strategy by proactively identifying threats and enabling quick mitigation, thus enhancing network security and reducing the risk of data breaches or service disruption.

3. Expected Outcomes and Deliverables

After the successful development of this application, it will be able to perform the following tasks:

- A working NIDS(Network Intrusion Detection System) system that can capture and analyze live network traffic in real time.
- Detection of predefined attack patterns such as DoS, port scanning, and unauthorized access such as login attempt.
- A flexible rule management interface allowing admins to add, modify, or remove detection rules.
- A set of sample rules for common attacks that can be customized as needed.
- The ability to upload and analyze historical network traffic (PCAP files).
- A detailed report showing packet analysis, protocol breakdown, and identified threats.
- A web-based interface to display real-time alerts, traffic statistics, and detection reports.
- Visual representations (graphs, charts, network maps) for easier analysis.
- Real-time alerts sent to admins when suspicious activity is detected.
- A dynamic view of the network, showing connected devices and highlighting any suspicious or unauthorized devices.
- A system optimized to handle high traffic volumes without performance issues.
- Performance testing to ensure the system can process data in real time.

Ability to export detection results and analysis reports in CSV, PDF.

4. Project risks, threats and contingency plans

Many risks and threats are expected during the development of this system. Some of which are as follows:

a. The software may provide false alerts.

b. The NIDS may fail to alert if the payload is not recognized or if the rule is not set.

c. Software may not be able to work with every network protocol.

d. The project might be quite difficult to implement.

e. The resources on the internet might not be sufficient to complete the project.

The risks shown above can be minimized by the following contingency plans:

a. Researching about network and networking protocols.
b. Asking the supervisors for help in case resources are unavailable on the internet.
c. Trying to complete the project within the deadline.
d. Researching the various technologies involved.
e. Provide a warning that the system may detect false alerts and always double check the results.

5. Methodology

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond. This methodology outlines a series of steps that divide the software development process into tasks you can assign, complete, and measure. (aws.amazon.com, 2024)

For my Final Year Project, I have decided to use the Rational Unified Process(RUP) SDLC methodology.

The Rational Unified Process Methodology (RUP) is an agile software development method, in which the life cycle of a project, or the development of software, is divided into four phases. Various activities take place during these phases: modelling, analysis and design, implementation, testing and application.

RUP is iterative, meaning repeating; and agile. Iterative because all of the process's core activities repeat throughout the project. (Janse, 2024)

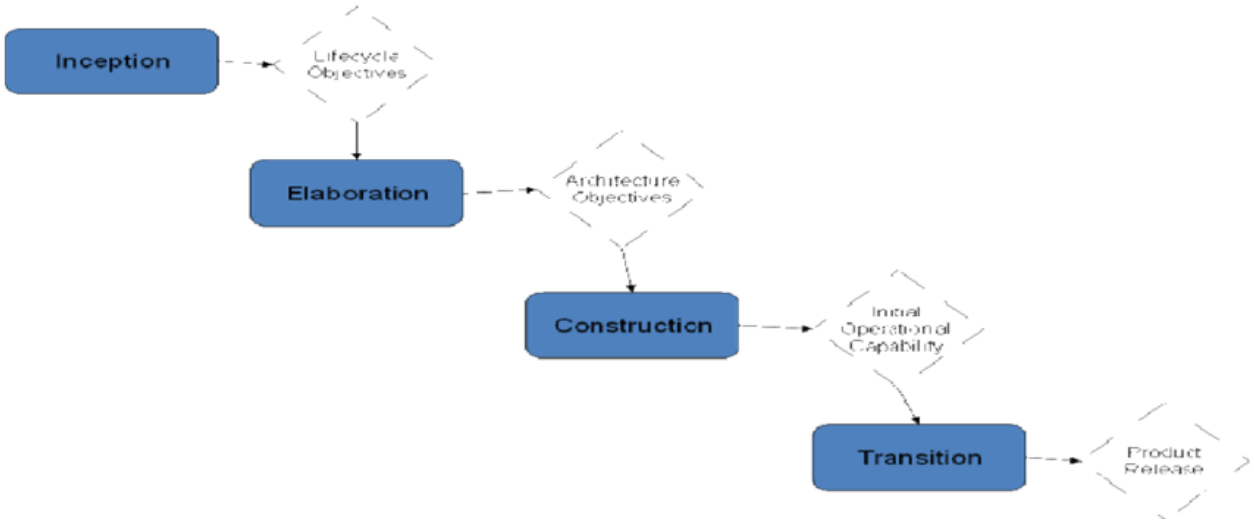


Figure 1 Phases of RUP Methodology

The Rational Unified Process (RUP) divides development into the four consecutive phases:

1. Inception phase
2. Elaboration phase
3. Construction phase
4. Transition phase

The first phase, Inception, is all about understanding the project and laying the groundwork. During this phase, the team defines the project's goals, determines its scope, and identifies stakeholders. They assess the feasibility of the project and consider potential risks. The focus is on ensuring that the project idea is both viable and worthwhile. By the end of this phase, the team delivers key documents like the vision statement, business case, and an initial project plan that outlines the project's direction.

Next comes the Elaboration phase, where the team dives deeper into the project requirements and designs the system architecture. This phase is like creating a detailed blueprint for the software, addressing major risks early, and refining the project plan. The team works to clarify how the software will work and might even create a prototype to validate ideas. The deliverables from this phase include the software architecture document and a more detailed project plan, setting a solid foundation for development.

The Construction phase is where the real work of building the software takes place. Developers focus on coding based on the designs created earlier, and testers rigorously

test the software to ensure it functions as expected. The team regularly reviews progress, addresses issues, and refines the product iteratively. The outcome of this phase is a functional version of the software, along with user manuals and training materials to help end-users understand the system.

Finally, the Transition phase prepares the software for deployment and real-world use. The team delivers the software to users, provides training, and offers support to ensure a smooth adoption process. They also address any remaining bugs or issues. By the end of this phase, the software is fully operational, and the team hands over a maintenance and support plan to keep the system running efficiently.

Each phase is finalized with a milestone. A milestone is a point in time where decisions of critical importance must be made. To be able to make those decisions, the objectives must have been accomplished.

6. Resource Requirements

6.1 Hardware Requirements

Development Workstation:

1. Operating System: A PC with Windows or Linux installed.

2. **Processor:** A modern processor (e.g., Intel i5 11th Gen or equivalent) to handle development tasks efficiently.

3. Memory: At least 8 GB RAM (16 GB recommended for smooth multitasking and virtualization).

4. Storage: Sufficient storage space for development tools, virtual machines, and project files.

Internet Connection:

5. Network Access: A stable, high-speed internet connection for downloading dependencies, packages, and collaborating with team members.

Testing Devices:

6. Physical Devices: PCs or mobile devices to generate and analyze network traffic.

6.2. Software Requirements

- **Integrated Development Environment (IDE):**

1. Visual Studio Code: The primary IDE for coding, debugging, and managing the backend and frontend.

- **Programming and Frameworks:**

2. Python (3.11 or later): Used for the backend and packet inspection module.

3. Django: A Python-based framework for backend development.

4. Scapy: A library for advanced network programming and packet manipulation.

5. HTML, CSS, and JavaScript: For front-end design and development.

- **Database and APIs:**

6. MySQL: For managing relational database operations.

7. GeoLite2 Database: To gather IP address information and reputation.

- **Version Control:**

8. Git and GitHub: For codebase management and collaboration.

Tools for Development and Diagramming:

9. Draw.io: For creating work breakdown structures (WBS) and other diagrams.

10. Team Gantt: For managing timelines and creating Gantt charts.

- **Virtualization Tools:**

11. VirtualBox or VMware: To run virtual machines for testing and network traffic generation.

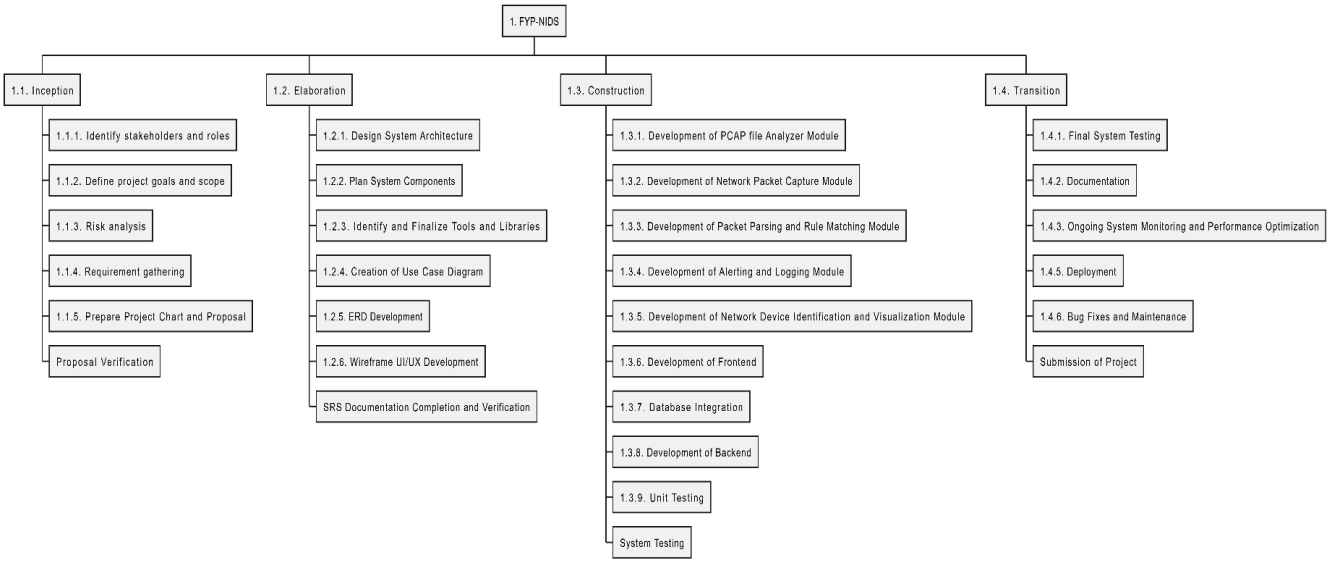
12. Operating System ISOs: Windows or Linux ISOs to set up virtual machines.

7. Work Break Down Structure

A work breakdown structure (WBS) is a visual, hierarchical and deliverable-oriented deconstruction of a project. It is a helpful diagram for project managers because it allows them to break down their project scope and visualize all the tasks required to complete their projects.

All the steps of project work are outlined in the work breakdown structure chart, which makes it an essential project planning tool. The final project deliverable, as well as the tasks and work packages associated with it rest on top of the WBS diagram, and the WBS levels below subdivide the project scope to indicate the tasks, deliverables and work packages that are needed to complete the project from start to finish.

Project managers make use of project management software to lay out and execute a work breakdown structure. When used in combination with a Gantt chart that incorporates WBS levels and task hierarchies, project management software can be especially effective for planning, scheduling and executing projects. (projectmanager.com, 2024)



The Work Break Down Structure of my project is made following the RUP methodology and is attached below:

Figure 2 Project WBS

8. Milestones Chart

A milestone chart is a project management chart that helps project managers visualize project milestones. It typically shows each milestone and when it needs to be completed, the status of the milestone (complete or incomplete) and sometimes the priority of the milestone. Many milestone charts use specific colors of symbols to show the most critical milestones. (Malsam, 2024)

The milestone chart of my Final Year Project is attached below.

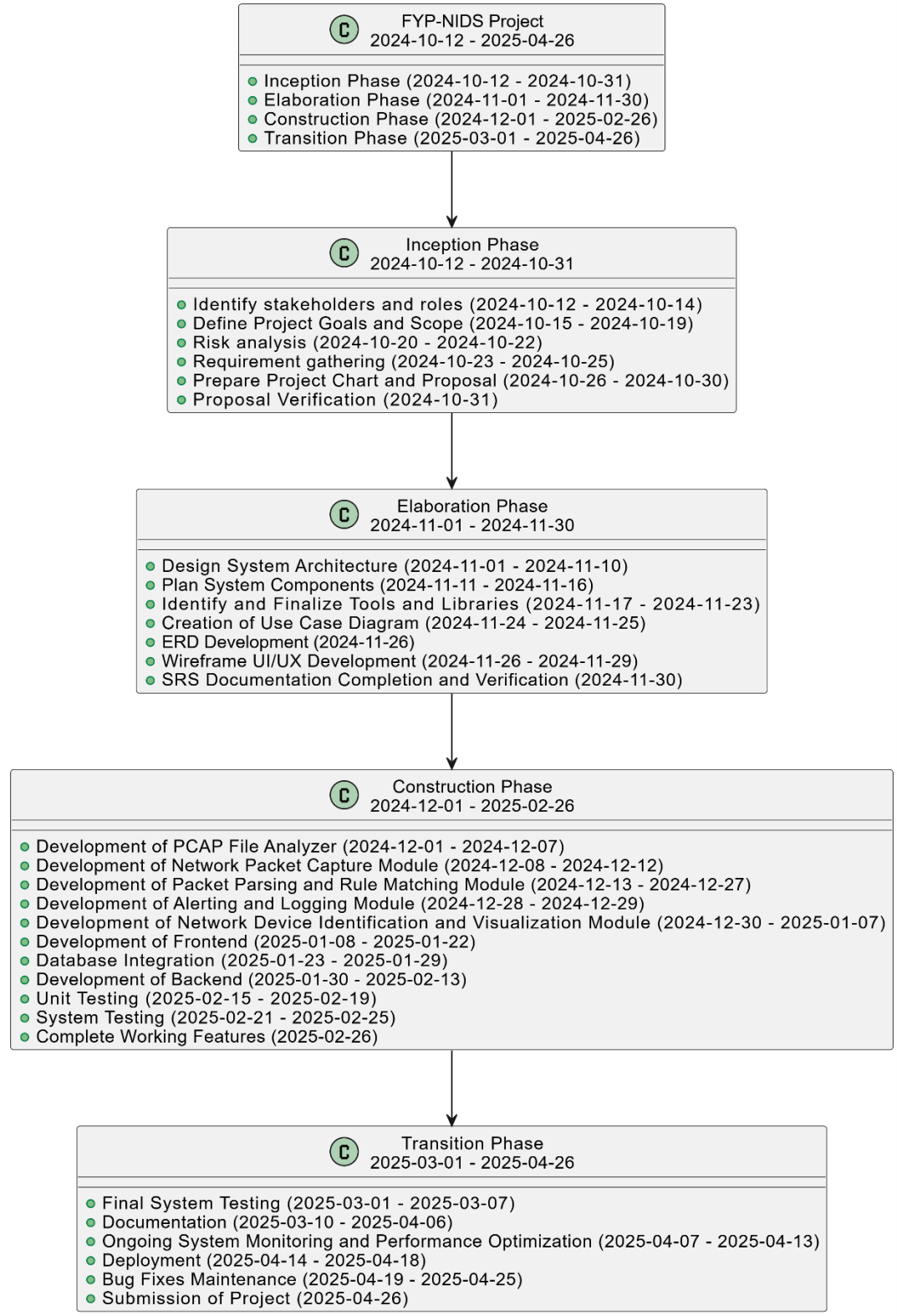


Figure 3 Project Milestone Chart

1. **Inception Phase (2024-10-12 to 2024-10-31)**

Identify Stakeholders and Roles (2024-10-12 to 2024-10-14)

Understand who will be involved in the project and define their responsibilities.

2. Define Project Goals and Scope (2024-10-15 to 2024-10-19)

Establish what the project will achieve and its boundaries to align expectations.

3. Conduct Risk Analysis (2024-10-20 to 2024-10-22)

Identify potential risks and plan mitigation strategies to avoid any issues that may arise during this project.

4. Requirement Gathering (2024-10-23 to 2024-10-25)

Collect detailed requirements from stakeholders to define the project's functionalities.

5. Prepare Project Chart and Proposal (2024-10-26 to 2024-10-30)

Draft the project proposal and chart outlining the timeline and deliverables.

6. Milestone 1: Proposal Verification (2024-10-31)

Verify and approve the project proposal with internal and external supervisor, marking the official start of the project.

2. **Elaboration Phase (2024-11-01 to 2024-11-30)**

7. Design System Architecture (2024-11-01 to 2024-11-10)

Create a high-level architecture for the system to ensure scalability and performance.

8. Plan System Components (2024-11-11 to 2024-11-16)

Define the components and modules needed for system development.

9. Identify and Finalize Tools and Libraries (2024-11-17 to 2024-11-23)

Choose the tools, frameworks, and libraries for development and testing.

10. Create Use Case Diagram (2024-11-24 to 2024-11-25)

Visualize the interaction between the system and its users to capture requirements.

11. Develop Entity-Relationship Diagram (ERD) (2024-11-26)

Design the database schema and relationships between entities.

12. Wireframe UI/UX Development (2024-11-26 to 2024-11-29)

Create mockups of the user interface to guide frontend development.

13. Milestone 2: SRS Documentation Completion and Verification (2024-11-30)

Finalize the Software Requirements Specification (SRS) document to align stakeholders.

3. **Construction Phase (2024-12-01 to 2025-02-26)**

14. Develop PCAP File Analyzer (2024-12-01 to 2024-12-07)

Build a module to analyze packet capture (PCAP) files.

15. Develop Network Packet Capture Module (2024-12-08 to 2024-12-12)

Create functionality to capture live network traffic.

16. Develop Packet Parsing and Rule Matching Module (2024-12-13 to 2024-12-27)

Implement parsing and rule matching to identify malicious packets and either log ignore or send alerts according to the actions specified in the rule.

17. Develop Alerting and Logging Module (2024-12-28 to 2024-12-29)

Build features to generate alerts and log suspicious activities such as sending mails or messages.

18. Develop Network Device Identification and Visualization Module (2024-12-30 to 2025-01-07)

Add functionality to identify network devices and visualize connections.

19. Develop Frontend (2025-01-08 to 2025-01-22)

Design and implement the user interface.

20. Integrate Database (2025-01-23 to 2025-01-29)

Connect the application with the database for seamless data management.

21. Develop Backend (2025-01-30 to 2025-02-13)

Implement server-side logic to handle requests and process data.

22. Unit Testing (2025-02-15 to 2025-02-19)

Test individual components for correctness.

23. System Testing (2025-02-21 to 2025-02-25)

Perform end-to-end testing to ensure system reliability.

24. Milestone 3: Complete Working Features (2025-02-26)

Ensure all features are functional and integrated successfully.

4. **Transition Phase (2025-03-01 to 2025-04-26)**

25. Milestone 4: Final System Testing (2025-03-01 to 2025-03-07)

Conduct rigorous final testing to validate the system's stability and performance.

26. Documentation (2025-03-10 to 2025-04-06)

Prepare user manuals, technical documentation, and project reports.

27. Ongoing System Monitoring and Performance Optimization (2025-04-07 to 2025-04-13)

Monitor the system and optimize performance where necessary.

28. Deployment (2025-04-14 to 2025-04-18)

Deploy the system in the production environment.

29. Bug Fixes and Maintenance (2025-04-19 to 2025-04-25)

Address post-deployment issues and perform maintenance tasks.

30. Milestone 5: Submission of Project (2025-04-26)

Submit the final project, including all deliverables, marking the project's conclusion.

9. Project Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are
- When each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much
- The start and end date of the whole project.

To summarize, a Gantt chart shows you what has to be done (the activities) and when (the schedule). (Duke, 2024)

The project Gantt Chart for my Final Year Project is given below:

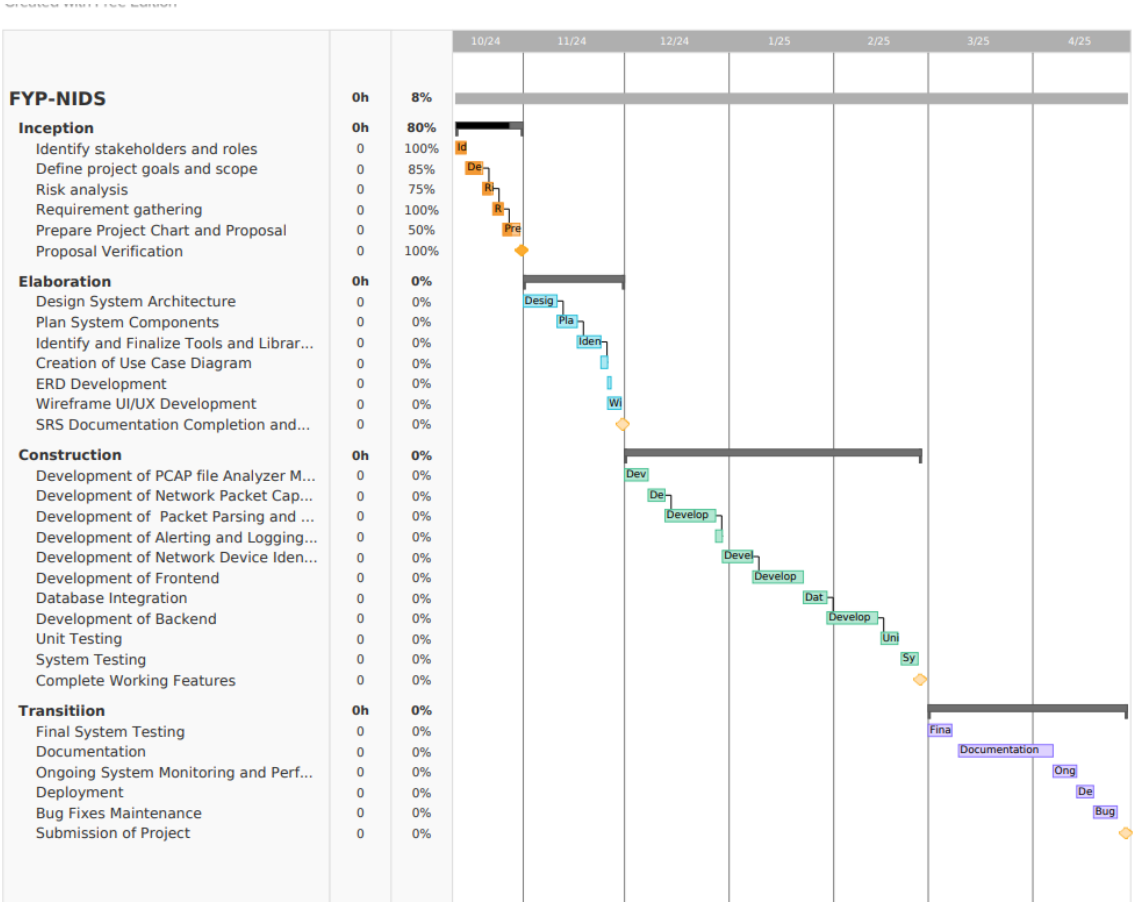


Figure 4 Project Gantt Chart

10. Conclusion

Overall, this project offers a comprehensive, scalable, and accessible solution for network intrusion detection. By combining the power of Snort's rule-based detection system with a user-friendly web dashboard, it addresses the key challenges faced by organizations in securing their networks against evolving threats. This system will not only enhance the accuracy and speed of threat detection but will also improve the overall efficiency of network security management, allowing organizations to stay ahead of cyber threats and better protect their valuable data and infrastructure.

11. References

aws.amazon.com, 2024. *aws.amazon.com.* [Online] Available at: https://aws.amazon.com/what-is/sdlc/ [Accessed 30 November 2024].

Duke, R., 2024. *Gantt.com.* [Online] Available at: https://www.gantt.com/

Jacky, 2023. *sapphire.net.* [Online] Available at: https://www.sapphire.net/blogs-press-releases/nids/

Janse, B., 2024. *Toolshero.* [Online] Available at: https://www.toolshero.com/information-technology/rational-unified-process-rup/ [Accessed 30 November 2024].

Malsam, W., 2024. *www.projectmanager.com.* [Online] Available at: https://www.projectmanager.com/blog/how-to-create-a-milestone-chart#:~:text=A%20milestone%20chart%20is%20a,the%20priority%20of%20the%20milestone. [Accessed 30 11 2024].

projectmanager.com, 2024. *www.projectmanager.com.* [Online] Available at: https://www.projectmanager.com/guides/work-breakdown-structure