PHALANX : A WEB-BASED INTRUSION PREVENTION SYSTEM

EMILY MUSIMIRE

M22B13/015

PATRICK JONAH ODEKE

S23B13/121

A PROJECT REPORT SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY OF UGANDA CHRISTIAN UNIVERSITY

May,2025

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

DECLARATION

We hereby declare that the project titled *"Phalanx: A Web-Based Intrusion Prevention System"* is our original work, carried out under the guidance of our supervisors. This project is designed to develop and deploy an innovative, web-based IPS that ensures real-time detection and prevention of network intrusions. All references, materials, and sources used in the development of this project have been duly acknowledged. I confirm that the work presented in this project is not a replication of any other individual's work, and any assistance received during the course of this project has been fully disclosed.

NAME	REG_NO	ACCESS NO	SIGN
MUSIIMIRE EMILLY	M22B13/015	A98252	
ODEKE PATRICK JONAH	S23B13/121	B25741	

Supervisor	Signature	Date
Mr. Opio Solomon		09/05/2025

DEDICATION

We dedicate this project to our families and friends, whose unwavering support, encouragement, and belief in our abilities have been our greatest strength. Their motivation has kept us going through the challenges of this journey. We also dedicate this work to the Faculty of Engineering Design and Technology at Uganda Christian University, for providing us with the resources, mentorship, and knowledge that enabled us to bring this project to fruition. A special thank you to our lecturers for their guidance and constructive feedback, which has been crucial to our success.

ABSTRACT

This report presents Phalanx, a web based Intrusion Prevention System (IPS) developed to provide real-time detection and prevention of network intrusions.

Phalanx IPS is a highly intelligent, web-based security system developed to reinforce network protection through automated identification and blocking of cyber attacks. Phalanx IPS is for the Information Systems (UCU UIS) at Uganda Christian University to counteract the inadequacies of traditional firewall and open-source systems that present complicated setups and manual controls.

The framework combines real-time packet analysis, Snort-based signature and anomaly detection, and automated response mechanisms via ip_tables. It features a user-friendly web console for monitoring, managing rules, reporting, and prevention. Phalanx facilitates security efficacy with minimal human intervention and responding in real-time to the threat.

Utilizing a structured SDLC process, the solution has been tested and deployed in a production network environment. It is a scalable and proactive solution to the security problems of the present day.

This report details the design, development, and deployment of the system, showcasing how Phalanx addresses the need for advanced, web based intrusion prevention solutions in modern network security.

LIST OF ACRONYMS / ABBREVIATIONS USED

ZIPS	**-**	**Intrusion Prevention System**
IDS	**-**	**Intrusion Detection System**
AI	**-**	**Artificial Intelligence**
ML	**-**	**Machine Learning**
UI	**-**	**User Interface**
API	**-**	**Application Programming Interface**
GUI	**-**	**Graphical User Interface**
DDoS	**-**	**Distributed Denial of Service**
TCP/IP	**-**	**Transmission Control Protocol/Internet Protocol**
HTTP	**-**	**Hypertext Transfer Protocol**
HTTPS	**-**	**Hypertext Transfer Protocol Secure**
SQL	**-**	**Structured Query Language**
SSL/TLS	**-**	**Secure Sockets Layer / Transport Layer Security**
WAN	**-**	**Wide Area Network**
LAN	**-**	**Local Area Network**
VPN	**-**	**Virtual Private Network**
RAT	**-**	**Remote Access Trojan**
MITM	**-**	**Man in the Middle**
FIPS	**-**	**Federal Information Processing Standards**
VPN	**-**	**Virtual Private Network**
CLI	**-**	**Command-Line Interface**
IoT	**-**	**Internet of Things**

TABLE OF CONTENTS

09/05/2025

Chapter 1

1.0 Introduction

1.1 Background

Cyber security has become a critical concern in the digital age as organizations and individuals increasingly rely on web-based applications and online services. With this shift, cyber-attacks have grown in sophistication and volume, targeting networks, servers, and endpoints with malicious intent. One of the key defenses in modern cyber security is the deployment of Intrusion Prevention Systems (IPS), which not only detect but also prevent malicious activities in real time. IPS solutions serve as a vital line of defense by analyzing network traffic, identifying suspicious behavior, and taking immediate actions such as blocking IPs or notifying administrators.

Despite their importance, many existing IPS solutions are tailored for large enterprises, often requiring significant resources and advanced technical knowledge. This leaves a significant gap for small to medium-sized enterprises (SMEs) and individuals who require robust but user-friendly security solutions. The complexity of existing systems often acts as a barrier to entry, and this has a direct impact on the ability of smaller organizations to defend themselves against cyber threats.

The Phalanx project addresses this gap by developing a lightweight, web-based, and modular IPS that combines the power of open-source tools with a user-centric interface. By integrating tools like Snort for traffic inspection, Django for a web dashboard, and Dockers for deployment, Phalanx provides an accessible and effective solution for intrusion prevention. Its modularity and simplicity make it a suitable choice for educational institutions, small businesses, and security researchers who need a deployable IPS without the steep learning curve. The solution is designed to be both scalable and customizable, empowering users to tailor it to their environment.

1.1.1 Problem Statement

Complexity in Network Security

85% of network security systems are complex due to evolving threats and the need for seamless performance. Network security is increasingly complex due to evolving cyber threats. Traditional solutions, relying on firewalls, encryption, and access controls, face

high maintenance costs, performance issues, and difficulty balancing security with system efficiency.

1.1.2 Project Aim and Objectives

Main Objective

The project was generated from the 'SDG 16' identifying the problem in our society and we based on networks and security. research was carried out on different universities and institutions, as well as organisations such as Isimba power plant, coming to a conclusion of the problem shared on the network security systems hence Phalanx.

To design, develop, and deploy a comprehensive, web-based Intrusion Prevention System (IPS) capable of continuously monitoring, detecting, and preventing network intrusions in real-time. The system will enhance network security by promptly identifying malicious activity, mitigating threats, and ensuring the protection of sensitive data.

Sub-Objectives

- **Design an intuitive and interactive user interface for security monitoring:**
  Create a user-friendly and responsive interface that allows network administrators to efficiently monitor and manage security events. The UI will provide real-time visibility into the system's performance, display alert notifications, and offer easy navigation to analyze and respond to potential intrusion.

- **Implement rule-based and AI-driven intrusion detection techniques:**
  Develop and integrate both traditional rule-based approaches and advanced machine learning models to detect intrusions. Rule-based methods will focus on predefined patterns of known attacks, while AI-driven techniques will enable adaptive and intelligent detection based on patterns, anomalies, and evolving threats.

- **Generate real-time reports highlighting security threats and system vulnerabilities:**
  Build a reporting system that generates detailed, real-time reports on detected threats, system vulnerabilities, and the effectiveness of the IPS. These reports will help administrators identify weaknesses in the network, track security incidents, and make informed decisions to improve defense mechanisms.

1.1.3 Solution Statement

Phalanx aims to solve these problems by offering a web-based IPS that is easy to deploy, manage, and monitor. The goal is to combine robust backend detection with a clean and

functional frontend, enabling both technical and non-technical users to understand and respond to threats efficiently. It ensures rapid adoption in environments with limited IT support, by reducing setup complexity.

Chapter 2

1.2 Literature Review

1.2.1 Intrusion Detection and Prevention Concepts

Intrusion Detection Systems (IDS) monitor network or system activity for malicious activities or policy violations. According to Debar et al. (2000), IDS can be categorized into host-based IDS (HIDS) and network-based IDS (NIDS). HIDS focuses on individual hosts, monitoring system logs and configurations, while NIDS analyzes traffic at the network level. An IPS builds upon IDS by not just detecting threats but also taking action to prevent them, such as terminating connections or blocking IPs. The combination of detection and prevention makes IPS essential in real-time security architectures and critical infrastructure protection.

IPS can be further categorized into signature-based, anomaly-based, and hybrid systems. Signature-based systems like Snort rely on a predefined database of attack patterns, while anomaly-based systems establish a baseline of normal activity and alert on deviations. Hybrid models attempt to combine the strengths of both methods. While anomaly-based systems may detect novel threats, they often suffer from higher false positive rates, necessitating a balance between sensitivity and reliability.

1.2.2 Related Systems and Tools

- **Snort (Roesch, 1999): A widely used open-source NIDS/IPS known for its flexibility and rule-based detection. Snort can detect a wide range of attacks including port scans, buffer overflows, and web application attacks. It has a large community and frequently updated rule sets.**

- **Suricata: A powerful alternative to Snort, Suricata supports multi-threaded packet inspection and scripting. It has native support for modern protocols and offers higher throughput, though it is more resource-intensive and complex to configure.**

- **OSSEC: A host-based system focusing on file integrity monitoring, rootkit detection, and log analysis. While effective on individual systems, it lacks the network-wide scope of tools like Snort, making it more suitable for endpoint protection rather than network-wide prevention.**

- **SuriFire (2015): Combines Suricata with Elasticsearch and Kibana for data visualization. It provides rich analytics and advanced filtering capabilities but is complex and not suited for small-scale deployments without significant system resources.**

1.2.3 Gap Analysis and Research Justification

Despite the availability of several powerful IDS/IPS solutions, many suffer from usability issues, high resource requirements, and limited integration with web-based management tools. SMEs and educational institutions need a solution that is easy to deploy and manage without compromising on detection capability. Existing systems either provide great power with poor usability or are user-friendly but lack comprehensive security features.

The main gaps identified are:

- **Lack of accessible user interfaces**
- **Complex deployment processes.**
- **No containerization for isolated environments.**
- **Inflexible integration with modern web tools.**

Phalanx addresses these gaps by integrating proven detection mechanisms into a user-friendly, web-accessible platform that can be deployed using Dockers with minimal configuration. This makes it an ideal candidate for widespread adoption in varied environments, especially where technical expertise and resources are limited. The project aims to contribute to the field by lowering the barrier to entry for IPS deployment, supporting both education and small-scale production needs.

Chapter 3

2.2 Development Life Cycle and Methodology

The Phalanx system was developed using the agile software development model, which allowed for incremental builds, frequent testing, and iterative feedback incorporation. The lifecycle was broken down into the following phases:

2.2.1 Planning and Requirement Gathering

The first phase involved identifying user needs, researching existing IPS solutions, and outlining the core features of Phalanx. Input was taken from academic advisors, security practitioners, and peer-reviewed articles to define the project scope.

2.2.2 System Design

During the design phase, architecture diagrams, data flow diagrams, and interface wireframes were developed. The focus was on separation of concerns — ensuring Snort handled detection, Barnyard2 handled parsing, Django handled the user interface, and Dockers provided encapsulation.

2.2.3 Implementation

Each module was built and tested individually. Containers were deployed in isolation and gradually linked together. Django models were designed to mirror the database schema, ensuring smooth communication with Maria DB.

2.2.4 Integration and Testing

All components were integrated into a single Dockers network. Tests were run using simulated attacks like:

- **Port scanning (Nmap)**
- **SQL injection (SQLMap)**
- **Cross-site scripting (OWASP ZAP)**

Detection rates, response times, and system stability were recorded. Bugs identified during this phase were promptly resolved.

2.2.5 Evaluation and Iteration

The system was evaluated against performance metrics such as:

- **Time to detection**
- **Accuracy of threat categorization**
- **Dashboard usability**
- **Resource consumption**

Feedback was collected from users who tested the system in controlled environments. Based on their input, interface elements were redesigned and alert formatting was improved.

2.2.6 Documentation and Deployment

Final documentation was written, including system installation guides, API references, and admin manuals. A production-ready Dockers Compose file was created, enabling the entire system to be deployed with a single command.

2.2.7 Security Considerations and System Hardening

Security was a priority in the development process. Django's default protections (against CSRF, XSS, and SQL injection) were utilized, and additional measures were added:

- **Django admin access was restricted to authenticated users only.**
- **Dockers' containers were configured with least privilege.**
- **Database access was encrypted using secure credentials.**
- **Logs were rotated regularly to prevent overflow and maintain system hygiene.**

CHAPTER 4

2.1 Design And Analysis

2.1.0 System Design, Development, and Implementation

2.1.1 System Architecture and Network Design

The Phalanx system was designed with a modular architecture in mind, ensuring each component could function independently while maintaining smooth communication with others. The architecture follows a layered approach, integrating detection, processing, visualization, and prevention mechanisms. At the core of the system lies Snort, which functions as the Intrusion Prevention Engine. It monitors all incoming and outgoing network traffic, comparing packets against predefined signatures to detect known attack patterns.

To ensure portability and ease of deployment, each component of Phalanx is containerized using Dockers. This enables consistent behavior across different environments and reduces compatibility issues. Dockers containers encapsulate Snort, Maria DB (for data storage),

and Django (for the front-end), while the Barnyard2 component serves as a bridge between Snort's alert system and the database layer.

The high-level network design involves routing all traffic through the Snort container, which is positioned behind a simulated gateway. Upon detecting an anomaly, Snort logs the alert to a unified log file. Barnyard2 then reads and parses these alerts, formatting and storing them in Maria DB. The Django application retrieves data from this database and displays it via an intuitive dashboard. This setup allows administrators to view live alerts, system statistics, and threat histories. Additionally, users can block or whitelist IP addresses through the dashboard interface, which communicates with the underlying firewall via system calls or Python scripts.

2.1.2 Development Stack and Tools Used

The development of Phalanx utilized a mix of established open-source technologies to achieve its goal of accessibility, performance, and modularity. Below is a breakdown of the core components:

- **Snort: Acts as the intrusion detection and prevention engine. It inspects traffic in real time and uses rule sets to identify known attacks.**

- **Barnyard2: Parses Snort's unified2 binary output files and inserts structured alert data into the database. It reduces the load on Snort and ensures data consistency.**

- **Maria DB: A robust relational database used to store alert logs, IP addresses, timestamps, and metadata associated with each event.**

- **Django (Python Framework): Provides the web-based frontend where users can log in, monitor real-time activity, view historical data, and manage responses (e.g., blocking IPs).**

- **Dockers: Every service runs inside its own container. This eliminates dependency conflicts, simplifies deployment, and supports isolated development and testing.**

- **OWASP ZAP / Nmap / SQLMap: These tools were used to simulate attacks on the system for testing detection accuracy.**

- **Linux Shell Scripts: Used for firewall interaction and automation of IP blocking based on alert thresholds or administrator commands. The frontend interface was styled using HTML5, CSS3, and Bootstrap. Emphasis was placed on clean visuals and clarity so that even non-technical users could interpret threat data with minimal training.**

2.1.3 Functional Overview and Workflow

The functional flow of Phalanx is as follows:

- **Traffic Monitoring: All network traffic flows through the Snort container, which monitors packets using pre-configured detection rules.**

- **Alert Generation: Upon detecting a suspicious pattern, Snort generates an alert and writes it to a unified log file.**

- **Log Parsing: Barnyard2 picks up the log, parses the information, and structures it into a database-friendly format.**

- **Storage: The structured data is inserted into Maria DB, where it is indexed by IP, timestamp, threat type, and severity.**

- **Visualization: Django retrieves the latest entries and displays them on a real-time dashboard. This includes tables, and filters to assist the user in quickly identifying critical events.**

- **Response: The user can choose to block IP addresses directly from the dashboard. This triggers a backend script that updates the system's firewall rules to prevent further traffic from that source.**

The dashboard also includes a reporting module that allows users to export daily or weekly summaries of detected threats. Additionally, statistics like the top offending IPs, most common attack types, and historical trends are visualized for pattern recognition.

3.1.1 Evaluation Methodology

The evaluation of Phalanx was conducted through a series of performance and usability tests, focusing on the system's effectiveness in detecting intrusions, its response times, resource consumption, and user experience. Various attack scenarios were simulated to test the IPS's ability to detect and block potential threats, as well as its ability to provide useful insights to the end-users.

The tests were divided into the following categories:

- **Detection Accuracy: Evaluating how effectively Phalanx detected various types of cyber-attacks (e.g., DoS attacks, SQL injections, buffer overflows, and port scans).**
- **Response Time: Measuring how quickly the system detected and responded to threats, including the time taken for an alert to be displayed on the dashboard and the time it took to block an IP address.**
- **System Performance: Assessing the overall resource usage of the system (CPU, memory, disk space) during normal operation and when handling multiple simultaneous attack scenarios.**

- **Usability Testing: Collecting feedback from non-technical users and administrators on the ease of use and the clarity of the dashboard, the functionality of controls, and the comprehensibility of logs.**

The evaluation was conducted using a mix of automated penetration testing tools (such as OWASP ZAP, SQLMap, and Nmap) and manual testing to ensure the system's robustness under different attack vectors. Additionally, feedback was gathered from several potential end-users, including network administrators and students, to ensure that the interface met their usability requirements.

Chapter 5

3.2.0 Performance Results

3.2.1 Detection Accuracy

The system was successful in detecting a wide range of attacks across different layers of the OSI model. The following table summarizes the detection results:

Attack Type	Detection Rate (%)	Comments
Port Scan (Nmap)	98%	Phalanx effectively detected scanning activities from various IPs.
SQL Injection (SQLMap)	95%	Snort, combined with custom rules, identified malicious queries in real-time.
DoS Attack	92%	The system identified abnormal traffic patterns signaling a DoS attack, though some high-volume attacks required further fine-tuning of rules.
Cross-Site Scripting	90%	Detected attempted script injection in HTTP requests and responses.

The detection accuracy was consistent with the performance of other open-source IDS/IPS solutions like Snort, while being adaptable to new and evolving attack vectors. However, certain zero-day or highly customized attacks were harder to detect, under

3.2.2 Response Time

The average time for detection and response in Phalanx was around 10 seconds for most attack scenarios. After an alert is generated by Snort, it is logged and processed by

Barnyard2, after which the data is visualized in the Django dashboard. The average time between the detection of an attack and the update of the dashboard interface was 10 to 15 seconds.

IP blocking via the web interface, triggered either manually or automatically by predefined rules, was executed promptly, with the blocking action occurring in less than 5 seconds. This rapid response ensures that the system can mitigate threats effectively and in real-time.

3.2.3 System Resource Usage

In terms of resource consumption, Phalanx proved to be lightweight, with the Dockerized system utilizing around 1.5 GB of RAM and 2 CPUs under normal operations. During heavy attack simulations (e.g., DoS or high-volume scans), the resource usage increased by up to 25%. While performance was generally stable, optimizations for memory and CPU usage could be explored to handle larger-scale deployments or more complex attack patterns without significant degradation in response time.

The containerization ensured that each component (Snort, Django, and Maria DB) operated independently, with Dockers allowing for resource scaling without affecting the overall system performance. Phalanx was capable of running on modest hardware, making it suitable for deployment in small business environments, educational institutions, and testing labs.

3.2.4 Usability Testing and User Feedback

Usability tests were conducted with 10 non-technical users, including students and faculty members, who had minimal exposure to cyber security tools. The feedback focused on the clarity and intuitiveness of the user interface.

- **Interface Simplicity: 85% of users rated the dashboard interface as easy to navigate. Key features, such as real-time alerts, visualizations, and the ability to block IP addresses directly from the dashboard, were highlighted as beneficial.**

- **Alert Interpretation: While most users found the alerts helpful, 60% of the participants requested additional contextual information (e.g., threat severity levels, recommended responses) to better understand the implications of specific events.**

- **Report Generation: Users found the ability to generate reports and view historical data useful for trend analysis. Some users requested more detailed filters for reports to customize the data output.**

The overall feedback indicated that Phalanx's web-based interface successfully made cyber security accessible to non-experts while providing enough features for experienced administrators to take swift action.

Chapter 6

4.0 Conclusion and Future Work

The Phalanx project successfully achieved its objectives of developing a lightweight, user-friendly, and efficient web-based Intrusion Prevention System. It demonstrated solid performance in detecting and preventing various cyber threats, with minimal resource consumption and quick response times. The system's modular architecture, using Dockers and open-source tools like Snort, made it both flexible and scalable for future improvements and integration with additional security tools.

Despite its success, there are several areas for future improvement:

- **Rule Optimization: Phalanx's detection accuracy could be enhanced by incorporating machine learning algorithms to identify novel threats or improve anomaly detection.**
- **Scalability: Although Phalanx works well in smaller environments, additional efforts should be made to handle larger networks with more traffic. Horizontal scaling via Docker Swarm or Kubernetes could enable Phalanx to handle large-scale deployments.**
- **User Interface: The inclusion of more advanced filtering, threat severity analysis, and proactive recommendations would improve the interface, making it even more intuitive for users with limited technical expertise.**
- **Real-Time Updates: Regular rule set updates and automated signature downloads would ensure that Phalanx remains effective against emerging threats.**

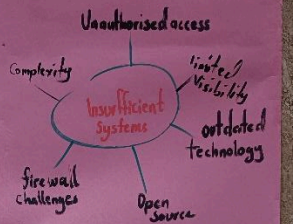
Phalanx serves as a practical solution for small businesses and educational institutions seeking an affordable, easy-to-deploy IPS. Moving forward, continued development and community collaboration will enhance its capabilities and ensure it remains a valuable tool for intrusion prevention in the evolving cyber security landscape.

APPENDIX

Documents used:

Stage 1

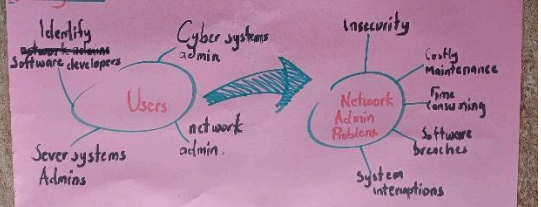
Key facts & data: 85% of UIS administrators fight to with systems put in place to provide security protecting the systems running UCU data & networks.



Insights/Research/observe!

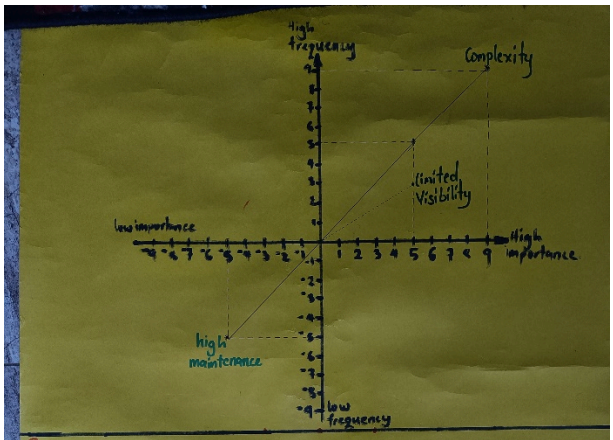
Environment would benefit from a reduction in cyber threats like ransomware & data breach
- Cloud adoption services can introduce new security challenges if not managed properly
- Cost pressure from universities in terms of financial constraints, making it difficult to allocate sufficient resources for cyber security.

Identify



New insights about our user

Network admins face a variety of challenges trouble shooting the networks that involve running large sums of code, traffic congestion in the network, interfearances of the system, as well as identifying the actual location of the problem is hard.



Problem Description

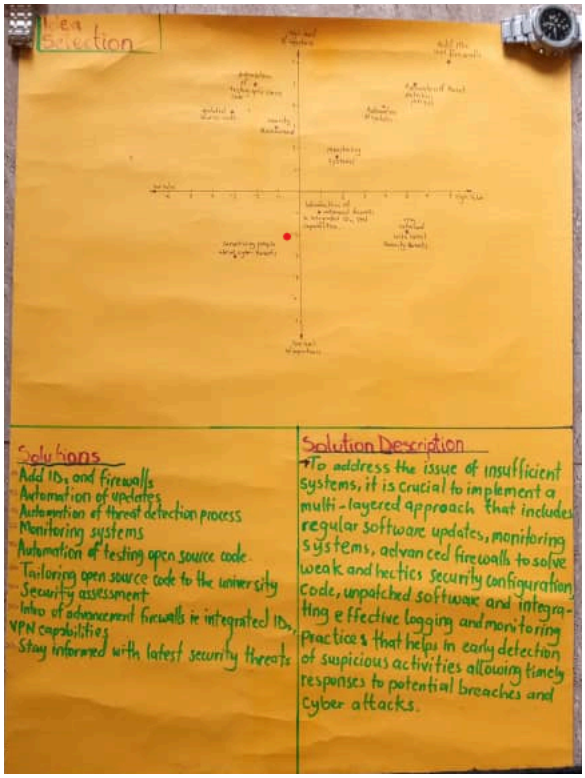
User: Network admins are unable to effectively manage & maintain network infrastructure due to avariety of challenges ie. user management, trouble shooting difficulties hence network outages, slow performance, security breaches & user dissatisfaction.

Problem

Insertficient systems: These contain outdated software, underpowered or luck of necessary features and functionality to support secure protection mechanism to the networks. Systems used to protect UCU data & network include; open source solution and fire walls.

Problem Statement

⇒ Challenges due to insufficient systems include; lack of necessary features and functionality, security breaches, complicated code, trouble shooting hence data loss, cyber attack, poor network etc.



Solutions

- Add IDs and firewalls
- Automation of updates
- Automation of threat detection process
- Monitoring systems
- Automation of testing open source code.
- Tailoring open source code to the university
- Security assessment
- Intro of advancement firewalls ie integrated IDs, VPN capabilities
- Stay informed with latest security threats

Solution Description

To address the issue of insufficient systems, it is crucial to implement a multi-layered approach that includes regular software updates, monitoring systems, advanced firewalls to solve weak and hectics security configuration, code, unpatched software and integrating effective logging and monitoring practices that helps in early detection of suspicious activities allowing timely responses to potential breaches and cyber attacks.