IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Pasidu Adithya Liyanage
Student Reference Number: 10899322

Module Code: PUSL3190	Module Name: Computing project
Coursework Title: The Basic Security Test Application	
Deadline Date: 5th of MAY 2025	Member of staff responsible for coursework: Dr. Pabudi Abeyrathne
Programme: BSc (Hons) Computer Security	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed: 

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

Overall mark _____% Assessors Initials _____ Date_____

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

UNIVERSITY OF PLYMOUTH

PUSL3190 Computing Individual Project

Final Project Report

The Basic Security Test Application

Supervisor: Dr. Pabudi Abeyrathne

Name: Pasidu A Liyanage

Plymouth Index Number: 10899322

Degree Program: Bsc(Hons) Computer Security

Acknowledgement

I am deeply grateful to all those who provided support and guidance during the development of this project.

I am deeply grateful to my project supervisor, Dr. Pabudi Abeyaratne, for his invaluable insight and expert guidance that determined the direction of this project. The foundational knowledge and inspiration for this project emerged from the BSc (Hons) Computer Security academic staff at the University of Plymouth.

I express my sincere gratitude to my supportive family members and dedicated friends who were with me unconditionally during the development of my research project. I express my gratitude to the open-source community along with the developers who created the core tools dramatically improved the technical implementation of the project. I express my sincere gratitude to everyone who supported me throughout the process, and I gained valuable insights from this project.

I am deeply appreciative of your kindness and love. Thank you all for being a part of this achievement.

Abstract

The Basic Security Auditing Application assists IT auditors in performing automated vulnerability assessments of systems and networks. The application combines a user-friendly front-end built with ReactJS and a robust Python-based back end, enabling efficient scanning, real-time logging, and the generation of detailed PDF audit reports. The system is agent-based, running on the auditor's side.

With the increasing complexity of IT infrastructure, traditional security audit processes are often slow, error-prone, and require specialized knowledge. Many tools on the market are powerful but overly complex for less technically advanced users. This project addresses the need for a simple yet effective tool that performs multiple automated scans, validates system integrity, and provides documentation of audit results.

The motivation behind this project stems from the challenges faced by IT auditors, such as steep learning curves, manual reporting processes, and delayed vulnerability detection in existing tools. There is a growing demand for solutions that are fast, intuitive, and capable of generating reliable security assessments without excessive manual intervention.

The frontend is developed using ReactJS, allowing auditors to enter a server URL and audit the code, initiate scans, and view scan results. The backend is developed using Python with FastAPI to integrate security tools and generate reports using FPDF. As a benefit to the client, the PDF report can be downloaded to the client side.

As the scanning process is underway, the auditor enters their code and server URL, the system checks user validation, detects the operating system, checks for open ports, outdated software, and misconfigurations.

Finally, after scanning, a detailed PDF report with vulnerability details and mitigation recommendations is automatically generated.

The final results show automated system and network scans, Real-time logging and professional PDF report generation, simple UI with minimal training required for auditors, successfully tested in simulated network environments. In the future, the application can be expanded to support cloud-based deployments for central management, real-time alerting, and notification systems. Enhanced report customization with charts, graphs, etc.

Ultimately, the basic security testing application fulfills the practical needs of security tools and IT auditors.

Table of Contents

List of Figures

1. Introduction

In today's rapidly changing digital environment, ensuring the security of information systems is essential. As cyber threats continue to evolve, the need for effective, timely, and accurate security audits has never been more critical. IT auditors play a vital role in protecting digital assets by conducting risk assessments and identifying vulnerabilities in both systems and networks. However, traditional security audit processes often involve complex tools, time-consuming procedures, and a high potential for human error, especially in environments that require the evaluation of large or diverse infrastructures.

The project aims to provide a functional solution to infrastructure security challenges by developing a basic security testing application. The application aims to transform the security audit assessment by IT auditors into an automated process that reduces technical constraints when assessing security audits. The system establishes agent-based functionality that is activated from the auditor's computer to authenticate users while managing backend server communication to launch scans. By combining modern web technologies with established security scanning tools, the application provides a streamlined, user-friendly interface for executing multiple security tests and generating detailed, automated reports.

The system is built using a front-end, providing auditors with a simple interface for interacting with the tool. System and network scanning is largely done using Python and the FastAPI-developed back-end component. The application performs vulnerability scanning through the use of the Nmap library, which detects open ports. In real time, the system logs each scan result before generating PDF reports using the FPDF library from Python.

The development of this application stems primarily from the wide access disparity between security tools and their intended end users. Industry-standard tools OpenVAS and Wireshark and Nmap require extensive technical knowledge through command-line interface operations that most auditors can handle. The Basic Security Testing application solves this problem by automating basic audit tasks while providing a user-friendly interface throughout the system.

Ultimately, this project aims to improve audit workflows by reducing the time, effort, and technical expertise required to conduct a complete risk assessment. Through its automated and user-friendly system, the Basic Security Testing application enables IT auditors to focus on tool configuration and reporting.

2. Background

In today's interconnected digital landscape, cybersecurity has become a critical issue for organizations of all sizes. Security auditing tools need to deliver effective solutions at high efficiency because sophisticated and more frequent cyberattacks have reached a critical level. IT audit firms defend enterprise systems through their identification of weaknesses that might transform into data breaches and service interruptions, or damage organizational reputation. The current assessment systems face hurdles due to their user-unfriendly interface, slow functions, and manual processes, which pose problems for both expert and less technology-oriented auditors.

The conventional procedure of security auditing requires manual execution because it takes an extended period and produces human mistakes. The deep scanning abilities of Nmap and OpenVAS require advanced knowledge that makes these tools problematic for novice users attempting to operate them effectively. The necessary configuration requirements, along with complex output interpretation and command line expertise make these tools negatively impact the audit speed and duration.

The basic security testing application exists to provide a solution for this deficiency. IT audit firms faced three main difficulties, which initiated the development of this project: they needed shorter audit durations and better testing consistency while seeking easier-to-use applications. A terminal-based application represents the proposed solution because it emphasizes testing the two essential cybersecurity pillars.

Network scanning and system scanning/security audits.

The Network Scanning module embeds ping tests together with port scanning features and DNS lookups, followed by packet examination and active connection monitoring tools to deliver complete network environment information. The system scan module performs checks for firewall conditions while analysing SUID/SGID files and system user accounts, besides examining open ports and installed packages, SSH configuration, and basic vulnerability findings. The combined method gives security teams a complete understanding of network and host security threats. The main innovation of this project centres around its automated processes linked to structured report creation. Once the scans are complete, the system automatically compiles a structured report highlighting the identified vulnerabilities, categorizing them by severity, and offering actionable remediation steps. This not only speeds up the audit process but also ensures consistent and professional documentation for server handover.

By providing a streamlined interface and removing unnecessary complexity, the tool can be effectively used by IT auditors with minimal training, expanding accessibility without compromising depth. The goal is to enable organizations to conduct faster, more accurate audits - thereby reducing security risks and supporting compliance efforts.

The project contains key functions to accelerate cybersecurity auditing operations while providing better ease of use and decreased error potential. The features of the Basic Security Testing application bring important benefits to IT auditing and security assessment practices.

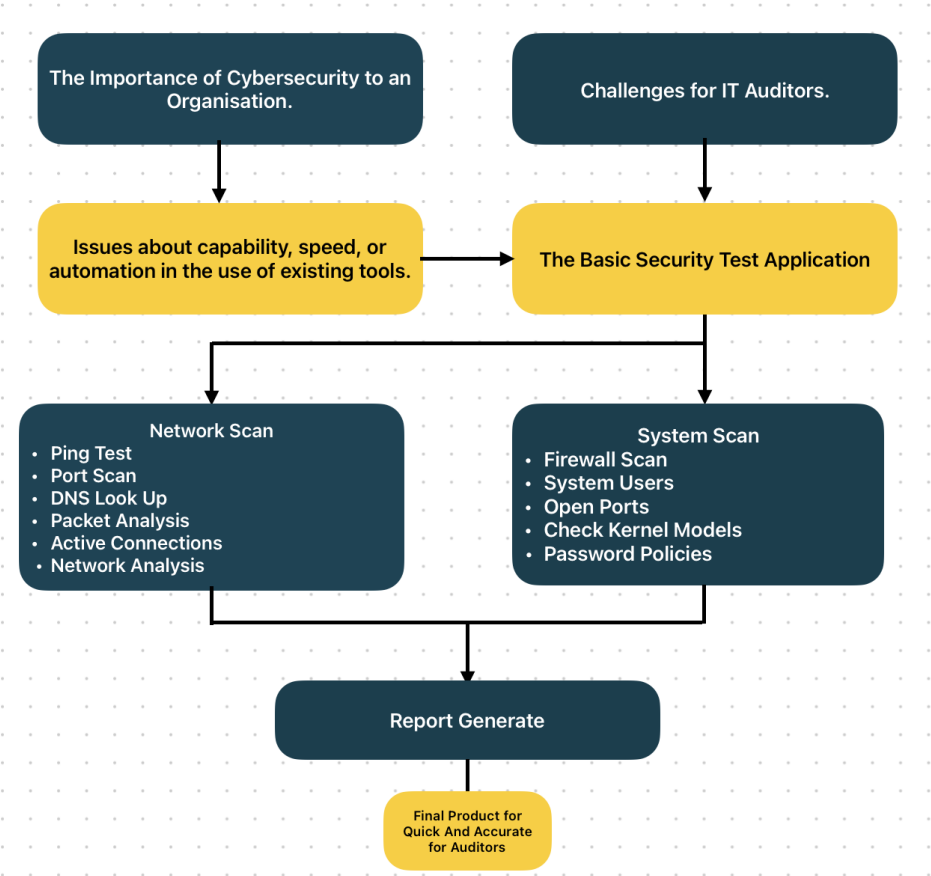


Figure 1 Workflow Overview of the Basic Security Test Application Development Process

Common Challenges and Best Practices in IT Security Auditing,

- Challenge 1: Defining a clear audit scope is crucial—too narrow, and key risks may be missed; too broad, and it can strain resources.

  Pro Tip: Collaborate with auditors to tailor the scope to your organization's specific systems, data, and risk profile.
- Challenge 2: Ensure Staff Participation

  Lack of involvement from key IT staff can delay audits and lead to incomplete or inaccurate results.

  Pro Tip: Involve all relevant stakeholders—like IT security, system admins, and department heads—early and throughout the audit.
- Challenge 3: Using ad hoc methods instead of standardized frameworks like CIS or NIST can undermine audit credibility, miss key performance areas, and hinder comparability.

  Pro Tip: Follow established frameworks to ensure structured, credible, and best-practice-aligned audits.
- Challenge 4: Lack of adequate documentation can hinder tracking trends, remediation, and incident response planning.

  Pro Tip: Keep detailed records of security policies, configurations, and incidents to simplify audits and follow-ups.
- Challenge 5: Prioritize remediation efforts: Focus on fixing the most critical risks first by evaluating their severity, impact, and ease of remediation. This prevents wasting resources on minor issues and strengthens your overall security posture.

Figure 2 common challenges in security Audit

3. Objectives

3.1 Primary Objectives

- Develop and introduce an automated security testing application for IT auditors:
  By automating the procedure, the Basic Security Test Application enables auditors to complete audits in a fraction of the time while maintaining accuracy.
  Automation will ensure that all critical sections of the system are properly checked for flaws, lowering the possibility of human mistakes and detecting vulnerabilities as soon as feasible.

- Improve the efficiency and accuracy of vulnerability scanning:
  Another key goal of the project is to enable IT auditors to execute various security tests simultaneously. The tool's concurrent testing feature will increase audit efficiency and assist IT auditors in meeting tight deadlines.

- The basic security testing application is designed to divide the security auditing process into two main types: network scanning and system scanning.
  The network scanning module focuses on evaluating the external exposure and connection status of the system. Key Features are Ping Test, Port Scanning, DNS Lookup, Packet Analysis, Active Connection Monitoring.
  The system scanning module evaluates the internal state of the system, focusing on configurations, access control, and privilege escalation or data breaches.
  Key Features are Firewall Status, Open Ports, SSH Configuration Check, SUID/SGID File Detection, System User Accounts.

- Generate detailed, automated security reports:
  The goal is to ensure that the application generates detailed, automatic reports following each audit. These reports will summarize the vulnerabilities discovered, prioritize them by severity, and make actionable suggestions for remedy. By automating this procedure, the technology will save auditors time while also ensuring that clients receive clear, professional reports following each audit.

4. Deliverables

The deliverable of this project is an application that contains all the necessary features that are included here.

4.1 Interfaces

1. Register
   User interface - This is the register page. Here, the users who wish to register on this system can enter their name, Email, email, Team, and Team Manager. Then users will be able to register a unique **Auditor Code** is displayed in green. Here, the account will be created for a specific IT Auditor. After you register, Auditor Code generates as **8A2AA90B.**
   Technology - Here the React has been used. likely, due to the localhost:3000, which is common for React dev servers. Simple state management is used with React Hook Form to handle forms. Likely uses **npm** package management development tool.

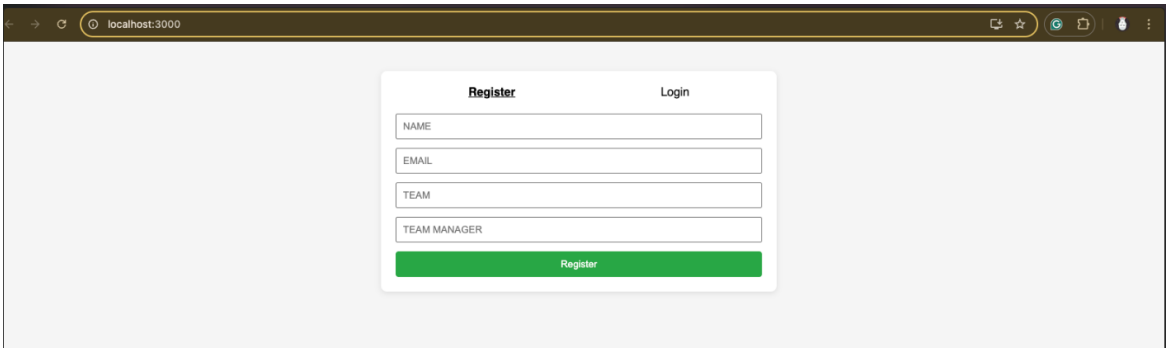


Figure 3 Registration page

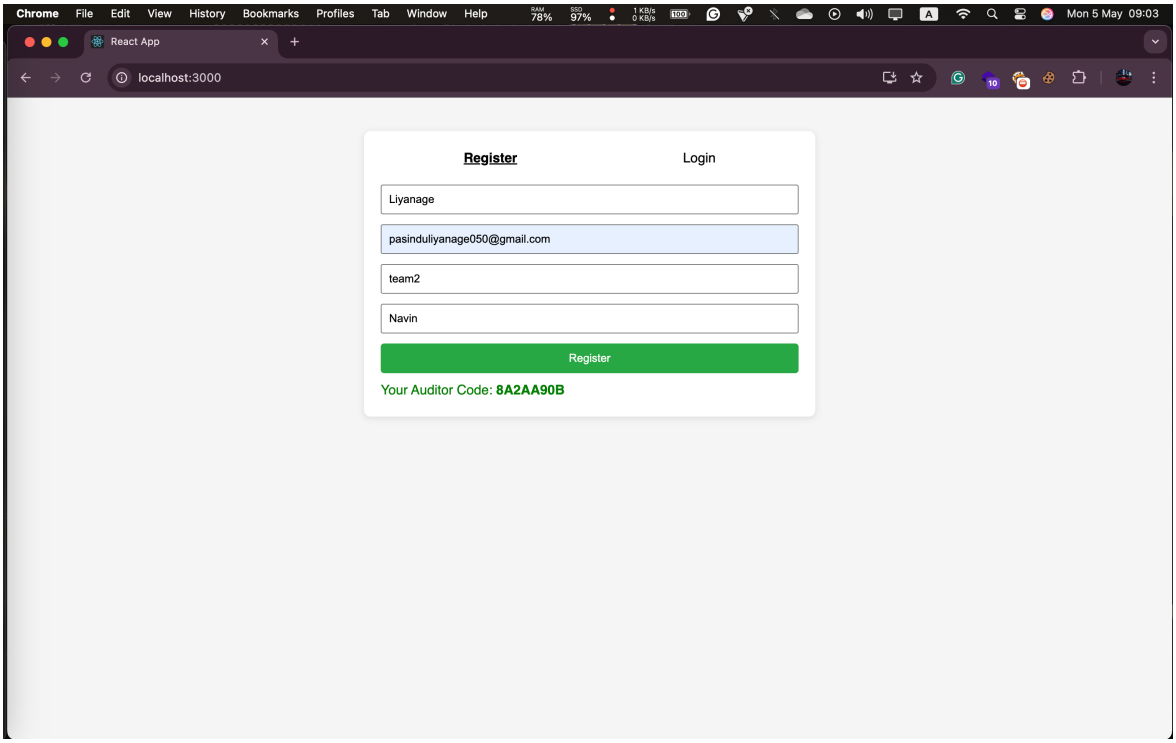


Figure 4 After the Register receive the Auditor Code

2. Login Page
   User interface - This is the login page. Here, users can simply enter their Auditor's code and get the audit report as a PDF.
   Technology - Backend service validating auditor code and fetching reports. Data is stored in a PostgreSQL database.
   View PDF link suggests: Reports are stored as PDFs. Could use server-side PDF generation libraries like pdfkit, jspdf, or Node tools.

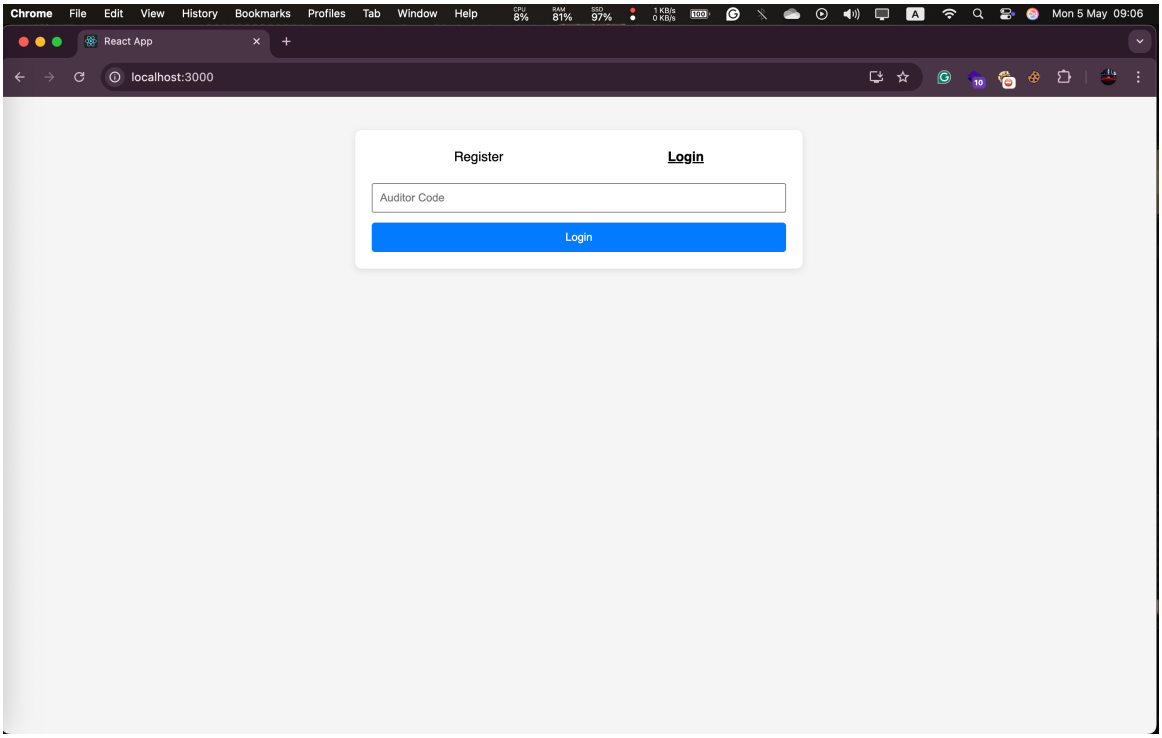


Figure 5 Login Page

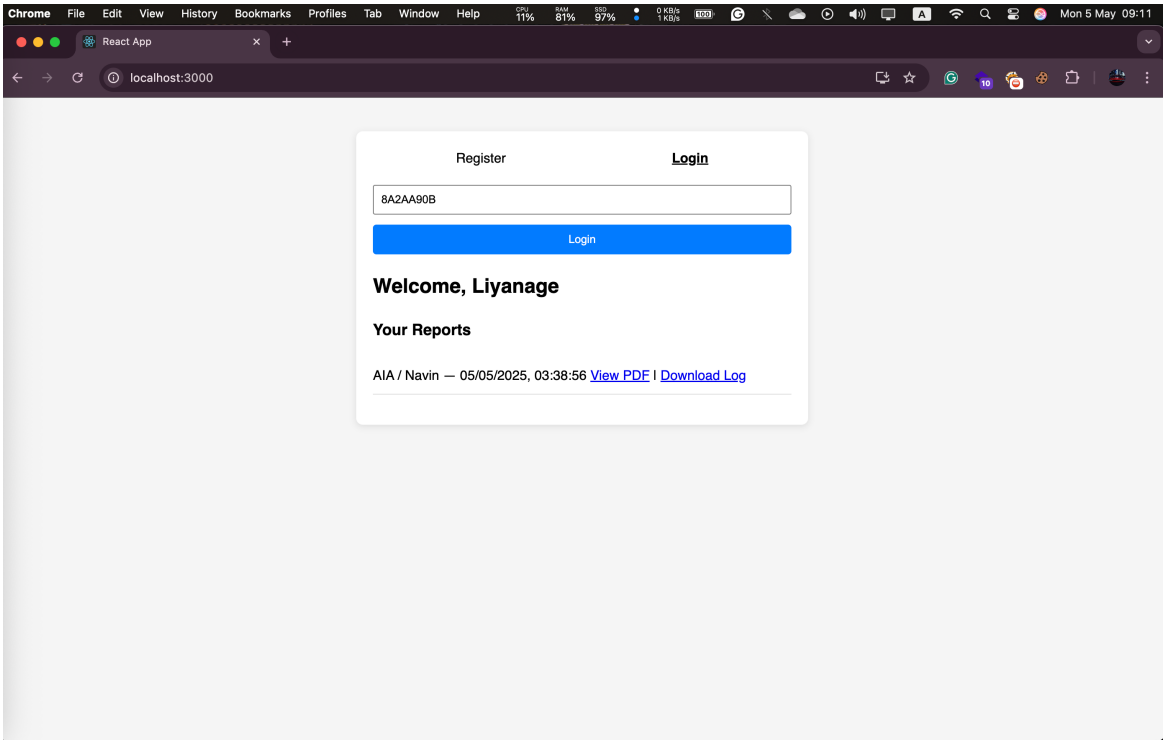


Figure 6 After Enter the Auditor's code

3. PDF after scan:
   User Interface – This is a PDF System and Network Scan report. PDF is automatically generated as the final output of scans.
   Technology - The backend language that runs in FastAPI application. Used for all logic database access, file reading, and PDF generation. A robust library for generating PDFs in Python. It allows for precise control over layout, text, fonts, and page manipulation. Canvas is used to draw text and design the content.

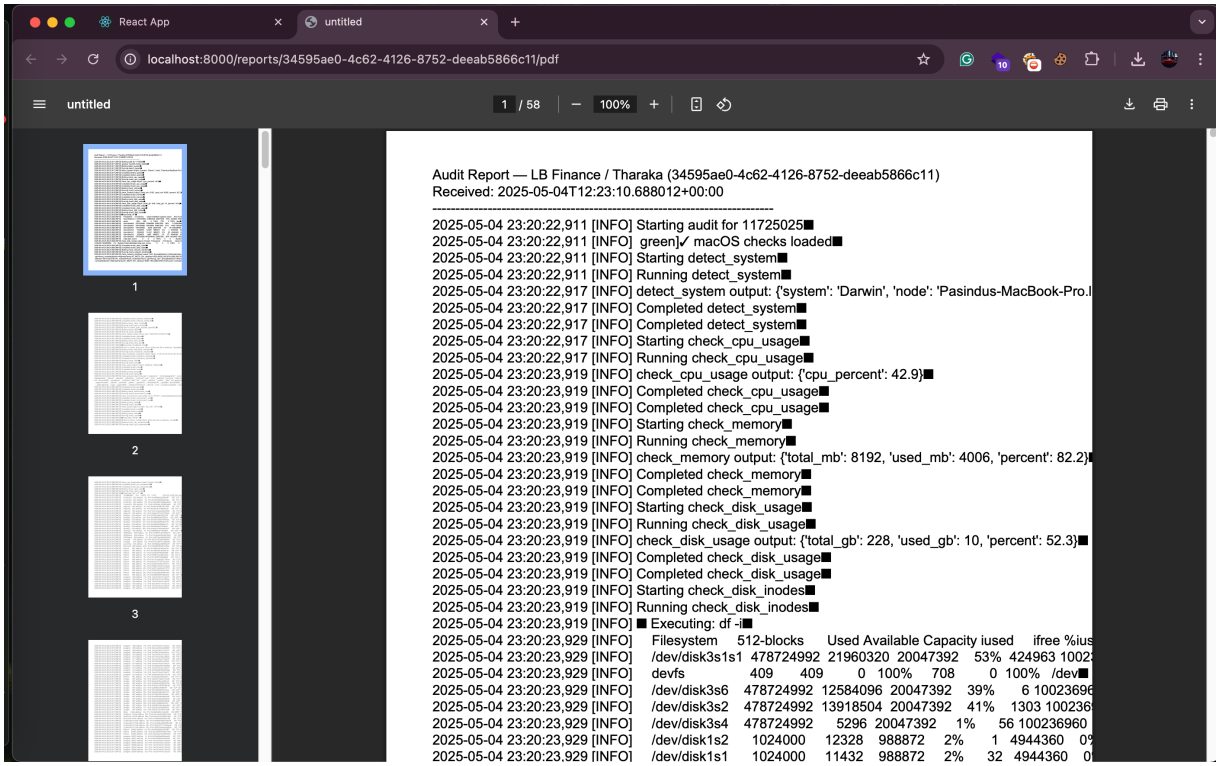


Figure 7 PDF generate after scan

This application is designed to receive and process sustainability data, store it in PostgreSQL, and provide an endpoint to generate sustainability reports in PDF format for users. Additionally, it facilitates user updates and retrieves sustainability rates from the database.

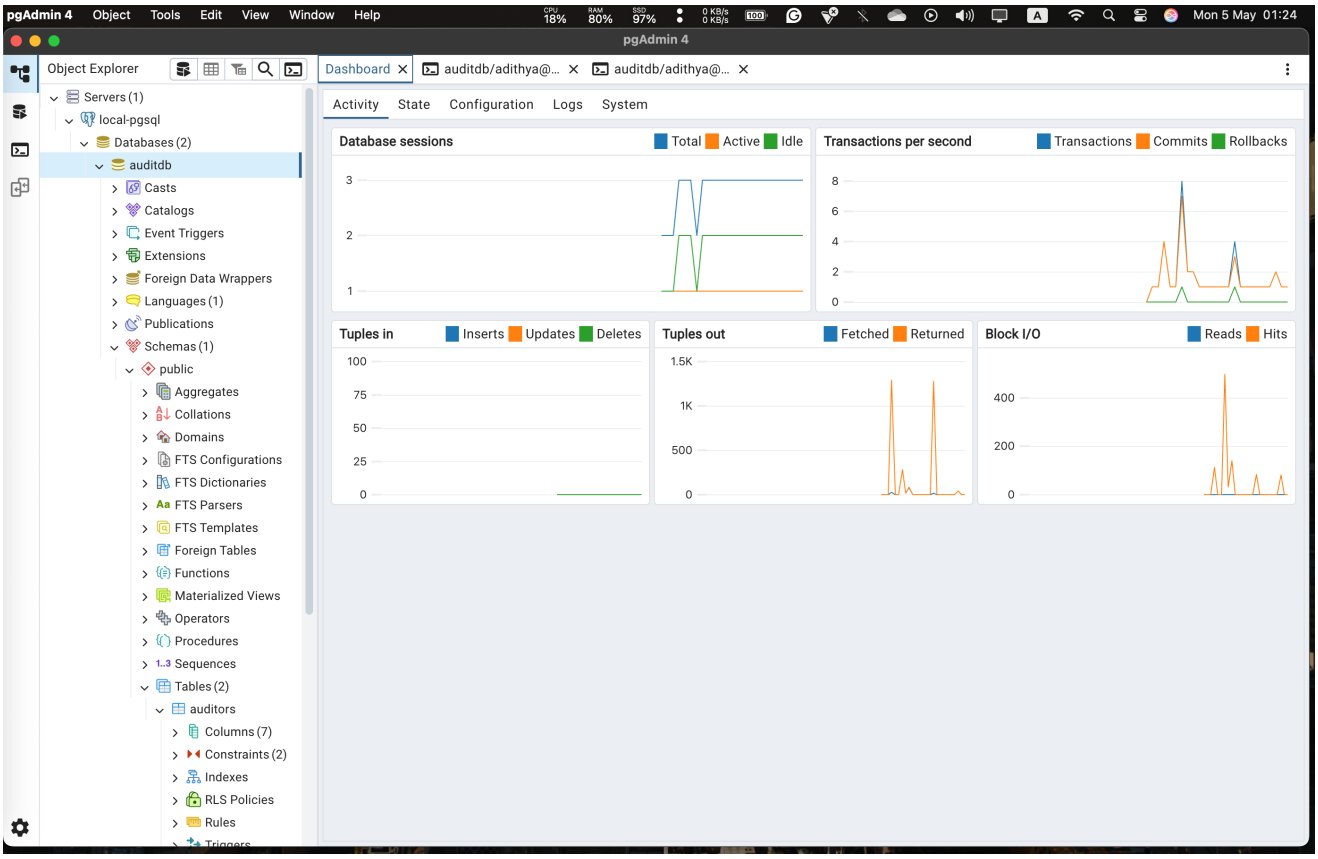


Figure 8 database dashboard

4.2 Programming languages

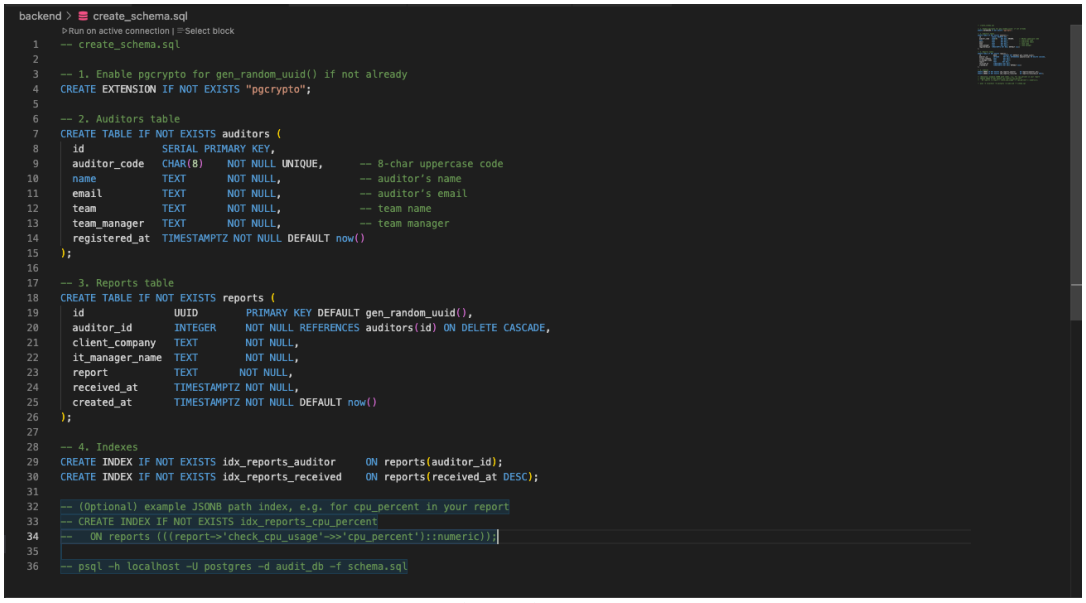
The website has been divided this into 4 parts.

1. Database - PostgreSQL
2. Front end - React JS
3. Back end - Python
4. Integration

1. Database

The Database part of this project involves the creation, management, and utilization of data storage and retrieval systems.

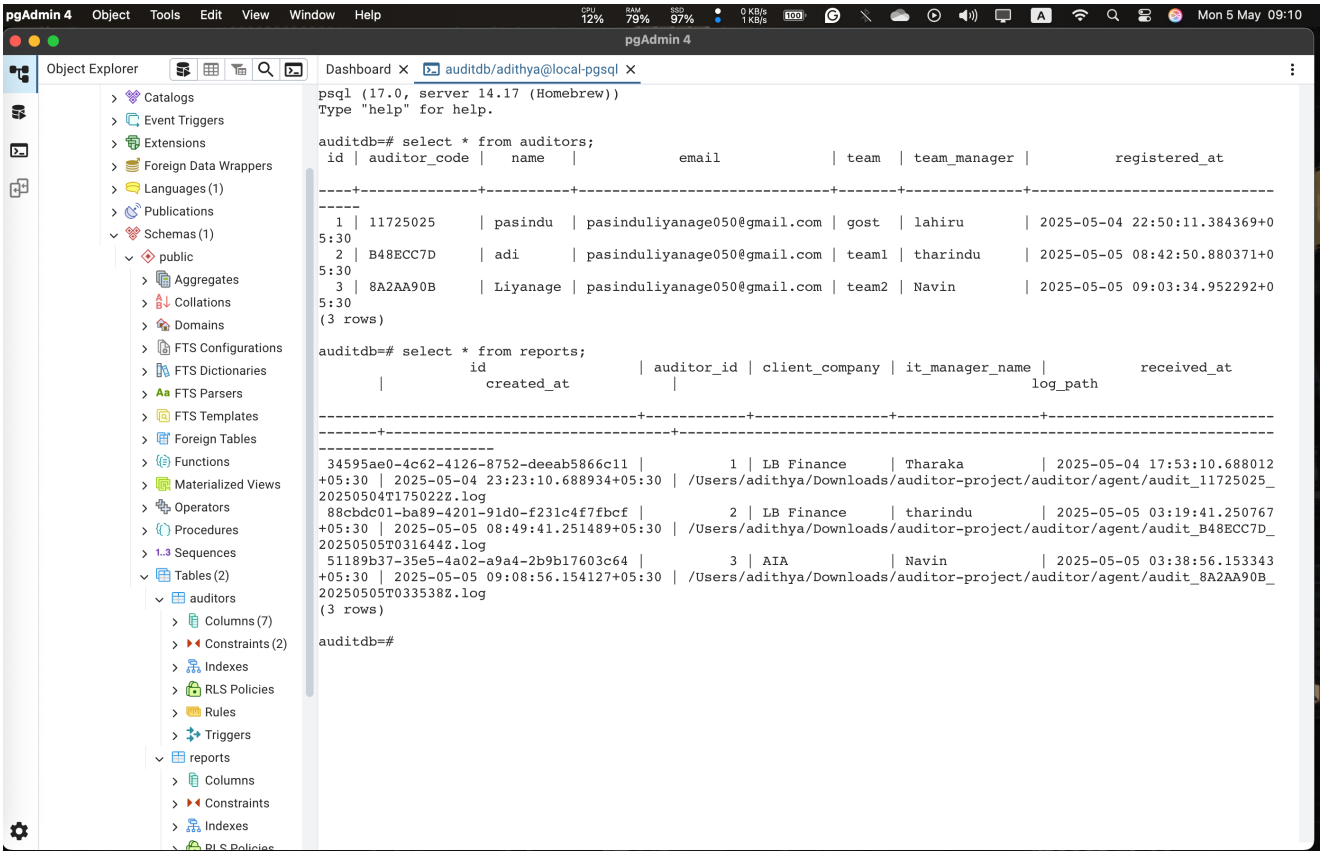
- PostgreSQL is known for its robustness. making it great for applications where data integrity is critical, like auditing systems.
- PostgreSQL is a relational database management system (RDBMS) with object-relational features.
- Setting up the database environment, ensuring proper configuration, security measures, and backup procedures.
- Developing scripts or programs to interact with the database, including data insertion, updates, retrieval, and deletion operations.



```
backend > create_schema.sql
▷ Run on active connection | ☰ Select block
-- create_schema.sql

-- 1. Enable pgcrypto for gen_random_uuid() if not already
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

-- 2. Auditors table
CREATE TABLE IF NOT EXISTS auditors (
  id             SERIAL PRIMARY KEY,
  auditor_code   CHAR(8)    NOT NULL UNIQUE,      -- 8-char uppercase code
  name           TEXT       NOT NULL,             -- auditor's name
  email          TEXT       NOT NULL,             -- auditor's email
  team           TEXT       NOT NULL,             -- team name
  team_manager   TEXT       NOT NULL,             -- team manager
  registered_at  TIMESTAMPTZ NOT NULL DEFAULT now()
);

-- 3. Reports table
CREATE TABLE IF NOT EXISTS reports (
  id              UUID        PRIMARY KEY DEFAULT gen_random_uuid(),
  auditor_id      INTEGER     NOT NULL REFERENCES auditors(id) ON DELETE CASCADE,
  client_company  TEXT        NOT NULL,
  it_manager_name TEXT        NOT NULL,
  report          TEXT       NOT NULL,
  received_at     TIMESTAMPTZ NOT NULL,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);

-- 4. Indexes
CREATE INDEX IF NOT EXISTS idx_reports_auditor     ON reports(auditor_id);
CREATE INDEX IF NOT EXISTS idx_reports_received    ON reports(received_at DESC);

-- (Optional) example JSONB path index, e.g. for cpu_percent in your report
-- CREATE INDEX IF NOT EXISTS idx_reports_cpu_percent
--   ON reports (((report->'check_cpu_usage'->>'cpu_percent')::numeric));

-- psql -h localhost -U postgres -d audit_db -f schema.sql
```

Figure 9 Database schema

This create_schema.sql is script define SQL schema write for the PostgreSQL database.

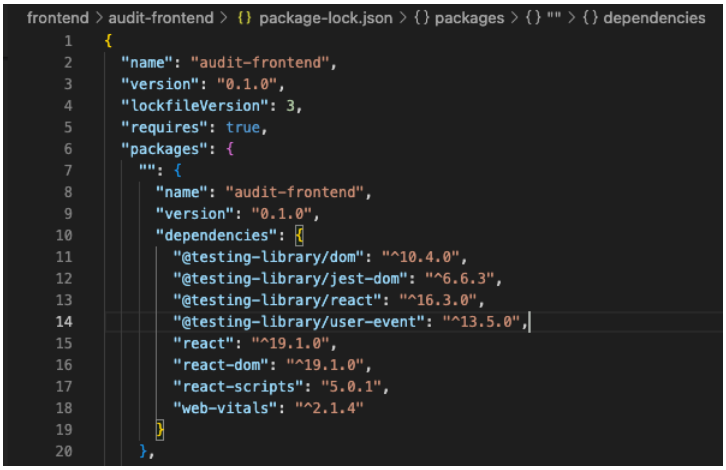


```
pgAdmin 4  Object  Tools  Edit  View  Window  Help        CPU 12%  RAM 79%  SSD 97%   Mon 5 May 09:10
pgAdmin 4

Object Explorer
  Catalogs
  Event Triggers
  Extensions
  Foreign Data Wrappers
  Languages (1)
  Publications
  Schemas (1)
    public
      Aggregates
      Collations
      Domains
      FTS Configurations
      FTS Dictionaries
      FTS Parsers
      FTS Templates
      Foreign Tables
      Functions
      Materialized Views
      Operators
      Procedures
      Sequences
      Tables (2)
        auditors
          Columns (7)
          Constraints (2)
          Indexes
          RLS Policies
          Rules
          Triggers
        reports
          Columns
          Constraints
          Indexes

Dashboard ×  auditdb/adithya@local-pgsql ×

psql (17.0, server 14.17 (Homebrew))
Type "help" for help.

auditdb=# select * from auditors;
 id | auditor_code |   name   |            email             | team  | team_manager |          registered_at
----+--------------+----------+------------------------------+-------+--------------+-------------------------------
  1 | 11725025     | pasindu  | pasinduliyanage050@gmail.com | gost  | lahiru       | 2025-05-04 22:50:11.384369+0
5:30
  2 | B48ECC7D     | adi      | pasinduliyanage050@gmail.com | team1 | tharindu     | 2025-05-05 08:42:50.880371+0
5:30
  3 | 8A2AA90B     | Liyanage | pasinduliyanage050@gmail.com | team2 | Navin        | 2025-05-05 09:03:34.952292+0
5:30
(3 rows)

auditdb=# select * from reports;
                  id                  | auditor_id | client_company | it_manager_name |          received_at
       |           created_at           |                                        log_path
--------------------------------------+------------+----------------+-----------------+-------------------------------
-------+--------------------------------+-----------------------------------------------------------------------------------------------------------
---------------------
 34595ae0-4c62-4126-8752-deeab5866c11 |          1 | LB Finance     | Tharaka         | 2025-05-04 17:53:10.688012
+05:30 | 2025-05-04 23:23:10.688934+05:30 | /Users/adithya/Downloads/auditor-project/auditor/agent/audit_11725025_
20250504T175022Z.log
 88cbdc01-ba89-4201-91d0-f231c4f7fbcf |          2 | LB Finance     | tharindu        | 2025-05-05 03:19:41.250767
+05:30 | 2025-05-05 08:49:41.251489+05:30 | /Users/adithya/Downloads/auditor-project/auditor/agent/audit_B48ECC7D_
20250505T031644Z.log
 51189b37-35e5-4a02-a9a4-2b9b17603c64 |          3 | AIA            | Navin           | 2025-05-05 03:38:56.153343
+05:30 | 2025-05-05 09:08:56.154127+05:30 | /Users/adithya/Downloads/auditor-project/auditor/agent/audit_8A2AA90B_
20250505T033538Z.log
(3 rows)

auditdb=#
```

Figure 10 Queries run against the audit.db

2. Front End

- The Front-End section focuses on creating the user interface and user experience of this application using React.js:
- Design and development of the user interface components using React.js, ensuring they are responsive, intuitive, and accessible.
- Interaction with the back-end server through API calls to fetch, display, and send data to the server.



This lists all the npm packages in the Front-end that are on to run, build, and test.

Figure 11 npm packages

3. Back End

The Back End part involves setting up the server, APIs, and logic to handle requests from the front end, process them, and return the appropriate responses:

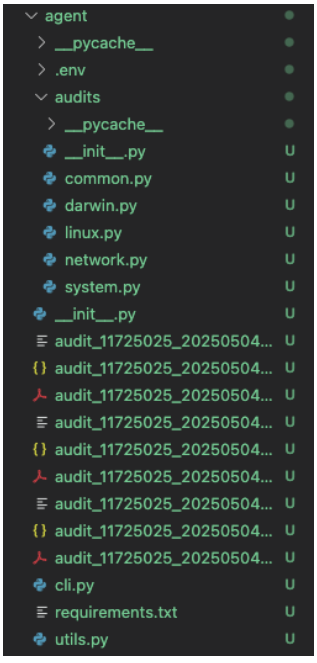
- The back-end is built with FastAPI to define routes and handle requests.
  The backend provides a secure and organized way to register auditors, collect and store report, access reports that logs, generate a PDF after Serve as the logic and data layer behind a front-end.

4. Integration between Front End and Back End

This final part covers the methods and practices to connect the front-end and back-end parts.

- Back-end main.py to identify exposed APIs and the front-end code to see how those APIs are being consumed.

4.2.1 Client Side

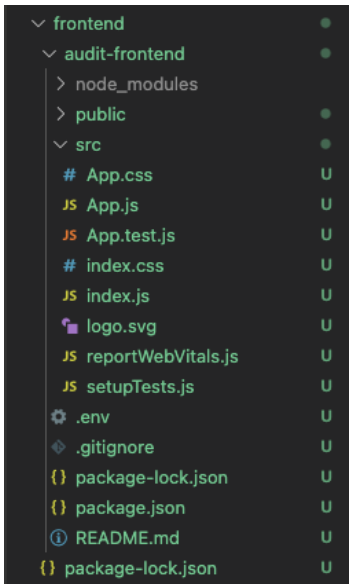


The agent in this audit tool performs that runs on the client's System to collect audit data. It performs local system checks and submits the results to the remote audit server.

- Detects OS and platform info and performs relevant checks.
- Checks security and compliance
- Collect all check results into a single report. Enter all logs.
- Saved report as .json and .PDF locally. That is a benefit for the client

Figure 12 Client-side code structure

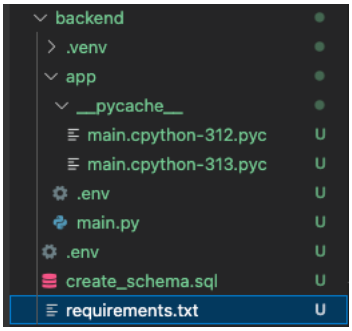
4.2.2 Front-end



This is the front end of the project. This has structured the React code with a focus on making to easier and extensible. All packages are listed on package.json.
App.js is the main component of the Register/login page. Also CSS styles for styling purposes.

Figure 13 Front-end code structure

4.2.3 Back-end



When building this backend for the project, use Python. .env stores environmental variables. The main application folder is main.py. It contains,

- Endpoint definitions
- Database connection setup
- Pydantic models
- Middleware (like CORS)

Figure 14 Back end code structure

5. Literature Review

5.1 Literature Review

To better understand the current state of IT security testing tools and auditing processes, this literature review analyses available technologies, their capabilities, and the challenges that IT auditors. The environment of information systems security auditing is continually changing as technology progresses and cyberattacks become more complex. The following overview takes on research papers to highlight the importance, techniques, and problems of conducting effective information system security audits.

Security audits are crucial tools for firms to analyse their information security postures. emphasizes the need of regular audits in establishing a security baseline, allowing vulnerabilities to be detected and addressed before they are exploited [2].
Similarly, highlights that cyber security audits conduct a thorough evaluation of an organization's IT infrastructure, finding dangers and weak links in the system [3].

Focus on vulnerability audits:
Describe the importance of using various tools and strategies to comprehensively cover all components inside an information system [4], [5]. The program makes it simple for auditors to understand how the tool increases productivity, accuracy, and reliability in finding vulnerabilities and emphasizing crucial areas.
IT audit processes include solutions that automate basic audit operations, such as scanning for open ports and firewalls and logging to detect unauthorized connections [6], [7].

Address an application module for wireless network security testing, which simulates potential attack pathways and assesses vulnerability [6].

The importance of port scan detection:
This reconnaissance phase includes methods like port scanning, in which attackers analyse TCP and UDP ports to find open communication channels and system weaknesses. Open ports, such as entry points in a building, can expose systems to unauthorized access, and systems with numerous open ports are generally more vulnerable. However, fewer open ports may represent a larger security risk in some configurations. [8]. This study underlines the significance of extensive testing for open ports on computer networks for assessing security [9].

Challenges and Technical Depth:
The specific nature of the skills required for these audits involves auditors having extensive technical knowledge.IS auditors require specialized abilities to effectively analyze technical observations. In order to accomplish this, the application should have a training module. Provides information and courses to help auditors improve their technical abilities in IS auditing. Includes case studies and examples that demonstrate frequent challenges in IS auditing and how to handle them.
Creates thorough audit reports that summarize results, make actionable recommendations, and identify areas for improvement. Management uses images (charts and charts) to convey data in a clear and effective manner [10], [11].

Modular framework for security scanning:
NSS's modular configuration offers focused scanning and auditing capabilities, making it both efficient and user-friendly. It attempts to identify system vulnerabilities in the same way as NSS can find and resolve exploits and information leaks [4].
Host Scan: Used to detect live hosts.
Port Scan: Detecting open and closed ports.
Pinging and NSLookup are used to test connectivity and resolve DNS information.

User Awareness and Training:
Several articles, notably, emphasize the crucial significance of user awareness and training in improving company safety [6], [9]. Regular discussions and presentations regarding security policies may significantly improve staff knowledge and compliance, lowering the likelihood of human errors that result in security breaches.

Integration with Governance:
This means,
that this project involves making important decisions to steer it in the right direction. When it comes to integrating information systems (IS) audits with control, the results of these audits should be directly linked to those higher-level decision-making processes.
Why an IS audit is important:

- An IS audit reviews a company’s IT infrastructure, systems, and practices. The goal is to:
- Data is accurate, reliable, and secure.
- IT systems are efficient and support business goals.
- Risks are managed appropriately.
- A well-conducted IS audit provides management with an understanding of what risks are operating in the IT environment and what needs to be changed.

This improves corporate governance,

- By providing this information, IS audits give company leaders:
- Understanding of cyber risks and how well they are being managed.
- Information to help make informed decisions about policies, investments, and priorities.

- Confidence that the company's digital assets are secure and compliant with laws/regulations.

Effective IS audits help to improve corporate governance by giving management with the insights it needs to make sound decisions. [3] explores how cyber risk management frameworks can bridge the gap between technical cybersecurity practices and executive governance, establishing a security culture throughout the firm.

Future Research Directions:
This means exploring or researching how emerging or developing technologies can be utilized. In this case, we are not talking about technology that is widely used today, but rather about what is emerging or becoming more advanced - especially in terms of how it might be applied in the future.
such as artificial intelligence (AI) and machine learning (ML)
discusses these future technologies:

Current computer systems fall under the category of artificial intelligence (AI) because they execute tasks that normally only humans could accomplish including language interpretation pattern recognition and decision-making.

The technology of machine learning exists as a section of AI that enables systems to acquire knowledge from provided data. ML models lack predefined instructions because they detect data patterns to create predictions or decisions without human guidance about their processing methods.

In improving audit processes,
Reviewing digital records of financial operations across audit tasks, AI and ML work to improve the audit process through these tasks:

- Using automation for routine checks saves time and reduces the possibility of human error.
- Identifying unusual financial data and suspicious behaviour patterns suggests fraud opportunities.
- The application provides proactive capabilities to identify possible cyber threats.
- Technology implementation should enable predictive capabilities that detect cyberattacks before they occur.

With ML and AI, systems can:

- A system should monitor for unknown network traffic patterns to identify potential threats.
- The system uses historical attack information to predict future system vulnerabilities.
- Managers should identify any unusual activity that points toward creating threats.
- The system must learn from previous security breaches to make automatic security improvements.

Investigating the usefulness of future technologies, such as artificial intelligence and machine learning, in improving auditing processes and giving predictive capabilities against potential cyber threats [5], [11].

5.2 Drawbacks of the existing system

Within the project proposal, readers will find an extensive breakdown regarding the boundaries of current vulnerability-scanning alongside auditing systems.

Time-consuming manual processes:
Auditors currently need to execute different tools consecutively through their existing systems since port scanning tools require manual succession with other vulnerability scanners. Manually integrated results from security tests need to be combined into one product. The manual process of data consolidation during audits brings an increased chance of human mistakes, while making the audit process exceedingly slower.

Reporting solutions that are not integrated and not automated:

Professional vulnerability scanners deliver detailed information, but their exports generate unprocessed or partially processed data. Auditors dedicate substantial time to converting raw outputs from vulnerability scanners into reports that satisfy clients' needs. The ability to use scripts for automation exists, yet many organizations lack the capabilities and technical know-how to develop their own programmed connections. Auditors waste numerous hours on scripting to process and normalize scan data because basic automated report generation capabilities are absent from their scanner tools.

Potential Gaps in Coverage
Multiple distinct specialized tools selected by auditors for network scans and system configurations and web application testing can result in protection gaps that individual scanners may fail to detect. A single scanning tool may perform well within open-port detection while lacking capability in misconfiguration assessment, which creates incomplete system security visualization results.

Human Error and Inconsistency
The need for manual intervention throughout multiple steps in existing scanning/reporting methods creates opportunities for human mistakes which surface most prominently during raw result consolidation and vulnerability priority determination. The use of inconsistent report formatting between different clients or projects hinders overall tracking of progress as well as hinders result comparison across projects.

6. Method of approach

6.1 Feasibility Study

IT auditors can easily implement the basic security testing application. The application integrates with standard workflows through audit procedure automation and provides an interface that requires minimal training for users. The program can be used by various auditor levels, and audit standards compliance enables its practical use across multiple system environments.

6.1.1 Operational Feasibility

Operational feasibility represents the suitability of the basic security testing application workflow to facilitate the current audit process and the suitability for the needs of IT auditors and security professionals. Successful implementation of this solution requires a risk analysis assessment, along with a deployment scenario analysis and evaluation of user adaptation needs along with training requirements. An operational feasibility assessment has been performed as this terminal-based application.

Suitability for IT Auditors and Security Professionals
The Security Testing application performs seamless scans through its core design, which makes it highly suitable for IT auditors. It is suitable for:

IT Auditors and Cybersecurity Consultants:
Security audit automation remains the main objective of this solution. Consultants obtain complete vulnerability reports from this application to effectively analyse security measures.

How Factors Affect Operational Capability in Detail
1. Administrator Access Requirements
IT auditors typically require administrator-level privileges/permissions to run comprehensive vulnerability scans and access critical system configurations. System-level security scans can be accessed using sudo/root.

For example, certain scans, such as port scans, vulnerability scans, or obtaining system logs, may require administrative privileges.

Auditors should have defined permissions that balance operational needs and security constraints.

2. Ease of use for IT auditors
Since the project is based on a command-line interface, simplicity in using terminal commands is essential.
Commands should be easy to remember, and parameters should be clear and concise.
Providing a complete help page or manual documentation is important for ease of adoption. This is what this command (--help) is used for.
The application should be easy for auditors to quickly understand the command syntax, parameters, and expected outputs.

3. Level of automation
Your tool should automate important factors such as vulnerability scanning, port analysis, and system configuration checks.
Scanning reduces manual intervention by auditors, increases accuracy, and reduces audit time.
After each scan, reports should be automatically generated within this project, clearly describing the risks, risk levels, and remediation proposals.

4. Data Security and Confidentiality
As automated scans produce reports containing sensitive risk information, data security is therefore important.
These outputs should be managed securely, and procedures should be clearly outlined to prevent unauthorized disclosure.

6. System Scalability
The system needs to smoothly handle security checks on both small host systems and extensive corporate network scans.
The software needs to maintain its operational capacity while performing many security audits simultaneously.
Scalability makes the application flexible by handling different audit assignments that arise across different environments.

7. How WPA/WPA2 Testing Fits the Project:
IT auditors need efficient tools to evaluate wireless network security, so WPA/WPA2 authentication tests are implemented exactly within their job responsibilities.
Saves time and effort:

- Security assessments of WPA/WPA2 through automation reduce human labor requirements, thereby bringing faster and more reliable audits.
- The application design should maintain user-friendliness so that non-expert IT auditors can execute detailed security testing procedures. This system passes operational feasibility as IT auditors can perform WPA/WPA2 network assessments through the application without facing any difficulty.

6.1.2 Technical Feasibility

This feasibility report discusses the technical feasibility of the basic security testing application, the auditor's technical competence, system compatibility, compliance with security standards, system resource utilization, and how to technically integrate existing security tools.
It further explores whether the project can be developed using existing technologies, tools, and expertise.

1. Auditor's Technical Competency
IT auditors should have basic proficiency in the Linux operating system.
For example, they should have knowledge of basic Linux commands such as sudo, grep, cat, nano, chmod, etc.
Basic understanding of networking principles such as TCP/IP protocols, open/close ports, firewall rules, etc.

The application should have extensive reading, including clear command examples, manuals, and sample scripts, to bridge the proficiency gaps in cybersecurity tools such as Nmap, OpenVAS, and basic penetration testing techniques.

2. System Performance and Scalability
The application needs to execute many parallel scans without overburdening system resources including CPU and memory. System reliability under various load conditions becomes more certain through regular testing of large network systems.

The application needs optimized Python scripts to maintain system resource efficiency for big-scale scanning operations.

3. System resource utilization
The terminal-based application should optimize its resource utilization to prevent straining system resources. Check if the application maintains an efficient operational state regardless of system hardware capacities.

4. Impact of WPA/WPA2 testing.
Your application can integrate both Scapy and Wireshark tools as Wireless Packet Analysis Tools to perform wireless network authentication analysis. The implementation of libraries that establish network interface connection for performing WPA/WPA2 authentication procedures.

Since the project depends on peripherals the system needs to process WPA/WPA2 scans using available hardware without additional specifications. Through available resources of programming expertise and necessary systems and appropriate technologies, the implementation of WPA/WPA2 security testing proves possible for the project.

6.2 Testing

The testing phase of the basic security testing application focused on the functionality of both system and network security modules in a live Linux environment. The tests were performed via SSH on a remote audit server using Python scripts such as systemSecurity.py and networkSecurity.py.

Key components tested included:

- Database checks validated the identification of running, installed, and binary-accessible databases.
- Firewall status: identified whether the firewall was enabled or disabled.
- User and process auditing: captured all system users, logged in sessions, and flagged suspicious processes or failed login attempts.
- SSH configuration: checked whether password-based SSH authentication was enabled.
- Ping and network tests: performed connectivity checks to internal and external IP addresses.

Backend testing

The backend of the IT Audit System was tested to ensure that all core functionalities are running correctly, securely. The tests were performed using tools such as Uvicorn, cURL, HTTP clients, and built-in FastAPI Swagger documents. The objectives focus on the backend test.

- Verify that API endpoints are working as expected.
- Validate data integrity in the database.
- Ensure proper error handling and security.
- Test file serving and PDF generation functionalities.
  Security Test focuses on – Ensure the correct access by given auditor_code. Verified log path and confirmed .env and files not exposed.

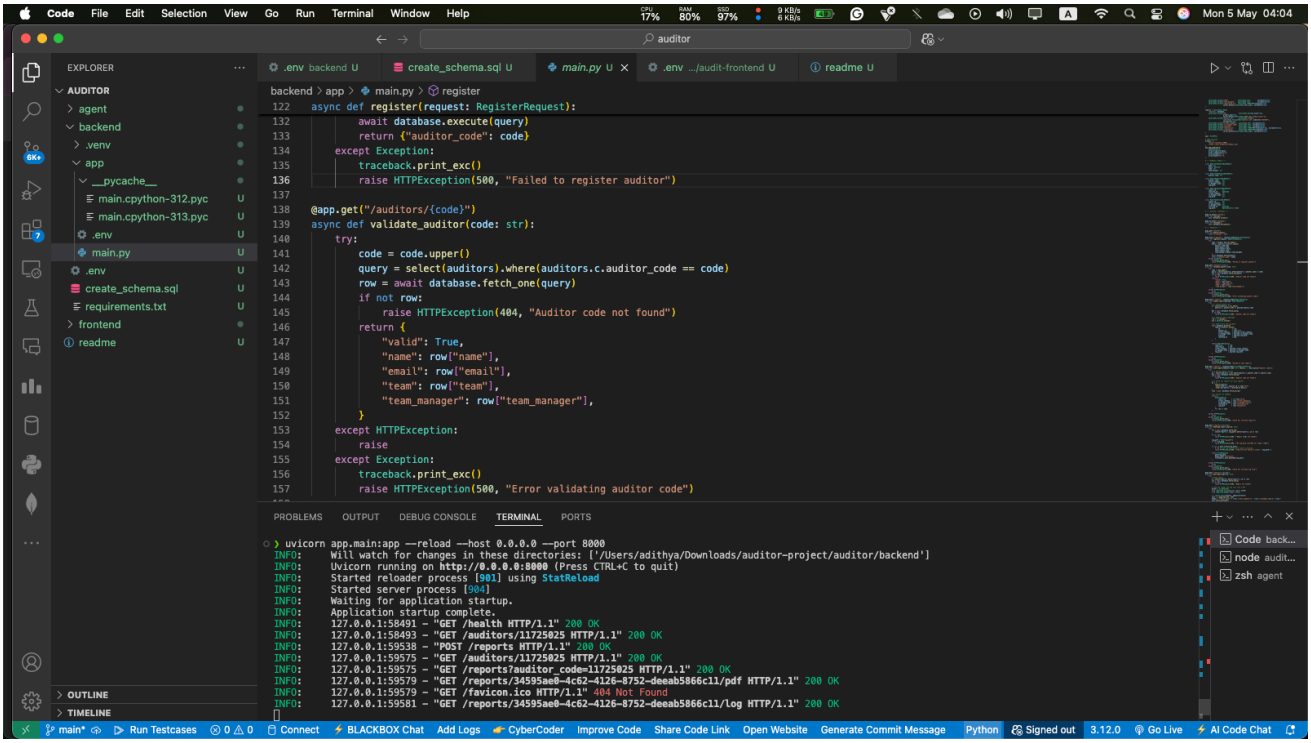


Figure 15 Backend testing

Front-end testing

The front-end application was tested using the React Testing Library with Jest as the testing framework. The testing environment verifies essential user actions and component appearance functionality across both registration and login procedures.

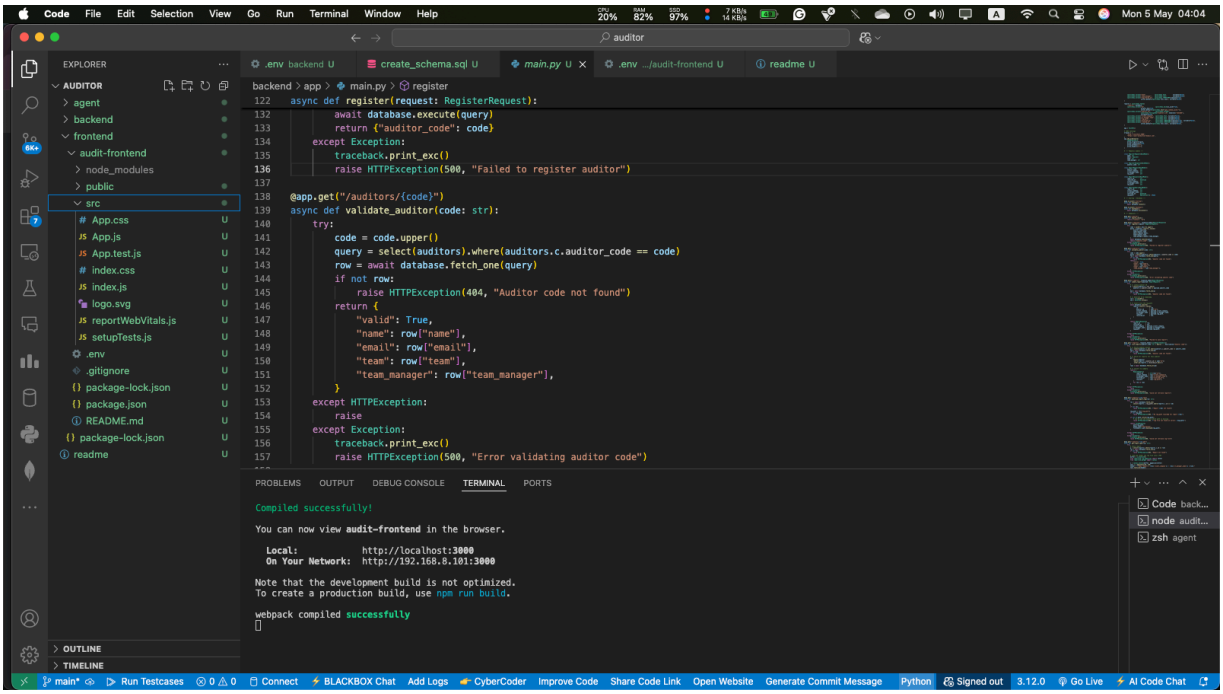


Figure 16 Front-end testing

Client-side testing

Client Side using local terminal environment. A client verification is performed here. Both of .json and .pdf report are saved locally with timestamps if their offline records keeps even submission failds.

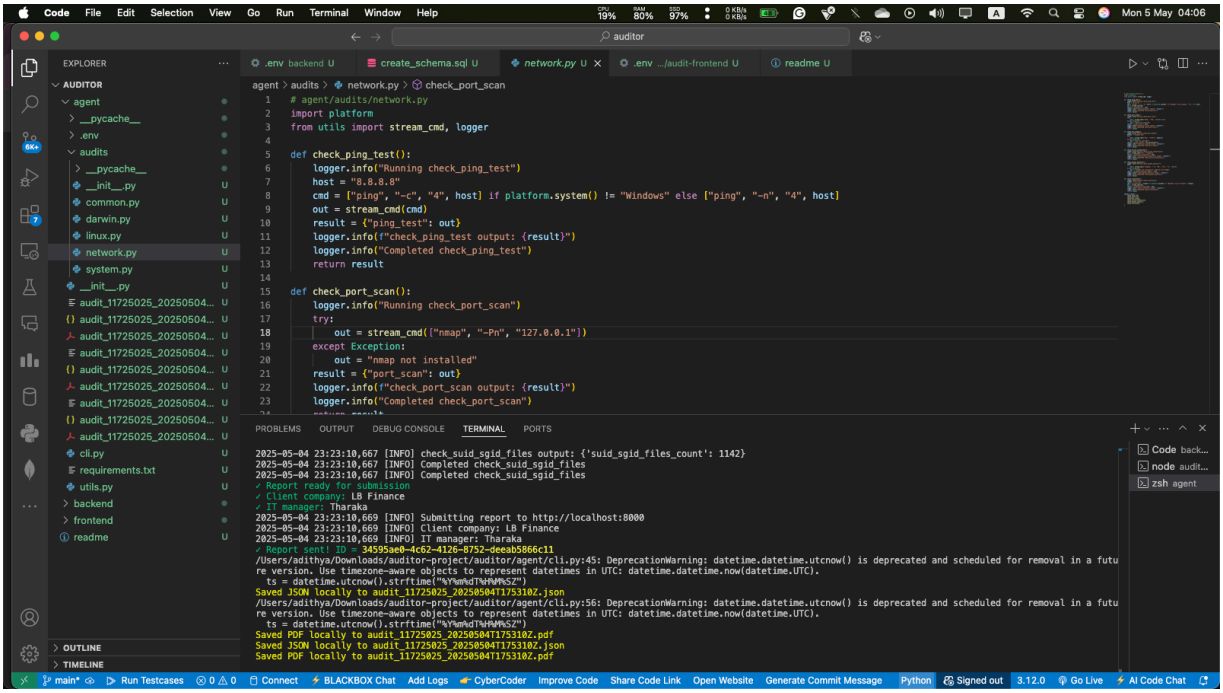


Figure 17 Client-side testing

Tests performed when implementing via AWS under future implementation.

These include essential tests that ensure the security of a system.

Backend testing

1. systemSecurity.py
   This shows the results from running your security audit script, systemSecurity.py.
   Incudes:
   - System Info: OS version, CPU architecture, RAM, and disk usage.
   - Network Info: Local/Public IP, DNS, Gateway.
   - Firewall Status: Marked as inactive — a security risk.
   - Listening Ports: Shows DNS and SSH ports active (ports 22 and 53).
   - Login History: Details on past logins and currently logged-in users.
   - Created User Accounts: Shows default and created users like nobody, ubuntu.
   - Process Listing: Running processes, including suspicious ones.
   - SSH Config: Flags on Password Authentication settings.
   - Database Checks: Found installed DBs (SQLite) and some running DBs, although binaries were missing.
   - Suspicious Activity: Detected suspicious processes and failed login attempts — a great addition for audit flagging.
   - Unauthorized Users: Lists system users that might be unnecessary or risky.
   - Cron Jobs: Nothing found, which might be okay depending on the system.

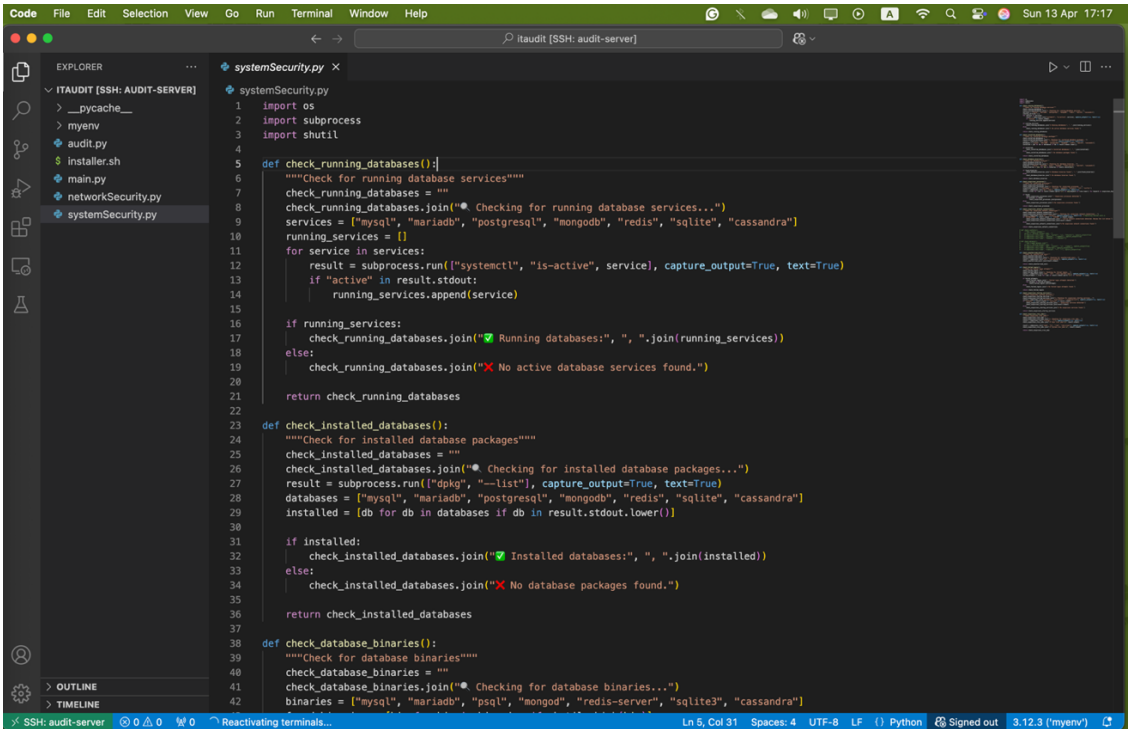


Figure 18 Backend Testing in System Security

2. networkSecurity.py

The script defines several core functions:

run_command:

Runs terminal commands from Python and returns the output, used throughout the script to collect data.

SystemInfo:

Gathers and formats details like:

- OS version
- CPU architecture
- RAM and disk usage

NetworkInfo:

Retrieves network-related data, including:

- Hostname
- Public & local IP addresses
- Default gateway
- DNS servers

Firewall_Status:

- Will check the firewall status and any active rules And many etc.

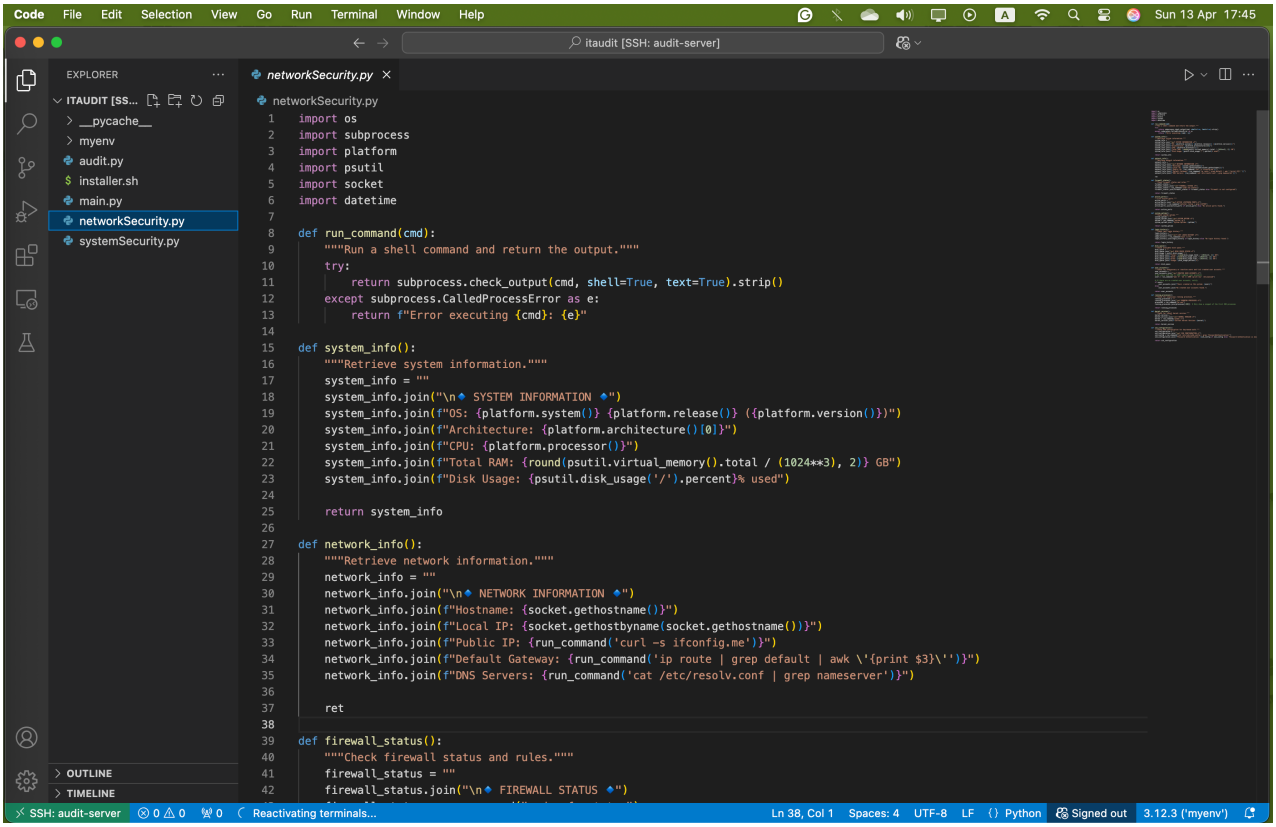


Figure 19 Backend Testing in Network Security

3. Final output – in the testing part

```
Last login: Sun Mar 23 12:33:54 on ttys000
❯ mv ~/Downloads/audit.pem ~/.ssh/
mv: /Users/adithya/Downloads/audit.pem: No such file or directory
❯ mv ~/Downloads/audit.pem ~/.ssh/
mv: /Users/adithya/Downloads/audit.pem: No such file or directory
❯ ssh audit-server
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Mar 23 07:06:31 UTC 2025

  System load:  0.0               Processes:             113
  Usage of /:   42.5% of 6.71GB   Users logged in:       1
  Memory usage: 24%               IPv4 address for enX0: 172.31.92.88
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sun Mar 23 06:12:57 2025 from 175.157.41.75
ubuntu@ip-172-31-92-88:~$ cd
ubuntu@ip-172-31-92-88:~$ ls
itaudit
ubuntu@ip-172-31-92-88:~$ cd itaudit
ubuntu@ip-172-31-92-88:~/itaudit$ ls
__pycache__  installer.sh  main.py  myenv  networkSecurity.py  systemSecurity.py
ubuntu@ip-172-31-92-88:~/itaudit$ python3
Python 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>>
KeyboardInterrupt
>>>
[1]+  Stopped                 python3
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$ cd
ubuntu@ip-172-31-92-88:~$
ubuntu@ip-172-31-92-88:~$
ubuntu@ip-172-31-92-88:~$
ubuntu@ip-172-31-92-88:~$ cd itaudit
ubuntu@ip-172-31-92-88:~/itaudit$ ls
__pycache__  installer.sh  main.py  myenv  networkSecurity.py  systemSecurity.py
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$
ubuntu@ip-172-31-92-88:~/itaudit$ python3
python3           python3-config     python3.12         python3.12-config
ubuntu@ip-172-31-92-88:~/itaudit$ python3
python3           python3-config     python3.12         python3.12-config
ubuntu@ip-172-31-92-88:~/itaudit$ python3
python3           python3-config     python3.12         python3.12-config
ubuntu@ip-172-31-92-88:~/itaudit$ python3
```

Figure 20 Linux terminal session connected to a remote server via SSH.

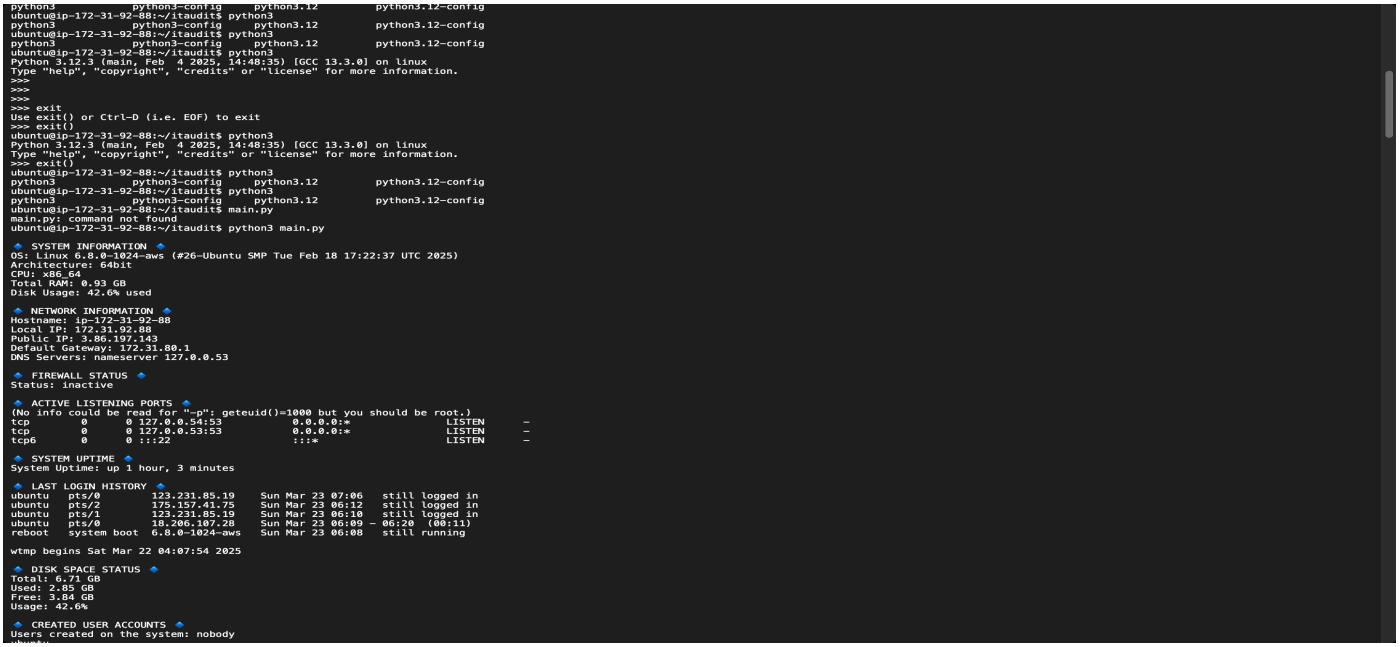
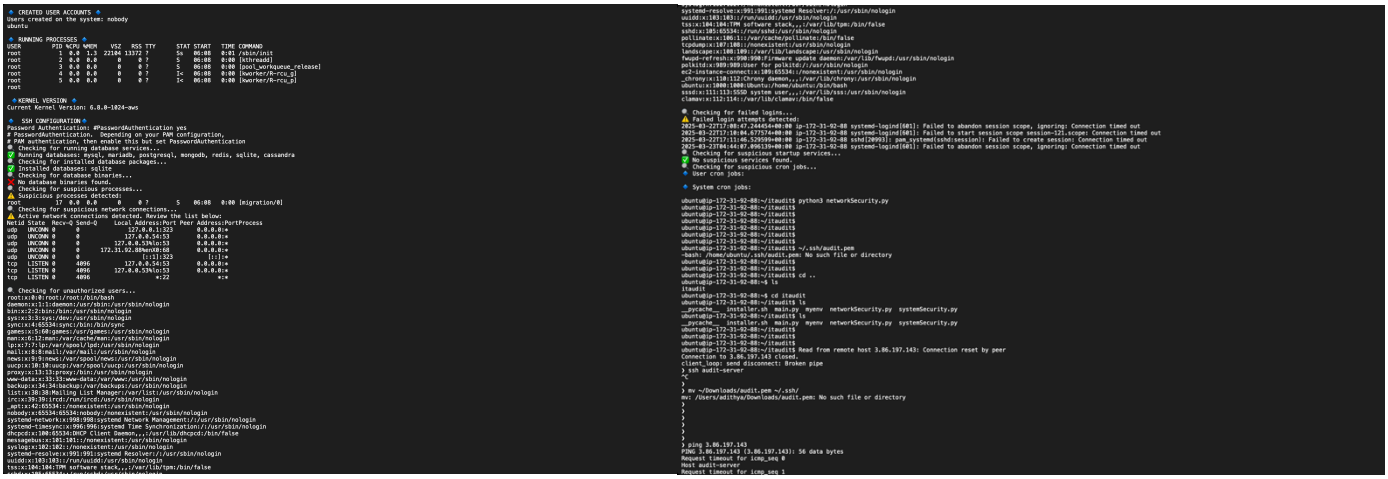


Figure 21 terminal output after running main.py script



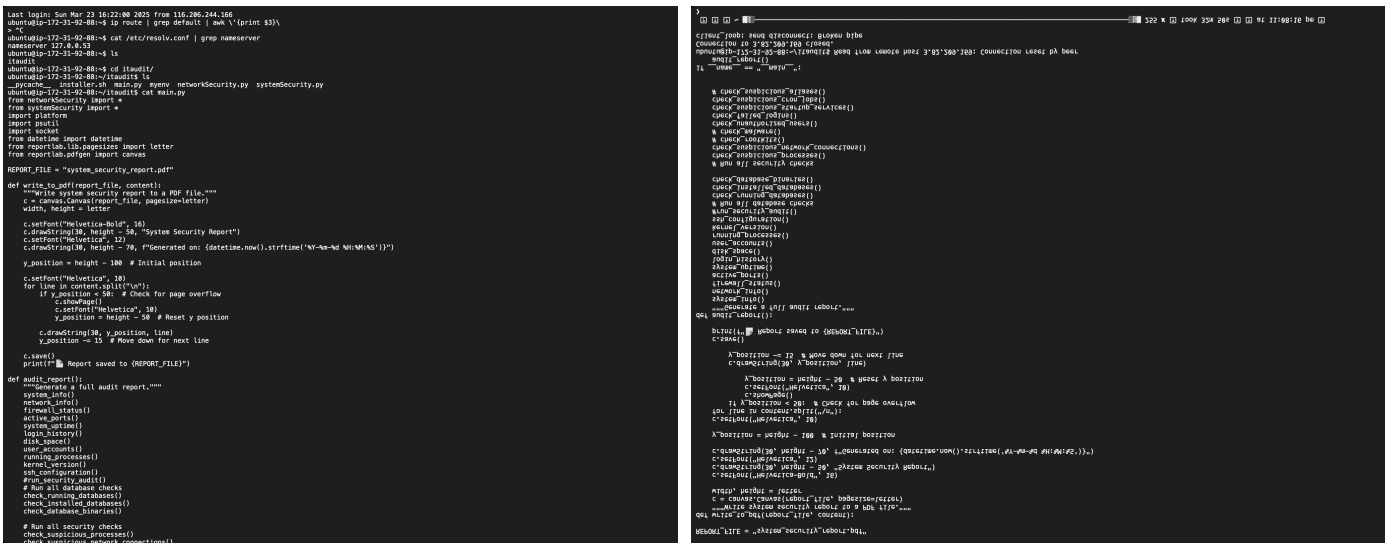


Figure 22 testing part

6.3 Development Methodology

The basic security testing application is developed following the Agile software development methodology. Agile software is chosen because of its iterative, flexible, and feedback-driven nature, which means continuous testing and improvement throughout the entire process. If new cybersecurity threats emerge, the application can quickly adapt and incorporate the necessary updates.

6.3.1 why Agile software is chosen

because,

- Continuous testing can identify bugs and improve security.
- Continuous iterations reduce development time due to rapid development cycles
- Improved security allows for frequent updates to allow for continuous security improvements.

6.3.2 How Agile is involved in this project

1. Development process and testing

The project is divided into manageable modules:

- Vulnerability scanning
- WPA/WPA2 security testing
- Automatic report generation etc.

Each module is developed and tested here. This minimizes the likelihood of major system failures.
Here, the system is verified to be functional and useful at every stage of development.

2. Regular testing and evaluation

Cybersecurity tools should be tested to ensure accuracy and reliability.

3. Alignment with IT audit processes

IT audits require flexibility in security over the network, infrastructure.
Agile allows the tool to be customized and refined based on feedback from potential users.
If auditors need additional scanning features, they can change the system without planning.

6.3.3 How Agile connects to other aspects of the project

Project aspect Agile development impact
Features using WPA3 in future updates allow for new security testing to be added gradually.
Daily testing helps to quickly identify bugs, improve system stability. Agile allows for easy changes.

The development of the basic security testing application uses the agile development methodology. Since security testing requires flexibility, frequent testing, and incremental improvements, Agile is the best fit for the basic security testing application.

6.4 Programming Languages and Tools

The basic security testing application is developed using a combination of secure programming languages and tools. These tools have been selected to ensure that the application is efficient, scalable, and secure while supporting cross-platform deployment.

6.4.1 Programming languages used

1. React JS

The Front-End section focuses on creating the user interface and user experience of this application using React.js:

2. Python:

Python is widely used in cybersecurity due to its ease of use, extensive library support, and readability. This makes it ideal for complex, automated security applications and vulnerability scanning.

System Interaction:
Executes shell commands within Python scripts for checks such as SSH configuration, firewall status, SUID/SGID file checks, and system account validations.

6.4.2 How programming languages and tools are involved in the project:

Python:
Quickly and accurately automate complex vulnerability scanning tasks through tools like Nmap and Python scripts.

Easy-to-use interface:
Python scripts handle the backend complexity, presenting simple output in an easy-to-read format directly on the terminal. Simple shell commands are used for terminal-based, intuitive interaction.

Simultaneous execution of multiple security tests:
Python's concurrent modules enable the execution of multiple scans in parallel, significantly reducing time.

Automated PDF report generation:
Provides automated PDF report generation directly from Python scripts, saving auditors significant documentation time.

6.4.3 Tools used

Security Testing Tools:
1. Nmap:
Important for the project as it is widely used for network exploration, security scanning, port scanning and vulnerability detection.
Helps auditors quickly identify open ports, services running on those ports, service versions and potential vulnerabilities.

2. Wireshark:
tool for packet analysis and deep packet inspection. Useful for validating network traffic during analysis stages or for detailed packet-level inspection.

System and Network Analysis Tools (Built-in Terminal Commands):
netstat: Used to check for active network connections and open ports.
Nmap: Scan the entire network for active devices.
Tcpdump: Capture and analyze network packets.
nmap & nectar: Discover open ports on a target host.
iptables (Linux) / pfctl (MacOS): To check firewall rules and firewall configuration status.
ping: To check network connectivity and latency.
nslookup/dig: Performs DNS lookups, supports DNS resolution and domain analysis.
ifconfig / ip: Provides detailed network interface details such as IP addresses, subnet masks, and interface status.
sudo, find: To identify vulnerable SUID/SGID files on Unix/Linux systems.

Automated reporting tools:
FPDF: Automates the creation of PDF reports directly in the Python environment, easily producing consistent and audit documentation.

Development tools:
Visual Studio Code: The recommended integrated development environment (IDE) for Python and shell scripting, offering plugins and debugging tools that streamline coding and development workflows.

6.5 Third Party Components and Libraries

Third-party build options help software developers expand product features while making processes easier and more dependable. These core third-party components and libraries need assessment during your security testing process:

1. Nmap

Used for network exploration, port scanning, and vulnerability detection to scan open ports, discover active hosts on the network, identify services running on those ports, and identify vulnerabilities related to network services and configurations.

Integrated using the python-nmap library.

python-nmap:

A Python-based integrated library that allows you to seamlessly execute Nmap commands from within Python scripts.

Simplifies the use and automation of Nmap directly from Python scripts.

2. FPDF (Python Library - PyFPDF)

A Python library used to programmatically generate PDF documents.

Provides simple methods for generating PDF reports directly in Python.

Improves the efficiency of audit documentation, eliminating manual reporting errors and inconsistencies.

3. OpenSSL/OpenSSH

Verifies and tests SSL/TLS and SSH configurations to assess encryption and security practices.

Essential industry-standard tools for validating and analysing secure communication protocols and authentication mechanisms.

4.

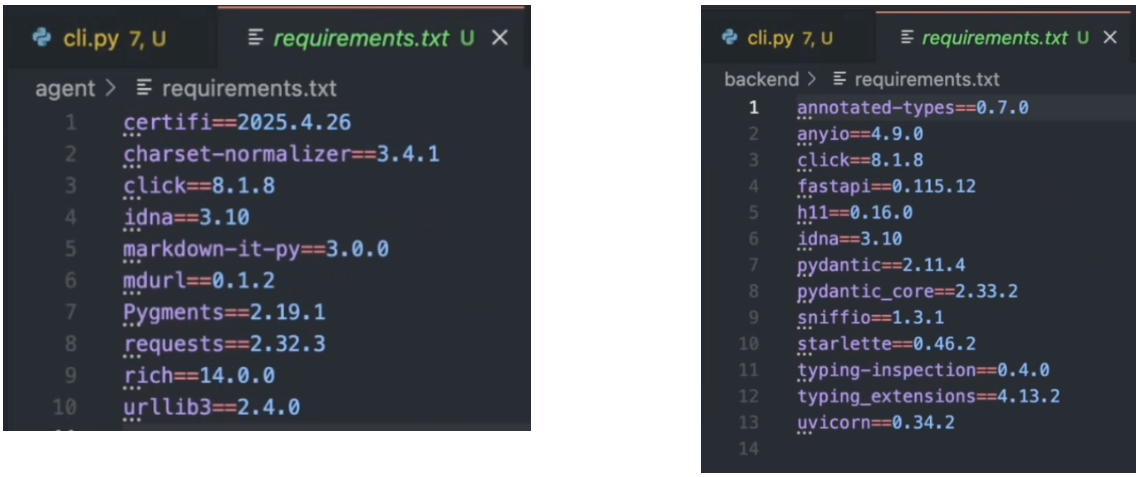


Figure 23 python libraries

These libraries are commonly used in a FastAPI-based web application, For use Web framework & ASGI Server, Data validation & Typing, ASGI Utilities, Networking, CLI Tools, Markdown & Syntax Highlights.

6.6 Outline Budget

No Budget Allocation for this Project. No Budget Allocation for the Project due to the use of open-source tools and technologies, no hardware costs, a standalone study/research-based project, and no staff or external development cost.

7. Requirements

7.1 Functional Requirements

The focus is on the basic features and behaviors that the system must provide to achieve project goals. These requirements are grouped into the following key areas.

7.1.1 Configuring and starting a scan

The system allows an auditor to specify scan parameters. For example, the target IP range, port range, protocols, or vulnerable plug-ins can be specified. The goal is to give auditors fine-grained control over the devices, networks, or applications to be assessed.

A. Select the IP range or specific system components to select the scope of the scan.
Enter the IP range or specific system components to be scanned.
Enter the ports to be scanned.

B. Select the scan types.
The application is useful for multiple security tests that can be run individually or in combination.
Identifies vulnerable open ports.
Checks for known vulnerabilities in applications, software versions, and system configurations
Reviews security settings, firewall rules, and system hardening measures.
Prevents unnecessary scans on large or partitioned networks, speeding up the process.

Start Scan **:-**
After configuring the configurations, auditors can start scanning.

A. Launch Scan
Initiates the scan through the user interface.

B. Monitor Scan
Provides updates for scans that are running in the application.
Auditors can pause, resume, or cancel ongoing scans.

C. Handle Scan Results
Once complete, the scan generates a detailed security report.
Allows users to export findings as a PDF report.

Benefits include efficient manual effort during security audits and multiple simultaneous tests.
Scheduled scanning and reporting reduce human error in identifying vulnerabilities.

7.1.2 Vulnerability scanning

Multiple security tests are run, reducing audit time and providing a comprehensive assessment of the target environment

Network and application-level scanning **-**
The system should scan network hosts and related applications for security vulnerabilities. This includes identifying open ports, services, and versions, and checking for outdated software or misconfigurations.

An auditor selects a target IP range to scan and checks each host for active services, their versions, and any known vulnerabilities associated with those versions.

Configurable scan types -
The system should allow auditors to choose different scanning methods based on their needs.
An auditor selects a quick port scan to verify which systems are alive, and then performs a full vulnerability scan that includes.

Real-time scan progress and status

- The system will display real-time progress for scans, such as percentage completed, time remaining, and critical findings.
- During a large-scale network audit, the system updates the auditor every few seconds with the number of hosts completed and the critical vulnerabilities found so far.
- Progress updates appear without having to refresh the page.
- The advantage of this is that it helps auditors manage their time and prioritize urgent vulnerabilities before a full scan is complete.

7.1.3 Automatic report generation

Comprehensive report creation -
A scan should be completed and the system should provide a complete audit report. The report includes risk summaries, severity levels, and recommended remediation steps.
Once a company's entire subnet is scanned, a PDF is automatically generated summarizing high-risk vulnerabilities in one area and low-risk issues in another. The advantage of this is that consistent, standardized reporting eliminates repetitive manual tasks for auditors.

Export and Sharing -
The exported file format maintains consistency. This makes it easier for auditors to distribute findings and allows stakeholders to review them immediately.

7.1.4 Performance and Scalability

- The system should handle scans without crashing.
- It should manage resources efficiently when performing large scans across multiple subnets.
- Many real-world networks have many hosts and services, and the system needs to be able to respond under stress.

7.1.5 Reliability and Reporting Accuracy

- The system should verify results with external vulnerability databases.
- Where applicable, the application can cross-check known vulnerabilities for improved accuracy.
- The system generates consistent and accurate reports.
- Scan results should match the actual vulnerabilities in the environment. The modeling should be consistent across repeated use.

7.2Non-Functional Requirements

7.2.1 Performance Requirements

Concurrent Scan Throughput -
The system provides parallel running capabilities for vulnerability scans that maintain both high speed and responsiveness. Running scans over networks with 50–100 hosts should complete their process within less than 2 hours when standard network conditions prevail.

Data Processing Speed -
The system needs to manage at least 1000 recorded vulnerabilities from individual scans without delays between them. The system needs to generate complete reports about at least 1,000 detected vulnerabilities in each scan run during processing time.

2. Reliability and Availability Requirements
Generation Time -
Between the scheduled audit period the application must operate with minimal service interruptions.

The system should resume operations within 5 to 6 minutes when it unexpectedly shuts down after recording the latest scan state.

Fault tolerance -
Errors that cause scans to fail should trigger application logging which enables users to either begin a scanned anew or resume their work from the point of failure. The solution needs to report clearly what results were obtained instead of presenting blank results whenever essential checks fail to execute.

Error recovery -
The system must attempt automatic recovery after it detects failure conditions.
Auditors must document all incomplete scans so they can refer to them in the future.

7.2.2 Security requirements

Secure data storage -
Secure protocols need implementation for all network connections to run between back-end modules and clients, as well as between server systems and clients. For example, TLS/HTTPS
API endpoints that handle scan management must use authentication tokens together with secure session cookies to stop unauthorized access.

7.2.3 Usability requirements

Ease of use -
The user interface should guide non-technical or moderately technical auditors through the process. From selecting scan targets to generating reports, demonstrate with minimal complexity.

7.2.4 Maintainability

Modular architecture -
A modular application design provides the application core with loosely coupled components that enable module updates independently from other parts of the system. Plugins and third-party security solutions can be integrated as new modules to the system design.

Configuration Management -
A system of centralized version management exists for all configuration files to enable change tracking. Specification of scanning parameters and user preferences along with licensing details takes place in the system.

All configurations need to be differentiated between default settings and user-modified options by the system design.

7.3 Hardware and Software Requirements

7.3.1 Software Requirements

1. Operating System
   - Host OS: Linux-based distribution.
   - Many security-oriented tools have native Linux support and can be easily configured on Linux.

2. Programming Languages

Python (3.x) is the main language for background logic for scanning, data processing, and report generation.
The main language used for background logic is pip. Pip is available for dependency management.

Python libraries used:

- Two Nmap-specific Python libraries enable interaction with Nmap.
- Our system uses FPDF for creating automatic PDF reports.
- The programming package includes network functions through sockets and allows communication with the operating system using os and subprocess modules.

3. Security and Scanning Tools
   - The Nmap tool is essential for port scanning, service/version detection.

4. Security and Access Control
   - If multiple auditors are using the tool, you may need authentication and role-based access.
   - In a firewall configuration, allow outbound scan traffic on the host operating system.

7.3.2 Hardware Requirements

Test machine: A stable network connection is required to perform port scans, etc.

Test network environment: virtual network interfaces to simulate real-world environments.
Test servers: To perform a system check on a Linux operating system.

8. End-Project Report

Project Summary:

The Basic Security Testing Application project was developed to make the security assessment process efficient and automated for IT auditors. The application combines both system and network scanning functionalities, enabling quick identification of vulnerabilities in IT environments. It reduces the reliance on manual testing by automating the scanning process and generating a detailed report. This aims to minimize human errors. The user-friendly interface design of this tool supports accessibility for both beginner and advanced technical users.

Key features include:

- Network scanning: Port scanning, DNS lookup, active connections, and packet analysis.
- System scanning: Firewall checks, kernel module reviews, SUID/SGID file detection, and database audits.
- Wireless security: WPA/WPA2 testing.
- Automatic, customizable report generation.

The application was developed using leveraged tools such as Python, Nmap, Python-nmap, and fpdf libraries.

Achievement

- Developed a functional security testing tool with a user-friendly GUI.
- Implemented both network and system scan modules.
- Integrated WPA/WPA2 security testing for wireless audits.
- Automated PDF reporting system to reduce manual report writing.
- Delivered a functional solution in compliance with IT audit and cybersecurity standards.

Project Objective

- To create a user-friendly interface and design with simplicity and accessibility in mind.
- To generate reports that are explained to the client's requirements, automatically executed using FPDF. The aim is to minimize human errors.
- To enable concurrent scanning processes. Designed to save time by scanning multiple components at once.
- The application can be used with minimal technical knowledge.

Realization of Business Objectives

The application successfully fulfils the business need for an IT audit tool. It provides the following business benefits:

- Time Saving: Reduces audit time by automating scanning and report generation.
- Accuracy of reports: Generates standardized reports, minimizing human errors.
- Cost Effectiveness: Eliminates the need for complex commercial tools or high training costs.
- Improving audit quality: Improves the accuracy and coverage of risk identification.

Changes and Their Impact

- Enhancements in User Interface: Simplified the tool's interface, which improved user satisfaction.
- The inclusion of the WPA/WPA2 test module following stakeholder feedback highlighted the importance of wireless network security.

- PDF report improvements: Improved structure and formatting for better readability.

Conclusion

The Basic Security Testing Application project has solved an essential problem for IT auditing through its development. An efficient security assessment tool is needed which combines accuracy with ease of use for audit testing operations. The application solves technical requirements along with business needs by unifying crucial system scanning abilities with network-assessment automation and operating through an easy-to-use interface. The tool's application value grows stronger due to its wireless security testing and real-time scan execution features for modern IT audit settings.

The end product consists of an expandable and functional system which helps IT auditors carry out their daily work and advances organizational security measures.

9. Project Post-Mortem

Overview

This post-mortem analysis reflects on the recently completed project to develop security audits in the IT industry in Sri Lanka. The project focused on creating a simplified security testing application for IT auditors. It aimed to reduce manual workloads, improve accuracy, and generate a report. The application was successfully developed with both system and network scanning interacting with the user interface, delivering real-time results and audit reports.

Evaluation of Project Objectives

The product was well-specified in relation to the security objectives. The primary objectives, including automation, simultaneous testing, and ease of use, were fully achieved. Reports were generated without human input. It addressed the core needs of the IT industry, such as improving operational efficiency, Specifications, and alignment with project objectives.

Product specifications: The product was well-defined in relation to the business objectives.
The application aligned well with the goals of the IT audit firms, such as improving operational efficiency and guest satisfaction. Minimizing audit time, reducing human error, and providing a scalable solution for risk scanning.

Development Process

Choice of Development Process: The Agile development process facilitated flexibility and responsiveness to changes. Nevertheless, more structured phases with stringent checkpoints before advancing could have prevented some scope creep and misaligned feature prioritization.

Technological Choices

Technology Suitability: The technologies chosen (React for frontend, Python for backend) were suitable for a scalable application. However, the decision to use certain third-party libraries without extensive testing led to integration issues that delayed the deployment

Personal Performance

Self-Reflection: My performance in terms of timeline management and resource allocation was problematic. This project significantly improved my skills in Python, network security, and user interface design.

Debugging complex scans and understanding WPA2 protocols was initially difficult but eventually became manageable through learning.

Client Feedback

Customer Feedback: Customer feedback was generally positive, with appreciation for the application's user-friendly interface and improved operational efficiency. However, there were a few suggestions for additional features and improvements that could be considered in future updates. The tool provided good feedback for its simplicity and automation capabilities.

Lessons Learned for the Future

- Planning and communication are key to successful project management. Clear objectives, regular updates, and proactive engagement with stakeholders can lead to better outcomes.
- Testing and quality assurance should be prioritized throughout the development process to identify and address issues early.
- Flexibility and adaptability are essential in a dynamic project environment. Being open to changes and feedback can lead to a more successful result.

The basic security testing application was an end-to-end solution that met both technical and business requirements. A tool can significantly improve IT audit efficiency.

10. Future Implementation

The future development of the basic security testing application requires the implementation of four key points that emphasize better performance and ease of use and extensibility.

1. Integration with AWS cloud infrastructure
Deploying the application on AWS will provide improved scalability with better manageability along with the benefits available. The application deployment combines EC2 hosting with S3 secure storage and integrates with AWS services for monitoring. The architectural transition creates a versatile platform that supports remote-based auditing and secure database operations for remote workers.

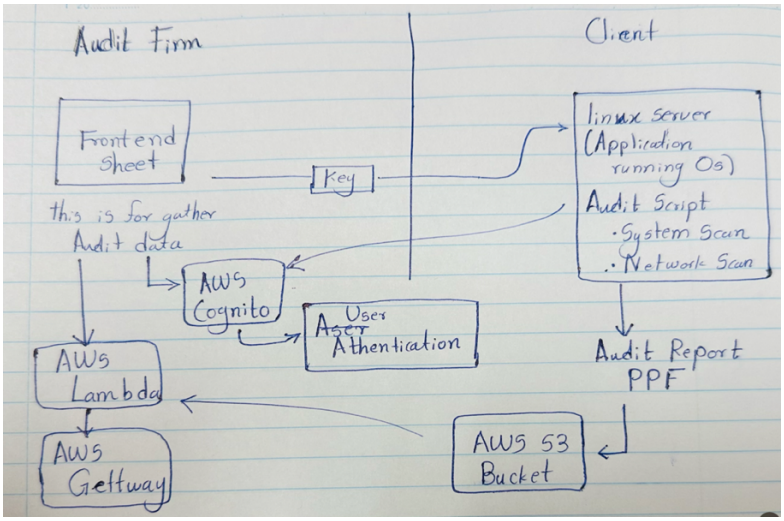


Figure 24 Integration with AWS

2. Efficient database integration
Implementing an optimized database structure will enable efficient log processing and storage in addition to result data and report information. Fast data lookups combined with improved performance during concurrent user operations along with historical audit result tracking are essential for both evaluation requirements and regulatory compliance.

3. Enhanced Reporting Features
Upcoming releases of the application will provide the following enhanced reporting features:

- Customizable report templates
- The application will include functionality for users to rank risks when creating reports.
- The software will provide visual representations in the form of risk graphs and risk trends.
- Server-branded summaries
- The application will allow users to export data via CSV and HTML options.

The implemented features improve the clarity and communication of results along with the usability of the system for users across technical and non-technical professional domains.

4. Inclusion of advanced security tests
IT auditors using these advanced security modules will receive a more thorough security analysis that goes beyond basic surface-level tests.

11. Conclusions

In conclusion, this project aimed to develop a comprehensive security application that caters to the IT Auditor in the IT Industry. The Basic Security Auditing application represents a significant development in the IT auditing field, addressing long-standing challenges of inefficiency, technical complexity, and inconsistent reporting. By integrating modern development tools such as Python, FastAPI, React, and PostgreSQL, the project offers a scalable, secure, and automated application tailored for IT auditors. Its main innovation lies in automating basic security auditing tasks, including system and network scanning, while maintaining an intuitive interface that lowers the barrier to entry, especially for less technically experienced auditors.

The application uses Nmap for network diagnostics and the FPDF library for report generation, which reduces manual work and provides consistent report formats. The project adheres to best practices by incorporating operational feasibility into its architecture and ensuring technical consistency across different environments. Its modular, terminal-based design promotes flexibility, with the integration of additional features and support for future technologies such as machine learning and cloud-based deployments envisioned. Adopting an Agile methodology ensured continuous development, quick adaptation to stakeholder feedback, and rapid prototyping of key modules such as PDF reporting and wireless audit capabilities.

From a business perspective, the tool improves audit speed, improves report accuracy, and reduces the costs associated with complex commercial software and extensive training. Client feedback accepted the solution positively because users found the system easy to operate and received clear reports while the audit process became more streamlined.

The Core Security Testing application fulfills technological specifications together with organizational business requirements through its efficient, reliable cost-effective security auditing solution. It empowers IT auditors to perform complete and efficient assessments with minimal overhead, thereby strengthening the organizational cybersecurity posture and compliance readiness. The developed project creates a baseline that future designers can use to develop automated security auditing solutions

Reference

[1] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of Vulnerabilities in Internet Firewalls," Mar. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404803003109

[2] E. C. Lo and M. Marchand, "SECURITY AUDIT: A CASE STUDY," Niagara Falls, ON, Canada, May 2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1344989

[3] K. D. Jadhav, "THE ROLE OF CYBER SECURITY AUDITS THE ROLE OF CYBER SECURITY AUDITS IN MANAGING COMPANY SYSTEMS AND APPLICATIONS," Jan. 2023. [Online]. Available: https://www.researchgate.net/publication/367559332_THE_ROLE_OF_CYBER_SECURITY_AUDITS

[4] G. Murali, M. Pranavi, Y. Navateja, and K. Bhargavi, "NETWORK SECURITY SCANNER," 2011. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/90636070/2af5cc555cae555660bb4226fab380a43594-libre.pdf?1662272260=&response-content-disposition=inline%3B+filename%3DNetwork_security_scanner.pdf&Expires=1730206193&Signature=JXcwMhMP5aGKSFIRlA03XQmd--AeCJvEeeji9C3Sm76svApm~d-yaAif7MZnAU9SirtkTktlpyJ8ZjUrgmdlzFDyh50YpRfAC2ILLx7suGETBBY0eJhX5Gg2iZ6uwuxYutWHZfUHSJgL1HmpGJfTZAM7rkieuNaZYohC8MZoNLCkdyyt3b1YySaB7TBSzaSUwYLsp33k1u4y~NHyUas~FfU-ry8k4jjEGpFH~pcIusDMOdsBh6OWyFENMjPeXENllyyX0OvWA8P4eC5stKRlHBO1nWQxRA3di94Y1J9y24ck8EAJdgLyVD650rmeOwgbBj3IKacbakgDgFhp1SPluQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[5] A.-M. Suduc, M. Bîzoi, and F. Gheorghe FILIP, "Audit for Information Systems Security," 2010. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8fe27c55f5298a4a67a8204f4596b3609440556e

[6] A. Aarthy Devi, A. K. Mohan, and M. Sethumadhavan, "Wireless Security Auditing: Attack Vectors and Mitigation Strategies," in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 674–682. doi: 10.1016/j.procs.2017.09.153.

[7] Bayu Rima Aditya; Ridi Ferdiana; Paulus Insap Santosa, *Toward Modern IT Audit– Current Issues And Literature Review*. Yogyakarta, Indonesia: IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8528627

[8] P. Boyanov, "A COMPREHENSIVE SCANNING FOR OPEN, CLOSED AND FILTERED PORTS IN THE COMPUTER SYSTEMS AND NETWORKS," *Original Contribution Journal scientific and applied research*, vol. 23, 2022.

[9] Jayant Gadge and Anish Anand Patil, *Port Scan Detection*. New Delhi, India: I E E E, 2008. Accessed: Feb. 02, 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4772622

[10] V. N. Sayankar and V. N. Sayankar, "A Review on Information Systems Audit," *Research J. Engineering and Tech*, vol. 4, no. 3, Sep. 2013, [Online]. Available: https://www.researchgate.net/publication/381653679_A_Review_on_Information_Systems_Audit

[11] S. Rao Vemula, “Automating Security Testing: Strategies for Vulnerability Scanning, Penetration Testing, and Compliance Checks,” *International Research Journal of Engineering and Technology (IRJET) e-International Research Journal of Engineering and Technology*, Jul. 2024, [Online]. Available: www.irjet.net

[12] S. Rao Vemula, “Automating Security Testing: Strategies for Vulnerability Scanning, Penetration Testing, and Compliance Checks,” *International Research Journal of Engineering and Technology (IRJET) e-International Research Journal of Engineering and Technology*, Jul. 2024, [Online]. Available: www.irjet.net

[13] J. P. Seara and C. Serrão, “Intelligent System for Automation of Security Audits (SIAAS),” *EAI Endorsed Transactions on Scalable Information Systems*, vol. 11, no. 1, Oct. 2024, doi: 10.4108/eetsis.3564.

[14] T. Vieira and C. Serrão, “Web Applications Security and Vulnerability Analysis Financial Web Applications Security Audit-A Case Study,” Dec. 2016. [Online]. Available: https://repositorio.iscte-iul.pt/bitstream/10071/12991/5/Web-Applications-Security-and-Vulnerability-Analysis-Financial-Web-Applications-Security-Audit%E2%80%93A-Case-Study.pdf

[15] A. Ziro, S. Toibayeva, S. Gnatyuk, A. Imanbayev, M. Iavich, and Z. Zhaybergenova, “Research of the Information Security Audit System in Organizations,” in *SIST 2023 - 2023 IEEE International Conference on Smart Information Systems and Technologies, Proceedings*, Astana, Kazakhstan: IEEE, May 2023, pp. 440–444. doi: 10.1109/SIST58284.2023.10223557.

Appendices

User Guide

1. installed (on macOS via Homebrew):

- brew install python@3.11
- node postgresql nmap tcpdump bind
- brew services start postgresql

2. Project Setup

- Unzip and enter the project directory

3. Database setup

- Create PostgreSQL role and database
- Apply the database schema
- Set up environment configuration

4. Start the Backend API

- Go to backend app directory.
- Create a python virtual environment named. venv in the current directory.
- Activate the virtual environment. Command - source .venv/bin/activate
- Upgrade the pip installer.
- Install the required backend libraries. Command use- pip install fastapi uvicorn databases[postgresql] sqlalchemy asyncpg python-dotenv reportla

5. Launch the API Server
- uvicorn app.main:app --reload --host 0.0.0.0 --port 8000

6. Start the front-end
- Go to the front-end directory
- Install npm
- Start npm

7. Register Account
- Register as you are IT Auditor. Fill the required details about your IT Auditor's details. After auto generate an auditor's code.

8. Then Go to the terminal, Install & Run the CLI Agent
- cd ../agent
  python3 -m venv .venv
  source .venv/bin/activate
  pip install --upgrade pip
  pip install click requests rich reportlab psutil

9. In the agent terminal run this command,
- python cli.py \
  --auditor-code " " \
  --server-url http://localhost:8000 \
  --client-company " " \
  --it-manager " "

10. Then go to the login page. Enter the auditor's code given. Then, view the audit report or download a report as a .pdf.

Network Scan

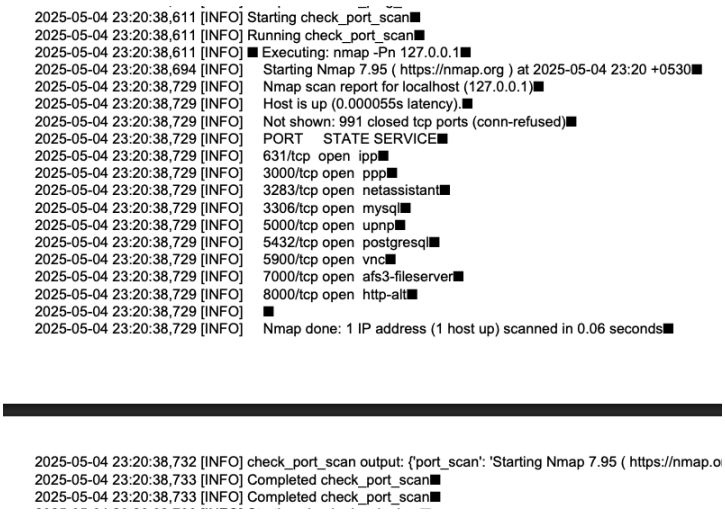


Figure 25 check_port_scan

```
2025-05-04 23:20:35,529 [INFO] ■ Executing: ping -c 4 8.8.8.8■
2025-05-04 23:20:35,591 [INFO]    PING 8.8.8.8 (8.8.8.8): 56 data bytes■
2025-05-04 23:20:35,591 [INFO]    64 bytes from 8.8.8.8: icmp_seq=0 ttl=115 time=57.692 ms■
2025-05-04 23:20:36,601 [INFO]    64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=62.886 ms■
2025-05-04 23:20:37,601 [INFO]    64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=59.265 ms■
2025-05-04 23:20:38,610 [INFO]    64 bytes from 8.8.8.8: icmp_seq=3 ttl=115 time=67.636 ms■
2025-05-04 23:20:38,610 [INFO]    ■
2025-05-04 23:20:38,610 [INFO]    --- 8.8.8.8 ping statistics ---■
2025-05-04 23:20:38,610 [INFO]    4 packets transmitted, 4 packets received, 0.0% packet loss■
2025-05-04 23:20:38,610 [INFO]    round-trip min/avg/max/stddev = 57.692/61.870/67.636/3.825 ms■
2025-05-04 23:20:38,610 [INFO] check_ping_test output: {'ping_test': 'PING 8.8.8.8 (8.8.8.8): 56 data bytes\
2025-05-04 23:20:38,610 [INFO] Completed check_ping_test■
2025-05-04 23:20:38,610 [INFO] Completed check_ping_test■
```

Figure 26 check_ping_test

```
2025-05-04 23:20:38,733 [INFO] Starting check_dns_lookup■
2025-05-04 23:20:38,733 [INFO] Running check_dns_lookup■
2025-05-04 23:20:38,733 [INFO] ■ Executing: dig +short example.com■
2025-05-04 23:20:39,070 [INFO]    23.192.228.84■
2025-05-04 23:20:39,071 [INFO]    23.215.0.136■
2025-05-04 23:20:39,071 [INFO]    23.215.0.138■
2025-05-04 23:20:39,071 [INFO]    96.7.128.175■
2025-05-04 23:20:39,071 [INFO]    23.192.228.80■
2025-05-04 23:20:39,071 [INFO]    96.7.128.198■
2025-05-04 23:20:39,073 [INFO] check_dns_lookup output: {'dns_lookup': ['23.192.228.84', '23.215.0.136', '
2025-05-04 23:20:39,073 [INFO] Completed check_dns_lookup■
2025-05-04 23:20:39,073 [INFO] Completed check_dns_lookup■
```

Figure 27 check_dns_lookup

```
2025-05-04 23:20:39,073 [INFO] Starting check_active_connections■
2025-05-04 23:20:39,073 [INFO] Running check_active_connections■
2025-05-04 23:20:39,073 [INFO] ■ Executing: ss -tunap■
2025-05-04 23:20:39,076 [ERROR] Error in check_active_connections: [Errno 2] No such file or directory: 's
```

Figure 28 check_active_connections

```
2025-05-04 23:20:39,076 [INFO] Starting check_packet_analysis■
2025-05-04 23:20:39,076 [INFO] Running check_packet_analysis■
2025-05-04 23:20:39,076 [INFO] ■ Executing: tcpdump -c 20 -nn -i any■
2025-05-04 23:20:39,090 [INFO]    tcpdump: any: You don't have permission to perform this capture on that
2025-05-04 23:20:39,090 [INFO]    (Attempt to open /dev/bpf0 failed - root privileges may be required)■
2025-05-04 23:20:39,091 [INFO] check_packet_analysis output: {'packet_capture_sample': "tcpdump: any: Y
2025-05-04 23:20:39,091 [INFO] Completed check_packet_analysis■
2025-05-04 23:20:39,091 [INFO] Completed check_packet_analysis■
```

Figure 29 check_packet_analysis

```
2025-05-04 23:20:39,091 [INFO] Starting check_traceroute■
2025-05-04 23:20:39,091 [INFO] Running check_traceroute■
2025-05-04 23:20:39,091 [INFO] ■ Executing: traceroute 8.8.8.8■
2025-05-04 23:20:39,096 [INFO]    traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 40 byte packets■
2025-05-04 23:20:39,186 [INFO]     1  192.168.8.1 (192.168.8.1)  3.383 ms  3.795 ms  2.032 ms■
2025-05-04 23:20:54,202 [INFO]     2  * * *■
2025-05-04 23:21:09,217 [INFO]     3  * * *■
2025-05-04 23:21:24,228 [INFO]     4  * * *■
2025-05-04 23:21:24,362 [INFO]     5  125.214.190.2 (125.214.190.2)  93.164 ms■
2025-05-04 23:21:24,508 [INFO]        125.214.190.8 (125.214.190.8)  38.991 ms  56.537 ms■
2025-05-04 23:21:24,743 [INFO]     6  125.214.191.33 (125.214.191.33)  49.048 ms  45.168 ms  49.538 ms
2025-05-04 23:21:24,948 [INFO]     7  125.214.191.32 (125.214.191.32)  113.107 ms■
2025-05-04 23:21:25,041 [INFO]        125.214.191.36 (125.214.191.36)  90.944 ms  91.702 ms■
2025-05-04 23:21:25,168 [INFO]     8  125.214.162.151 (125.214.162.151)  38.906 ms■
2025-05-04 23:21:25,350 [INFO]        142.250.163.116 (142.250.163.116)  85.502 ms  75.168 ms■
2025-05-04 23:21:30,479 [INFO]     9  142.250.163.116 (142.250.163.116)  71.232 ms  52.212 ms *■
2025-05-04 23:21:35,678 [INFO]    10  dns.google (8.8.8.8)  70.701 ms *  119.527 ms■
2025-05-04 23:21:35,678 [INFO] check_traceroute output: {'network_traceroute': 'traceroute to 8.8.8.8 (8.8.8
2025-05-04 23:21:35,678 [INFO] Completed check_traceroute■
2025-05-04 23:21:35,678 [INFO] Completed check_traceroute■
```

Figure 30 check_traceroute

System Scan

```
2025-05-04 23:21:35,678 [INFO] Starting check_firewall_scan■
2025-05-04 23:21:35,678 [INFO] Running check_firewall_scan■
```

```
2025-05-04 23:21:35,678 [INFO] ■ Executing: pfctl -sr■
2025-05-04 23:21:35,683 [INFO]     pfctl: /dev/pf: Permission denied■
2025-05-04 23:21:35,683 [INFO] check_firewall_scan output: {'firewall_rules': 'pfctl: /dev/pf: Permission deni
2025-05-04 23:21:35,683 [INFO] Completed check_firewall_scan■
2025-05-04 23:21:35,683 [INFO] Completed check_firewall_scan■
```

Figure 31 check_firewall_scan

```
2025-05-04 23:21:35,683 [INFO] Starting check_system_users■
2025-05-04 23:21:35,683 [INFO] Running check_system_users■
2025-05-04 23:21:35,683 [INFO] ■ Executing: cat /etc/passwd■
2025-05-04 23:21:35,688 [INFO]     ##■
2025-05-04 23:21:35,688 [INFO]     # User Database■
2025-05-04 23:21:35,688 [INFO]     #■
2025-05-04 23:21:35,689 [INFO]     # Note that this file is consulted directly only when the system is runni
2025-05-04 23:21:35,689 [INFO]     # in single-user mode.  At other times this information is provided by■
2025-05-04 23:21:35,689 [INFO]     # Open Directory.■
2025-05-04 23:21:35,689 [INFO]     #■
2025-05-04 23:21:35,689 [INFO]     # See the opendirectoryd(8) man page for additional information abou
2025-05-04 23:21:35,689 [INFO]     # Open Directory.■
2025-05-04 23:21:35,689 [INFO]     ##■
2025-05-04 23:21:35,689 [INFO]     nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false■
```

Figure 32 check_system_users

```
2025-05-04 23:21:35,691 [INFO] Starting check_kernel_modules■
2025-05-04 23:21:35,691 [INFO] Running check_kernel_modules■
2025-05-04 23:21:35,691 [INFO] ■ Executing: kextstat■
2025-05-04 23:21:35,698 [INFO]     Executing: /usr/bin/kmutil showloaded■
2025-05-04 23:21:35,729 [INFO]     No variant specified, falling back to release■
2025-05-04 23:21:35,908 [INFO]     Index Refs Address            Size       Wired      Name (Version) UUID <Li
2025-05-04 23:21:35,908 [INFO]       1  220 0                  0          0          com.apple.kpi.bsd (24.4.0) A86C2
2025-05-04 23:21:35,908 [INFO]       2   11 0                  0          0          com.apple.kpi.dsep (24.4.0) A86C2
2025-05-04 23:21:35,908 [INFO]       3  248 0                  0          0          com.apple.kpi.iokit (24.4.0) A86C2
2025-05-04 23:21:35,908 [INFO]       4  250 0                  0          0          com.apple.kpi.libkern (24.4.0) A86
2025-05-04 23:21:35,908 [INFO]       5  234 0                  0          0          com.apple.kpi.mach (24.4.0) A86C
2025-05-04 23:21:35,908 [INFO]       6  168 0                  0          0          com.apple.kpi.private (24.4.0) A86
2025-05-04 23:21:35,908 [INFO]       7  152 0                  0          0          com.apple.kpi.unsupported (24.4.(
2025-05-04 23:21:35,908 [INFO]       8   19 0xfffffe0007bbc7c0 0x193a0    0x193a0    com.apple.kec.corec
2025-05-04 23:21:35,908 [INFO]       9    0 0xfffffe0007401ef0 0x51f      0x51f      com.apple.kec.AppleEnc
```

Figure 33 check_kernel_modules

```
2025-05-04 23:21:35,968 [INFO] Starting check_password_policies■
2025-05-04 23:21:35,968 [INFO] Running check_password_policies■
2025-05-04 23:21:35,968 [INFO] check_password_policies output: {'password_policies': 'Password policy ch
2025-05-04 23:21:35,968 [INFO] Completed check_password_policies■
2025-05-04 23:21:35,968 [INFO] Completed check_password_policies■
```

Figure 34 check_password_policies

```
2025-05-04 23:21:35,968 [INFO] Starting check_suid_sgid_files■
2025-05-04 23:21:35,968 [INFO] Running check_suid_sgid_files■
2025-05-04 23:21:35,968 [INFO] ■ Executing: find / -xdev -perm +4000■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/top■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/atq■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/crontab■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/atrm■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/newgrp■
2025-05-04 23:21:36,036 [INFO]     /usr/bin/su■
```

Figure 35 check_suid_sgid_files

PID DOCUMENT

IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Pasidu Adithya Liyanage
Student Reference Number: 10899322

Module Code: PUSL3190	Module Name: Computer Project
Coursework Title: PID Document	
Deadline Date: 06/01/2025	Member of staff responsible for coursework: Dr. Pabudi Aberathne
Programme: BSc (Hons) Computer Security	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed: 

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

Overall mark _____% Assessors Initials _____ Date_____

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

1. Introduction

The "Basic Security Test Application" is intended to meet the growing demand for efficient, accurate, and user-friendly solutions for IT auditors who do vulnerability assessments. As cyberattacks get more complex, organisations require reliable processes to examine and secure their systems. This project streamlines vulnerability assessment and reporting while minimising reliance on complicated tools, making it usable even by non-technical auditors.

The project's inspiration comes from the issues that IT audit firms confront, such as lengthy assessment times, discrepancies in manual testing, and the high learning curve of existing tools like Nmap or OpenVAS. By automating essential procedures, the program seeks to improve cybersecurity practices, minimise risk, and promote trust between organisations and their stakeholders.

2. Business Case

Business Need

Modern IT auditing demands a balance between speed and precision. Manual vulnerability assessments are prone to errors, time-consuming, and resource-intensive. Complex software tools, while powerful, often require significant technical expertise, limiting their usability for general auditors. The "Basic Security Test Application" addresses these challenges by:
Automating key security checks.
Simplifying report generation.
Providing intuitive user interfaces.
These features will empower IT auditors to conduct thorough, efficient, and accurate audits, ultimately enhancing organizational cybersecurity resilience.

Business Objectives

The primary objectives include:

- **Efficiency:** Reduce the time required for audits by automating vulnerability scanning.
- **Accuracy:** Minimize human errors through automated processes and detailed reporting.
- **Accessibility:** Make security auditing tools usable for individuals with varying technical expertise.
- **Cost-effectiveness:** Lower operational costs for IT auditing firms by streamlining processes

3. Project Objective

- **Quick, Automated Method to Scan for System Vulnerabilities:**

The major goal of the project is to develop a program that can quickly and automatically scan company systems for vulnerabilities. Traditional vulnerability scanning approaches frequently rely on manual steps, which can be slow and costly. By automating the procedure, the Basic Security Test Application enables auditors to complete audits in a fraction of the time while maintaining accuracy.

Automation will ensure that all critical sections of the system are properly checked for flaws, lowering the possibility of human mistake and detecting vulnerabilities as soon as feasible.

- **Provide an easy-to-use interface for IT auditors with minimal training:**

A primary goal is to build the program with usability in mind. Many current security tools are extremely complex, needing significant instruction before clients can utilize them effectively. The Basic Security Test Application, on the other hand, will be created with an intuitive user interface that will allow auditors to quickly learn how to use the tool without the need for specialist training.

This will make the tool more accessible to a large variety of users, allowing even individuals with less technical knowledge to conduct successful vulnerability assessments.

- **Allow multiple security tests to run simultaneously:**

Another key goal of the project is to enable IT auditors to execute various security tests simultaneously. This will significantly prevent the time needed to complete audits. Instead, the system will automatically execute numerous checks at once, ensuring that all important aspects of the system's security are evaluated simultaneously.

This capability will be especially useful for large or complex systems, where performing individual tests can take a long time. The tool's concurrent testing feature will increase audit efficiency and assist IT auditors in meeting tight deadlines.

- **Simplify audit documentation by creating detailed automated reports:**

The ultimate goal is to ensure that the application generates detailed, automatic reports following each audit. These reports will summarize the vulnerabilities discovered, prioritize them by severity, and make actionable suggestions for remedy. By automating this procedure, the technology will save auditors time while also ensuring that clients receive clear, reports following each audit.

4. Literature Review

Literature Review
To better understand the current state of IT security testing tools and auditing processes, this literature review analyses available technologies, their capabilities, and IT auditors' challenges. The environment of information systems security auditing is continually changing as technology progresses and cyberattacks become more complex. The following overview takes on research papers to highlight the importance, techniques, and problems of conducting effective information system security audits.

Security audits are crucial tools for firms to analyse their information security postures. emphasizes the need of regular audits in establishing a security baseline, allowing vulnerabilities to be detected and addressed before they are exploited [2].
Similarly, highlights that cyber security audits conduct a thorough evaluation of an organization's IT infrastructure, finding dangers and weak links in the system [3].
Focus on vulnerability audits: Describe the importance of using various tools and strategies to achieve comprehensive coverage of all components inside an information system [4], [5]. The program makes it simple for auditors to understand how the tool increases productivity, accuracy, and reliability in finding vulnerabilities and emphasizing crucial areas. IT audit processes include solutions that automate basic audit operations, including scanning for open ports, firewalls, and logging to detect unauthorized connections [6], [7]. Address an application module for wireless network security testing, which simulates potential attack pathways and assesses vulnerability [6].
The importance of port scan detection:
This reconnaissance phase includes methods like port scanning, in which attackers analyse TCP and UDP ports to find open communication channels and system weaknesses. Open ports, such as entry points in a building, can expose systems to unauthorized access, and systems with numerous open ports are generally more vulnerable. However, fewer open ports may represent a larger security risk in some configurations. [8]. This study underlines the significance of expansive testing for open ports on computer networks for assessing security [9].

Challenges and Technical Depth:
The specific nature of the skills required for these audits involves auditors having extensive technical knowledge.IS auditors require specialized abilities to effectively analyze technical observations. In order to accomplish this, the application should have a training module. Provides information and courses to help

auditors improve their technical abilities in IS auditing. Includes case studies and examples that demonstrate frequent challenges in IS auditing and how to handle them.
Creates thorough audit reports that summarize results, make actionable recommendations, and identify areas for improvement. Management uses images (charts and charts) to convey data in a clear and effective manner [10], [11].

Modular framework for security scanning:
NSS's modular configuration offers focused scanning and auditing capabilities, making it both efficient and user-friendly. It attempts to identify system vulnerabilities in the same way as NSS can find and resolve exploits and information leaks [4].
Host Scan: Used to detect live hosts.
Port Scan: Detecting open and closed ports.
Pinging and NSLookup are used to test connectivity and resolve DNS information.
User Awareness and Training:
Several articles, notably, emphasize the crucial significance of user awareness and training in improving company safety [6], [9]. Regular discussions and presentations regarding security policies may significantly improve staff knowledge and compliance, lowering the likelihood of human errors that result in security breaches.
Integration with Governance:
Effective IS audits help to improve corporate governance by giving management with the insights it needs to make sound decisions. [3] explores how cyber risk management frameworks can bridge the gap between technical cybersecurity practices and executive governance, establishing a security culture throughout the firm.
future Research Directions:
Investigating the usefulness of future technologies, such as artificial intelligence and machine learning, in improving auditing processes and giving predictive capabilities against potential cyber threats [5], [11].

5. Research Gap

The focus is on risk scanning technologies, automated reporting systems and simple user-friendly interfaces designed to audit companies.

Nmap and OpenVAS are widely used in IT audits to identify network vulnerabilities. However, it demands a thorough understanding of command-line interfaces, which may be difficult for non-technical auditors. The project will provide a simple, streamlined tool that can perform multiple security checks (e.g. port scanning and vulnerability verification) simultaneously, allowing auditors to perform audits quickly without requiring extensive technical knowledge. Common vulnerabilities can improve the basic security testing application's capacity to efficiently find security issues.
Gap detection: While tools like Nmap and OpenVAS provide extensive scanning capabilities, they are difficult for non-technical people to understand. Provides IT auditors with a simple application without overshadowing essential security information.
Existing security testing applications provide perfect technical results but an automated reporting system aims to automatically identify problems and provide the necessary guidance in a short period of time.
By using these principles, the initial security testing application can significantly reduce audit timeframes and increase the efficiency of IT audit processes, providing auditors with a comprehensive security assessment in a fraction of the time required for sequential testing. Gap detection: An automated reporting system detects hazards while providing simple, clear and actionable information. Reports should be simple to understand and directly applicable to an audit setting, with technical terminology understandable to auditors or clients.
The next goal of the project is to create a user-friendly interface that allows auditors to perform tests and generate reports with minimal effort. The interface will be user-friendly, reducing the need for extensive instructions and making the audit process more efficient.

Wireshark, for example, is an industry-standard network protocol analyzer, but its interface can be intimidating to new users with deep packet inspection.
Gap detection: IT auditors will benefit from a system with a simple interface that facilitates risk testing and reporting.
These systems automate report generation, reducing the need for manual report writing, which can be time-consuming and error prone. Automated reports that prioritize risks based on severity and provide remediation recommendations have been shown to accelerate client decision-making and increase overall audit efficiency.

6. Method of Approach

Methodology

The project adopts an Agile methodology to ensure iterative development and continuous feedback. Core steps include:
Requirements gathering and system design.
Development of scanning, reporting, and UI components.
Testing and refinement based on stakeholder feedback.

Tools and Frameworks

- **Backend:** Python (with libraries like python-nmap and FPDF for PDF generation).
- **Frontend:** Simple, accessible UI with clear navigation.
- **Scanning Libraries:** Integration with tools like Nmap for network vulnerability assessments.
- **High-Level Architectural Diagram**
- **Input Layer:** User configures tests.
- **Processing Layer:** Backend scans for vulnerabilities.
- **Output Layer:** Automated PDF report generation.

Conceptual Diagram

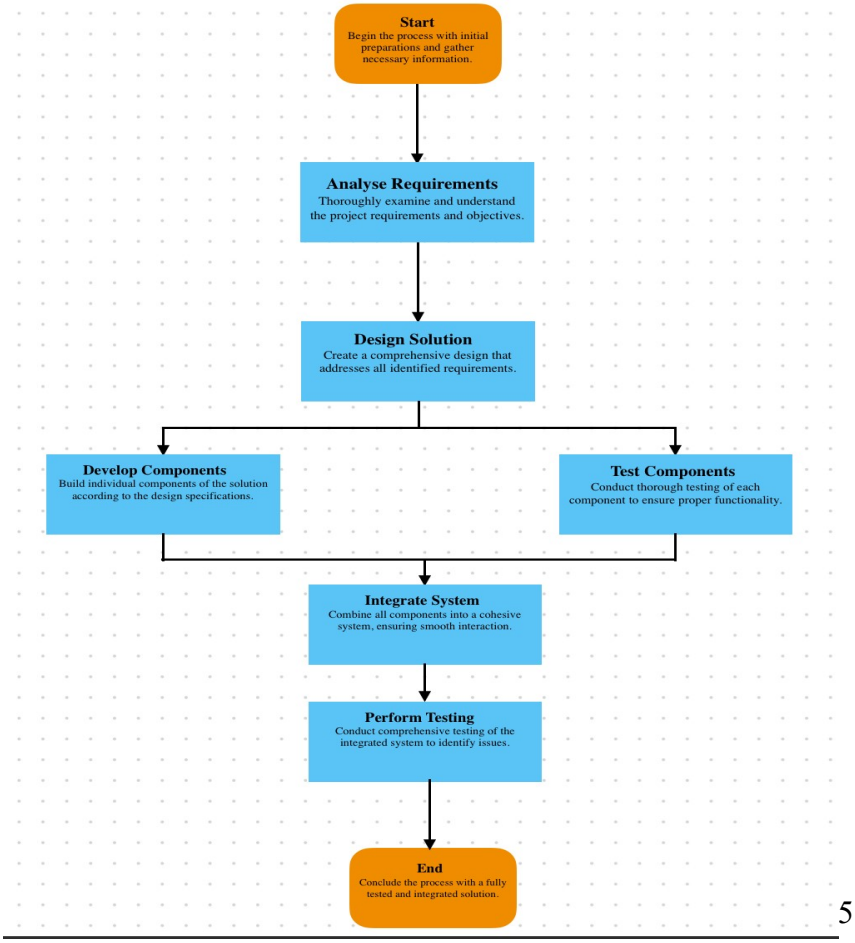


Figure 36 Conceptual Diagram

High Level Architectural Diagram

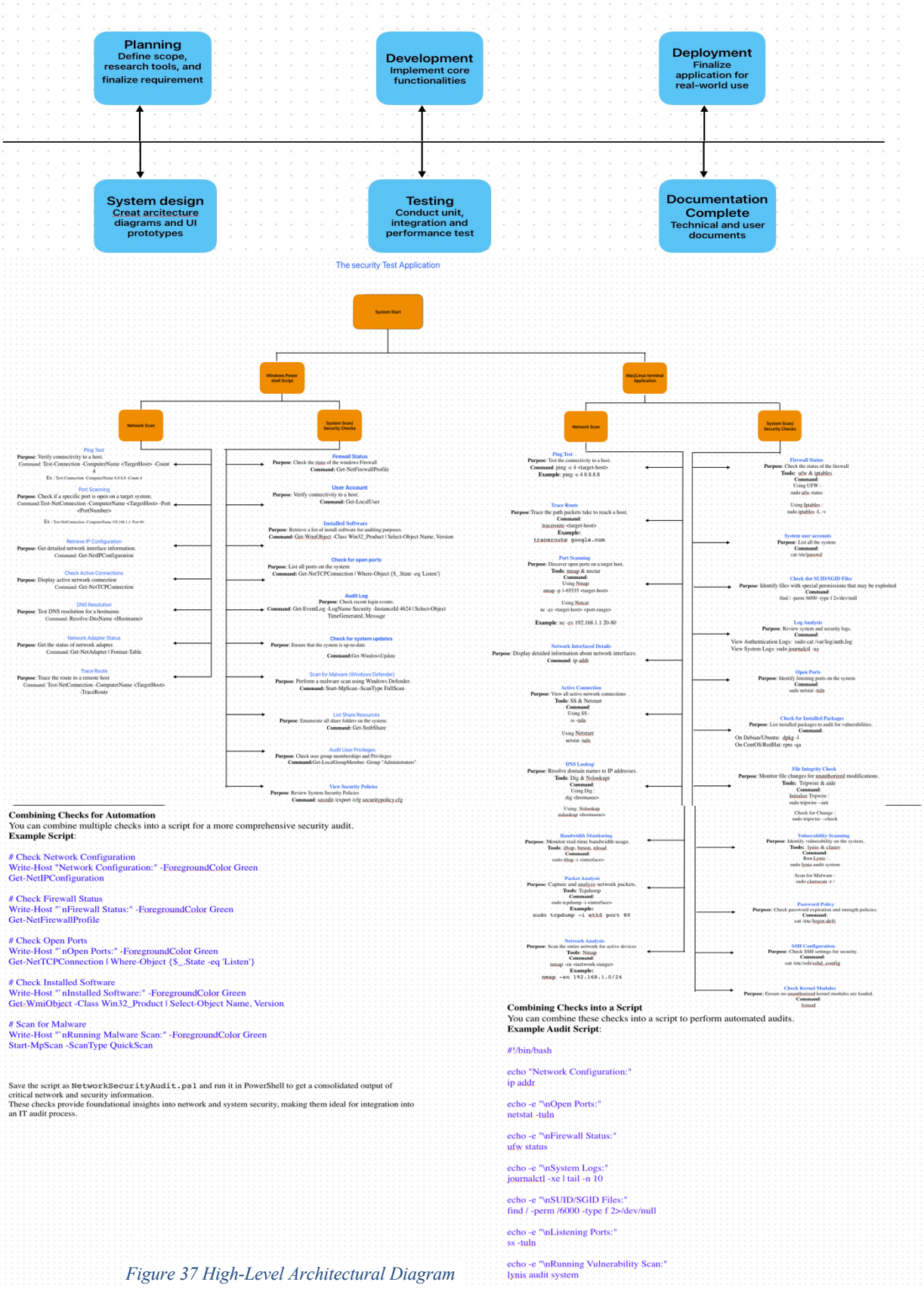


Figure 37 High-Level Architectural Diagram

project timeline

1. Planning **(October - November)**

Activities include setting project goals and parameters, conducting preliminary research and literature analysis, and compiling a list of the necessary tools and resources. Conduct a literature review on relevant cybersecurity frameworks, vulnerability scanning tools, and IT auditing procedures. Research and System Design.

Outcome: This phase gives you a foundation of information, allowing you to determine the project's goals and ensure your solution meets the specific demands of IT auditors.

2. System Design **(November - December)**

Activities include designing and developing the system architecture, as well as providing simple user interface designs. System architecture takes two weeks, while wireframe design takes four weeks.

- Determine the overall structure of the application.
- Divide the system into basic components such as the scanning tool, automatic reporting and user interface.
- Choose the technologies and libraries (such as Python, FPDF, and Nmap) that will be used to build each module.

3. Phase of Development **(November - December)**

This phase focuses on developing your system design into a working application. You'll start by developing the application's basic functionality, making sure that the main modules are correctly integrated. Create the vulnerability scanning module, which will enable the program to perform a variety of security tests. Create the user interface, making it simple and intuitive for IT auditors. Implement an automatic reporting system that generates PDFs based on test results [1].

4. Testing **(December-January)**

Thoroughly test the program to ensure that it functions properly, discovers

vulnerabilities, and generates accurate reports. Tasks include doing functional and nonfunctional testing, debugging errors, and improving the system based on feedback from early tests. Evaluate the application's performance in various network situations. Outcome: A stable and optimized program that is free of major problems and ready for usage in the real world.

5. Implementation **(January-February)**

Finalise the application for use, optimize it for efficiency, and make it ready for use.

Tasks: Fine-tune the system based on testing results, improve the automatic reporting feature, and guarantee that all security tests run concurrently avoiding problems with performance.

Outcome: A fully functional, user-friendly application that works the needs of IT auditors.

6. Phase of Documentation **(February-March)**

Write the final report and provide project documentation.

Documentation Create a final project report that includes technical documentation and screenshots of the user interface. Ensure that all stages of the development process are well explained.

Outcome: A complete project report outlining your efforts and explaining how the program works.

7. Phase of Submission
   The successful submission of the final year project marks the end of the development and evaluation proces.

Gantt chart

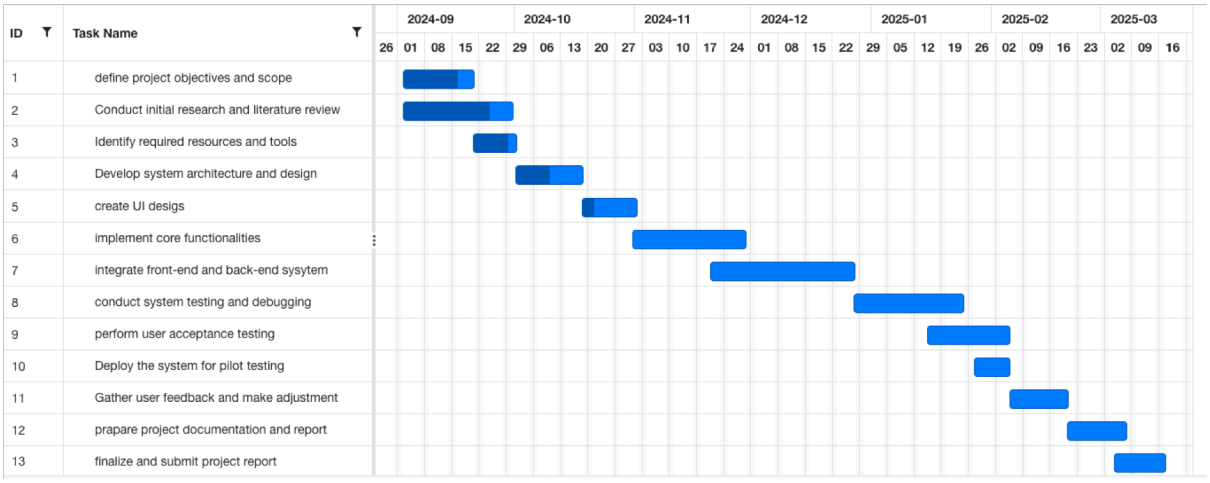


Figure 38 Gantt Chart

Reference

[1] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of Vulnerabilities in Internet Firewalls," Mar. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404803003109 https://www.sciencedirect.com/science/article/pii/S0167404803003109

[2] E. C. Lo and M. Marchand, "SECURITY AUDIT: A CASE STUDY," Niagara Falls, ON, Canada, May 2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1344989

[3] K. D. Jadhav, "THE ROLE OF CYBER SECURITY AUDITS THE ROLE OF CYBER SECURITY AUDITS IN MANAGING COMPANY SYSTEMS AND APPLICATIONS," Jan. 2023. [Online]. Available: https://www.researchgate.net/publication/367559332_THE_ROLE_OF_CYBER_SECURITY_AUDITS

[4] G. Murali, M. Pranavi, Y. Navateja, and K. Bhargavi, "NETWORK SECURITY SCANNER," 2011. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/90636070/2af5cc555cae555660bb4226fab380a43594-libre.pdf?1662272260=&response-content-disposition=inline%3B+filename%3DNetwork_security_scanner.pdf&Expires=1730206193&Signature=JXcwMhMP5aGKSFIRlA03XQmd-AeCJvEeeji9C3Sm76svApm~d-yaAif7MZnAU9SirtkTktlpyJ8ZjUrgmdlzFDyh50YpRfAC2ILLx7suGETBBY0eJhX5Gg2iZ6uwuxYutWHZfUHSJgL1HmpGJfTZAM7rkieuNaZYohC8MZoNLCkdyyt3b1YySaB7TBSzaSUwYLsp33k1u4y~NHyUas~FfUry8k4jjEGpFH~pclusDMOdsBh6OWyFENMjPeXENllyyX0OvWA8P4eC5stKRlHBO1nWQxRA3di94Y1J9y24ck8EAJdgLyVD650rmeOwgbBj3IKacbakgDgFhp1SPluQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[5] A.-M. Suduc, M. Bîzoi, and F. Gheorghe FILIP, "Audit for Information Systems Security," 2010. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8fe27c55f5298a4a67a8204f4596b3609440556e

[6] A. Aarthy Devi, A. K. Mohan, and M. Sethumadhavan, "Wireless Security Auditing: Attack Vectors and Mitigation Strategies," in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 674–682. doi: 10.1016/j.procs.2017.09.153.

[7] Bayu Rima Aditya; Ridi Ferdiana; Paulus Insap Santosa, *Toward Modern IT Audit– Current Issues And Literature Review*. Yogyakarta, Indonesia: IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8528627

[8] P. Boyanov, "A COMPREHENSIVE SCANNING FOR OPEN, CLOSED AND FILTERED PORTS IN THE COMPUTER SYSTEMS AND NETWORKS," *Original Contribution Journal scientific and applied research*, vol. 23, 2022.

Stage Plans

1. Planning & Analysis - Completed
    - Gather the requirements needed by an IT Auditor.
    - Define the project scope, objectives, and deliverables.
    - Identify key stakeholders and establish communication channels.
    - Develop a project plan, including timelines, milestones, and resource allocation.

2. Define Requirements – Completed
    - Define detailed functional and non-functional requirements for the application.

3. Design – Completed

4. Development – Completed

5. Testing – Completed
    - Perform unit tests, integration tests, and system tests.
    - Identify and resolve any bugs or issues.

6. Deployment - Completed
    - Prepare the application for deployment to the IT auditor.
    - Deploy the application to production and monitor for any issues.
    - Provide IT auditors with insight into how the application is being used.

7. Maintenance (Ongoing)
    - Provide ongoing maintenance and support for the application.
    - Monitor performance and user feedback to identify areas for improvement.
    - Implement updates and enhancements based on feedback and change.

requirements.

This stage plan outlines the key phases of the project, from initial planning and analysis to deployment and maintenance. It provides a structured approach to managing the project and ensures that all aspects of the development lifecycle are addressed.

PUSL3190 Computing Individual Project

Interim Report

The Basic Security Test Application

Supervisor: Dr. Pabudi Abeyrathne

Name: Pasidu A Liyanage
Plymouth Index Number: 10899322

Degree Program: Bsc(Hons) Computer Security

Chapter 01 – Introduction

1.1 Introduction

The Basic Security Testing Application solves a fundamental requirement for IT auditing through its reliable and efficient user-friendly risk assessment platform. Security has developed into an urgent priority because of cyber threats which highlights the necessity for accurate audit solutions. The IT auditing market confronts multiple difficulties because their procedures demand excessively long, and error-prone manual testing combined with complicated Nmap and OpenVAS tools which require steep learning curves and manual report generation is inefficient. These problems lead to lengthy audits and unfinished assessments that raise business exposure to risks and increase the chance of cyber incidents together with regulatory violations and reputational damage.

Through automation The Basic Security Testing Application completes essential risk scans which decreases operational costs and expenses and enhances the precision of audit outcomes. Security tests execute in parallel from within the application through its simultaneous running of port scans combined with vulnerability checks and firewall status checks. The system's ability to run tests simultaneously shortens evaluation durations for completing system evaluations particularly when auditing extensive complex network systems.

The platform produces detailed reports automatically which reflect advanced professional standards. The automated system brings simplicity to documentation tasks by eliminating human data compilation mistakes as well as creating standardized reports based on client needs to enhance clarity and improve decision efficiency. Auditors and clients benefit from the generated reports that categorize detected vulnerabilities according to risk levels while offering remediation solutions for resolving serious security issues.

Agile development methodologies enable the project to integrate Python programming as its backend foundation. The structured approach enables ongoing refinement through continuous testing with feedback loops to keep auditors and industries up to date. The Basic Security Testing application serves to enhance corporate cybersecurity practices through better risk assessment efficiency while improving accuracy alongside usability to address current gaps.

1.2 Problem Definition

Security audits are essential to maintaining the integrity of IT systems. However, traditional auditing techniques involve manual processes that are time-consuming and prone to human error. In this role, auditors must look for vulnerabilities in a variety of systems, including networks, software, databases, and hardware components. The lack of an efficient, automated tool significantly impacts the accuracy and speed of security assessments.

Performing risk assessments stands among IT auditors' primary responsibilities because these assessments may require thorough scans of extensive network and system systems. The evaluation follows several sequential steps that need different security testing skills including penetration testing along with network analysis and system configuration checks. System vulnerabilities require immediate attention because untested systems are directly exposed to possible threats. The market now requires a solution which allows IT auditors to conduct their examinations more swiftly without compromising precision or depth of assessment.

IT auditors encounter significant challenges due to reporting problems that occur during hands-on testing and report compilation activities. Auditors need to perform distinct independent assessments throughout the system before they generate their assessment report by collecting findings from each test. The availability of Nessus and OpenVAS tools has drawbacks since they demand complex configuration before achieving successful operation and their usage requires mastery of steep learning curves. A solution which simplifies testing operations and reporting tasks along with reducing complex procedures must be developed because of essential requirements.

The lack of an effective solution for vulnerability scanning and audit report generation can have major consequences for companies and their IT audit teams. Hackers or malicious individuals are more likely to exploit vulnerabilities. Furthermore, limitations can cause auditors to overlook critical security issues. As a result, it is essential to deploy a solution that allows IT auditors to conduct thorough, timely assessments of

system vulnerabilities, while limiting oversight risk and ensuring that companies maintain strong cybersecurity defences.

1.3 Project Objectives

Develop and introduce an automated security testing application for IT auditors:
By automating the procedure, the Basic Security Test Application enables auditors to complete audits in a fraction of the time while maintaining accuracy.
Automation will ensure that all critical sections of the system are properly checked for flaws, lowering the possibility of human mistake and detecting vulnerabilities as soon as feasible.

Improve the efficiency and accuracy of vulnerability scanning:
Another key goal of the project is to enable IT auditors to execute various security tests simultaneously. The tool's concurrent testing feature will increase audit efficiency and assist IT auditors in meeting tight deadlines.

Generate detailed, automated security reports:
The goal is to ensure that the application generates detailed, automatic reports following each audit. These reports will summarize the vulnerabilities discovered, prioritize them by severity, and make actionable suggestions for remedy. By automating this procedure, the technology will save auditors time while also ensuring that clients receive clear, professional reports following each audit.

Chapter 02 - System Analysis

2.1 Facts Gathering Techniques

The Basic Security Testing application provides multiple fact-gathering methods which let IT auditors detect and inspect vulnerabilities found in networks and software systems as well as databases while examining system configurations. The system performs automated audit work through the combination of different security scanning tools and reporting tools. The security project relies on the following main fact-gathering approaches:

1. Vulnerability Scanning Techniques
Multiple systematic scanning methods allow the application to detect existing security weaknesses during its operation.
System port scanning establishes an overview of open, closed and filtered ports in order to locate unauthorized access points.

How it works:

- Network Scanning:
  The tool displays which hosts are active with their services along network paths for security vulnerability analysis.
- Configuration Audit:
  The evaluation tool detects security weaknesses through assessment of system-level port status and default user passwords and incorrect firewall setups.
- Security Libraries:
  Network service and web application scanning can be performed through security tools including Nessus and OpenVAS.

2. Automated Security Testing Systems
The application uses industry standard security testing tools together with libraries to effectively obtain facts and data.

- Nmap serves as a networking tool that performs host discovery together with port scanning and network mapping functions.
- The system uses OpenVAS to execute comprehensive vulnerability scans which detect security problems including incorrect configurations and outdated software as well as known vulnerabilities.
- Python scripts based on Python language perform automated security assessments and collect data through security checks.

3. Data Collection and Analysis

Log File Analysis - Analyzes system logs to identify suspicious events, unauthorized access attempts, and potential breaches.
Packet Inspection - Examines network traffic at the packet level to identify anomalies and signs of an attack.
Risk Prioritization - Assigns severity levels to detected vulnerabilities based on predefined security criteria.

How it works:

- System or application logs are collected and analyzed to identify anomalies (e.g., repeated failed login attempts, unexpected service restarts).
- Can be integrated with Security Information and Event Management (SIEM) solutions if necessary.

Collected facts:

- Login attempts (both successes and failures)
- Changes in system services or user privileges
- Timeline of events that may highlight potential breaches

4. Automated reporting of collected facts
The software transforms unstructured scan results together with configurations into an easy-to-use PDF report

How it works:
The system gathers information about background scan results together with vulnerability data and configuration check results.

The FPDF library of Python automates report structure creation to produce readable documents which contain the following elements:

- List of discovered hosts and services
- Severity ratings of vulnerabilities
- Suggested remediation steps

5. Network and host scanning
The goal is to discover all hosts contained within specified network areas while determining their basic characteristics.

Through the utilization of tools such as Nmap the network segment receives a "ping" scanning operation to reveal existing hosts. The device discovery method reveals servers among other hardware components which require scanning.
Information collected:

- IP addresses of active hosts
- Port scanning and service enumeration.

2.2 Existing System

Current security assessment systems operate through different security tools targeting unique objectives. Security assessment systems deliver thorough checks but create multiple problems in the security framework.

1. Nmap

The open source tool Nmap exists primarily to perform both network discovery and port scanning functions. Nmap aids in detecting open TCP and UDP ports that allow port scanning on an intended target device. Service enumeration identifies active services through port inspections to discover the running programs HTTP and SSH and detects their version numbers.

The Python libraries python-nmap use Nmap application as back-end automation to generate network discovery reports about target hosts and services and provide unified reporting through automated input of raw scan results such as open ports and service banners.

2. OpenVAS

OpenVAS delivers complete vulnerability checks for known software flaws through its open source framework structure. OpenVAS represents an open source solution when customers seek an alternative choice to commercial tools like Nessus.

The OpenVAS API supports basic security testing via its application programming interface with automated checks alongside CLI command execution to find potential vulnerabilities. OpenVAS results form the core data evaluation for final audit reports.

4. Nessus

The proprietary vulnerability scanner Nessus provides extensive vulnerability coverage through its large database of operating system and network device plugins while allowing users to refine and integrate its built-in reporting capabilities.

Organizations that possess Nessus licenses can choose to integrate the security testing application which will merge results into a project's automated final report while enabling a comprehensive validation of vulnerabilities through open source scanner comparisons (e.g. OpenVAS).

5. Other,

Wireshark

Wireshark functions as a network traffic capture tool even though it is not a dedicated vulnerability scanner but it reveals unexpected or suspicious network communications.

FPDF or similar PDF generation libraries

Through Python-based libraries the project extracts report data from initial tool results and converts it into PDF format.

These systems are important because,

The port enumeration and network scanning capabilities of Nmap work well together while the tool lacks any extensive vulnerability detection capabilities. OpenVAS and Nessus excel in providing detailed vulnerability databases that can sort issues based on their severity values. A basic security testing application obtains comprehensive network and system and web vulnerability coverage when its tools are used together.

Comprehensive reporting: Everything from host discovery to high-level vulnerability summaries in a final report.

Organization and combination of data obtained from diverse systems enable identification of exposed ports along with their corresponding business ramifications and severity levels.

2.3 Use case diagram

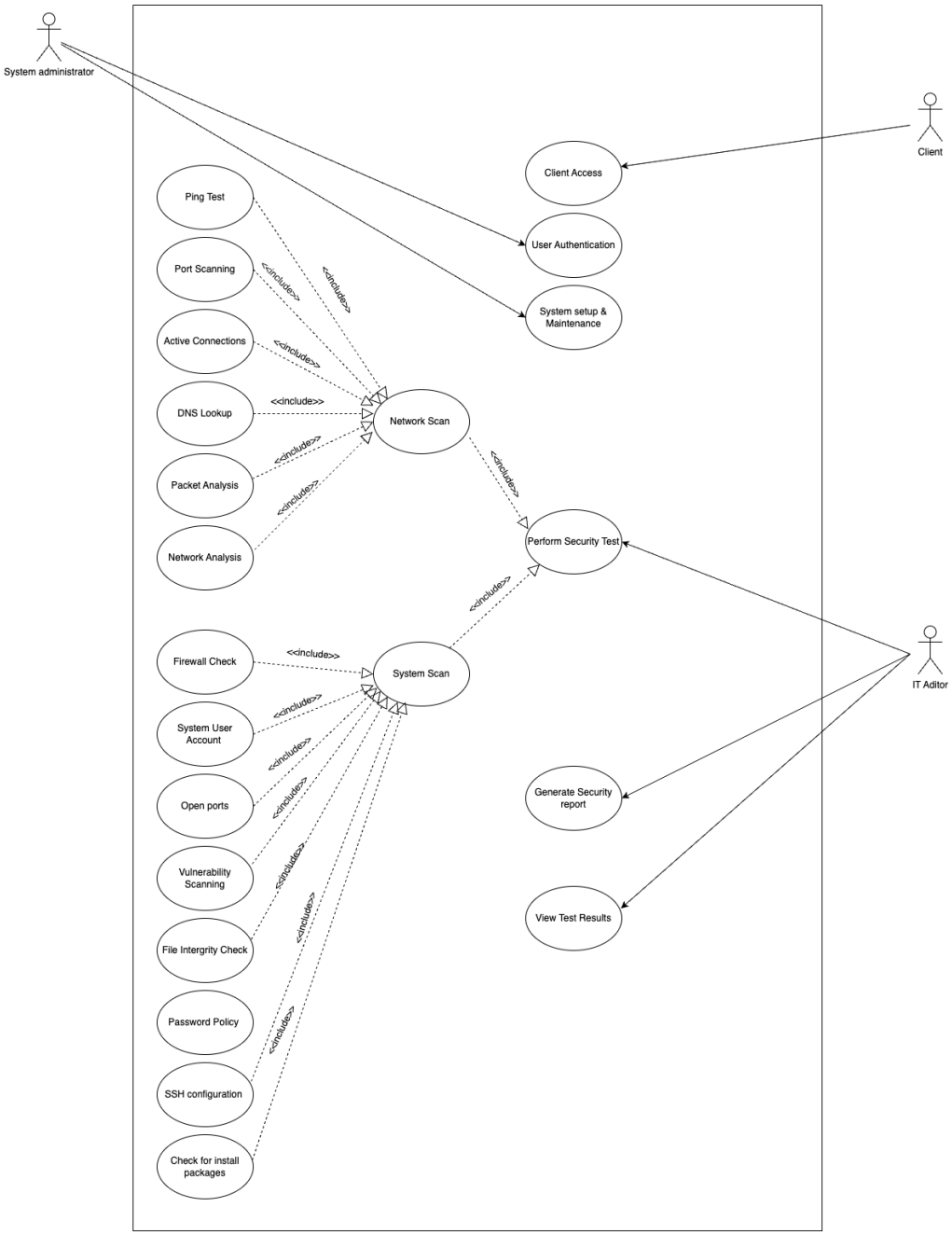


Figure 39 User case diagram

2.4 Drawbacks of the existing system

Within the project proposal readers will find an extensive breakdown regarding the boundaries of current vulnerability-scanning alongside auditing systems.

Time-consuming manual processes:
Auditors currently need to execute different tools consecutively through their existing systems since port scanning tools require manual succession with other vulnerability scanners. Manually integrated results from security tests need to be combined into one end product. The manual process of data consolidation during audits brings an increased chance of human mistakes while making the audit process exceedingly slower.

Reporting solutions that are not integrated with each other and not automated:
Professional vulnerability scanners deliver detailed information but their exports generate unprocessed or partially processed data. Auditors dedicate substantial time to convert raw outputs from vulnerability scanners into reports that satisfy clients' needs. The ability to use scripts for automation exists yet many organizations lack the capabilities and technical know-how to develop their own programmed connections. Auditors waste numerous hours on scripting to process and normalize scan data because basic automated report generation capabilities are absent from their scanner tools.

Potential Gaps in Coverage
Multiple distinct specialized tools selected by auditors for network scans and system configurations and web application testing can result in protection gaps that individual scanners may fail to detect. A single scanning tool may perform well within open-port detection while lacking capability in misconfiguration assessment which creates incomplete system security visualization results.

Human Error and Inconsistency
The need for manual intervention throughout multiple steps in existing scanning/reporting methods creates opportunities for human mistakes which surface most prominently during raw result consolidation and vulnerability priority determination. The use of inconsistent report formatting between different clients or projects hinders overall tracking of progress as well as hinders result comparison across projects.

Chapter 03 - Requirements Specification

3.1 Functional Requirements

The focus is on the basic features and behaviors that the system must provide to achieve project goals. These requirements are grouped into the following key areas.

3.1.1 Configuring and starting a scan

The system allows an auditor to specify scan parameters. For example, the target IP range, port range, protocols, or vulnerable plug-ins can be specified. The goal is to give auditors fine-grained control over the devices, networks, or applications to be assessed.

A. Select the IP range or specific system components to select the scope of the scan.
Enter the IP range or specific system components to be scanned.
Enter the ports to be scanned.

B. Select the scan types.
The application is useful for multiple security tests that can be run individually or in combination.
Identifies vulnerable open ports.
Checks for known vulnerabilities in applications, software versions, and system configurations
Reviews security settings, firewall rules, and system hardening measures.
Prevents unnecessary scans on large or partitioned networks, speeding up the process.

Start Scan **:-**
After configuring the configurations, auditors can start scanning.

A. Launch Scan
Initiates the scan through the user interface.

B. Monitor Scan
Provides updates for scans that are running in the application.

Auditors can pause, resume, or cancel ongoing scans.

C. Handle Scan Results
Once complete, the scan generates a detailed security report.
Allows users to export findings as a PDF report.

Run Automated Scan -
The system integrates with existing SIEM tools for further analysis.

Benefits include efficient manual effort during security audits and multiple simultaneous tests.
Scheduled scanning and reporting reduces human error in identifying vulnerabilities.

3.1.2 Vulnerability scanning

Multiple security tests are run, reducing audit time and providing a comprehensive assessment of the target environment

Network and application-level scanning -
The system should scan network hosts and related applications for security vulnerabilities. This includes identifying open ports, identifying services and versions, and checking for outdated software or misconfigurations.

An auditor selects a target IP range to scan and checks each host for active services, their versions, and any known vulnerabilities associated with those versions.

Configurable scan types -
The system should allow auditors to choose different scanning methods based on their needs.
An auditor selects a quick port scan to verify which systems are alive, and then performs a full vulnerability scan that includes.

Real-time scan progress and status

- The system will display real-time progress for scans, such as percentage completed, time remaining, and critical findings.
- During a large-scale network audit, the system updates the auditor every few seconds with the number of hosts completed and the critical vulnerabilities found so far.
- Progress updates appear without having to refresh the page.
- The advantage of this is that it helps auditors manage their time and prioritize urgent vulnerabilities before a full scan is complete.

3.1.3 Automatic report generation

Comprehensive report creation -
A scan should be completed and the system should provide a complete audit report. The report includes risk summaries, severity levels, and recommended remediation steps.
Once a company's entire subnet is scanned, a PDF is automatically generated summarizing high-risk vulnerabilities in one area and low-risk issues in another. The advantage of this is that consistent, standardized reporting eliminates repetitive manual tasks for auditors.

Export and Sharing -
The system should export reports in standard formats and allow sharing via email or a secure link.
An auditor generates a PDF and immediately emails it to the client's IT manager for review.

The exported file format maintains consistency. This makes it easier for auditors to distribute findings and allows stakeholders to review them immediately.

3.1.4 Performance and Scalability

- The system should handle scans without crashing.
- It should manage resources efficiently when performing large scans across multiple subnets.
- Many real-world networks have many hosts and services, and the system needs to be able to respond under stress.

3.1.5 Reliability and Reporting Accuracy

- The system should verify results with external vulnerability databases.
- Where applicable, the application can cross-check known vulnerabilities for improved accuracy.
- The system generates consistent and accurate reports.
- Scan results should match the actual vulnerabilities in the environment. The modeling should be consistent across repeated use.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

Concurrent Scan Throughput -
The system provides parallel running capabilities for vulnerability scans that maintain both high speed and responsiveness. Running scans over networks with 50–100 hosts should complete their process within less than 2 hours when standard network conditions prevail.

Data Processing Speed -
The system needs to manage at least 1000 recorded vulnerabilities from individual scans without delays between them. The system needs to generate complete reports about at least 1,000 detected vulnerabilities in each scan run during processing time.

2. Reliability and Availability Requirements
Generation Time -
Between the scheduled audit period the application must operate with minimal service interruptions.

The system should resume operations within 5 to 6 minutes when it unexpectedly shuts down after recording the latest scan state.

Fault tolerance -
Errors that cause scans to fail should trigger application logging which enables users to either begin a scanned anew or resume their work from the point of failure. The solution needs to report clearly what results were obtained instead of presenting blank results whenever essential checks fail to execute.

Error recovery -
The system must attempt automatic recovery after it detects failure conditions.
Auditors must document all incomplete scans so they can refer to them in the future.

3.2.2 Security requirements

Secure data storage -
The encrypted storage of scan results together with user credentials and configuration details should be a part of the framework. For example, AES-256.
Secure protocols need implementation for all network connections to run between back-end modules and clients as well as between server systems and clients. For example, TLS/HTTPS

API endpoints that handle scan management must use authentication tokens together with secure session cookies to stop unauthorized access.

3.2.3 Usability requirements

Ease of use -
The user interface should guide non-technical or moderately technical auditors through the process. From selecting scan targets to generating reports, demonstrate with minimal complexity.

3.2.4 Maintainability

Modular architecture -
A modular application design provides the application core with loosely coupled components that enable module updates independently from other parts of the system. Plugins and third-party security solutions can be integrated as new modules to the system design.

Configuration Management -
A system of centralized version management exists for all configuration files to enable change tracking. Specification of scanning parameters and user preferences along with licensing details takes place in the system.

All configurations need to be differentiated between default settings and user-modified options by the system design.

3.3 Hardware / Software Requirements

3.3.1 Hardware requirements required for this project:

Test machine : A stable network connection is required to perform port scans, vulnerability scans, etc.

Test network environment:Switches, routers, or virtual network interfaces to simulate real-world environments.
Test servers : To perform a system check on a Linux operating system.

3.3.2 Software requirements required for this project:

1. Operating System
   - Host OS: Linux-based distribution.
   - Many security-oriented tools have native Linux support and can be easily configured on Linux.

2. Programming Languages
Python (3.x) is the main language for background logic for scanning, data processing, and report generation. The main language used for background logic is pip. Pip is available for dependency management.

Python libraries used:
   - Two Nmap-specific Python libraries enable interaction with Nmap.
   - Our system uses FPDF for creating automatic PDF reports.
   - The programming package includes network functions through sockets and allows communication with the operating system using os and subprocess modules.

3. Security and Scanning Tools
   - The Nmap tool is essential for port scanning, service/version detection.
   - Optional tools include Wireshark (for packet analysis and troubleshooting), Metasploit (for deep penetration testing scenarios - to verify vulnerabilities you've found it primarily).

4. Security and Access Control
   - If multiple auditors are using the tool, you may need authentication and role-based access.

- In a firewall configuration, allow outbound scan traffic on the host operating system.

3.4 Networking Requirements (Optional)

Systems require internet access. Security scans run in test conditions find all system weaknesses that would not impact operating systems during production.

3.4.1 Network Scanning Capabilities

The system is capable of performing network scans across a variety of environments, including local area networks (LANs), wide area networks (WANs) but the following network scanning features are essential:

IP Range Scanning-
This scans multiple IP addresses or an entire subnet to identify active hosts and open ports.

Port Scanning -
Scans open, closed, and filtered ports on a system to identify potential vulnerabilities.
Finds running services and assesses whether they are vulnerable, for example, outdated SSH or FTP servers.

Example: Open ports such as port - 22 (SSH), port - 80 (HTTP), port - 443 (HTTPS), and port - 389 (RDP).

Service and version detection -
Testing needs to identify the exact service versions that run the application to check for insecure older versions.
Secure your computer by using both firewall and IDS evasion methods.
A scanning technique divides network packets into segments to escape particular firewall protections and study security status.
An IDS/IPS system finds attack signals to discover if it recognizes scanning efforts.

Recommended network components-
Network segmentation controls need testing software to connect with diverse network areas. Adjust the testing area into separate parts to check the system's functioning with different security barriers. Basic system functions to connect and direct scans across multiple subnets should be added. This approach makes it possible to test networks of different scale to find out scan effectiveness and speed.

3.4.2 Identifying Network Security Processes.

These security steps need to be put in place to stop new security flaws from appearing in the application.
The system uses SSL/TLS to protect scan results sent through networks and connects to remote databases through VPN or SSH secure connections.

Network scans can create so many connection problems that control methods will reduce this issue.
Our system provides advanced settings to both manage scan speed and request numbers so networks stay protected from overloads.

Chapter 04 Feasibility Study

4.1 Operational Feasibility

The basic security testing application's fit with workflow and suitability for IT auditors and security professional requirements represents operational feasibility. The successful implementation of this solution requires evaluation of user adaptation needs alongside deployment situation analysis and training demands together with risk analysis assessment. An operational feasibility assessment has been conducted because this terminal-based application.

4.1.1 Suitability for IT auditors and security professionals

The security testing application performs seamless scans through its basic design which makes it highly suitable for IT auditors. It is suitable for:

IT auditors and cybersecurity consultants:
Security audit automation remains the main purpose of this solution. Consultants obtain complete risk reports from this application to analyse security measures effectively.

Enterprise security teams (SOC analysts and IT administrators):
The application provides continuous security monitoring as one of its capabilities. The system performs scheduled scans which connect smoothly with current SIEM infrastructures to minimize human intervention.

4.1.2 How factors impact operational viability in detail

1. Administrator access requirements

IT auditors typically require administrator-level privileges/permissions to run extensive vulnerability scans and access critical system configurations. System-level security scans can be accessed using sudo/root.

For example, certain scans, such as port scans, vulnerability scans, or obtaining system logs, may require administrative privileges.

Auditors should have defined permissions that balance operational needs and security constraints.

2. Ease of use for IT auditors

Since the project is based on the Command Line Interface, simplicity in using terminal commands is essential.
Commands should be easy to remember, and parameters should be clear and concise.
Providing a complete help page or manual documentation is important for ease of adoption. This is what this command (--help) is used for.
The application should be easy for auditors to quickly understand the command syntax, parameters, and expected outputs.

3. Level of Automation

Your tool should automate important factors such as vulnerability scanning, port analysis, and system configuration checks.
Automation reduces manual intervention by auditors, increases accuracy, and reduces audit time.
After each scan, reports should be automatically generated in this project, clearly describing the vulnerabilities, vulnerability levels, and remediation proposals.

4. Compatibility with auditor workflow

The system should align well with the auditors' existing workflow to ensure seamless integration into audit processes.
The generated reports are exported to a common format such as PDF, CSV, or JSON for easy inclusion in server deployments.
Integration with widely used audit standards and frameworks (ISO 27001, NIST, CIS benchmarks) ensures relevance and usability within current audit methodologies.

5. Data security and confidentiality

As automated scans produce reports containing sensitive risk information, data security is therefore important. These outputs should be managed securely to prevent unauthorized disclosure and procedures should be clearly outlined.

6. System Scalability

The system needs to handle security checks on both small host systems and extensive corporate network scans smoothly.

The software needs to maintain its operating capacity while performing numerous security audits simultaneously.

Scalability makes the application flexible by handling different audit assignments that emerge across various environments.

7. How WPA/WPA2 Testing Fits into the Project:

IT auditors require efficient tools to evaluate wireless network security therefore WPA/WPA2 authentication testing functions precisely within their job responsibilities.

Saves time and effort:

Security assessments of WPA/WPA2 through automation decrease human labor requirements which brings faster and more trustworthy audits.

The application design should maintain user-friendliness which enables non-specialist IT auditors to execute detailed security testing procedures.

This system passes operational feasibility because IT auditors can conduct WPA/WPA2 network assessments through the application without encountering difficulty.

4.2 Technical Feasibility

This feasibility report discusses the technical feasibility of the basic security testing application, the auditor's technical competence, system compatibility, compliance with security standards, system resource utilization, and how to technically integrate existing security tools.

It further explores whether the project can be developed using existing technologies, tools, and expertise.

1. Auditor's Technical Competency

IT auditors should have basic proficiency in the Linux operating system.

For example, they should have knowledge of basic Linux commands such as sudo, grep, cat, nano, chmod, etc.

Basic understanding of networking principles such as TCP/IP protocols, open/close ports, firewall rules, etc.

The application should have extensive reading, including clear command examples, manuals, and sample scripts, to bridge the proficiency gaps in cybersecurity tools such as Nmap, OpenVAS, and basic penetration testing techniques.

2. System Compatibility (Server Systems)

Since the application is explicitly Linux-based, the clients should have a native Linux environment.

Alternative methods such as live USB distributions or Docker images for non-Linux environments allow auditors to easily run the application.

3. System Performance and Scalability

The application needs to execute many parallel scans without overburdening system resources including CPU and memory. System reliability under various load conditions becomes more certain through regular testing of large network systems.

The application needs optimized Python scripts to maintain system resource efficiency for big-scale scanning operations.

4. Compliance with security standards

The reporting framework should implement standards including ISO 27001 and GDPR and NIST to provide more reliable and meaningful results. The security assessment of auditors depends on standards-compliant reports for their clients to demonstrate both relevance and security compliance needs.

Standardized reports enable client organizations to understand issues easily and implement necessary remedial steps.

5. System resource utilization

The terminal-based application should optimize its resource utilization to prevent straining system resources. Check if the application maintains an efficient operational state regardless of system hardware capacities.

6. Impact of WPA/WPA2 testing.

Your application can integrate both Scapy and Wireshark tools as Wireless Packet Analysis Tools to perform wireless network authentication analysis. The implementation of libraries that establish network interface connection for performing WPA/WPA2 authentication procedures.

Since the project depends on peripherals the system needs to process WPA/WPA2 scans using available hardware without additional specifications. Through available resources of programming expertise and necessary systems and appropriate technologies the implementation of WPA/WPA2 security testing proves possible for the project.

4.3 Outline Budget

No Budget Allocation for this Project. No Budget Allocation for the Project due to use of open-source tools and technologies, no hardware costs, standalone study/research-based project, and no staff or external development costs

Chapter 04 System Architecture

5.1 Class Diagram of Proposed System

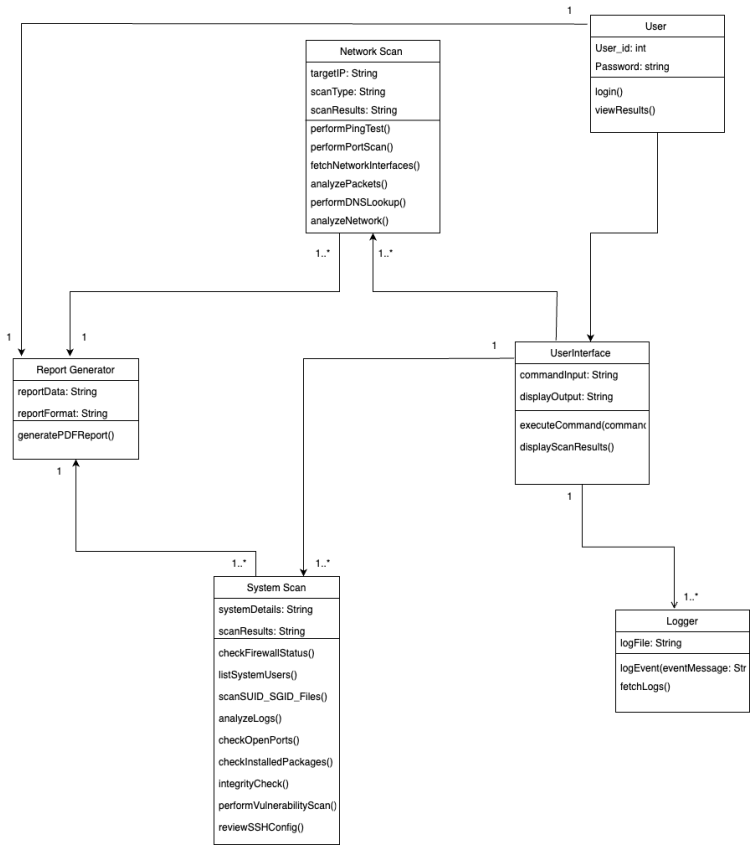


Figure 40 Class Diagram of Proposed System

5.2 ER diagram

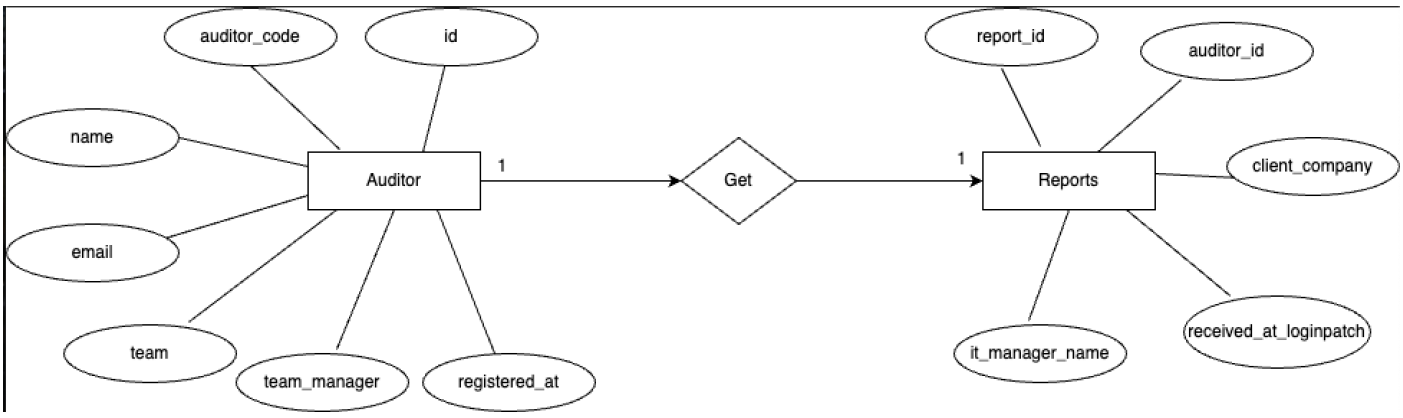


Figure 41 ER diagram

5.3 High-level Architectural Diagram

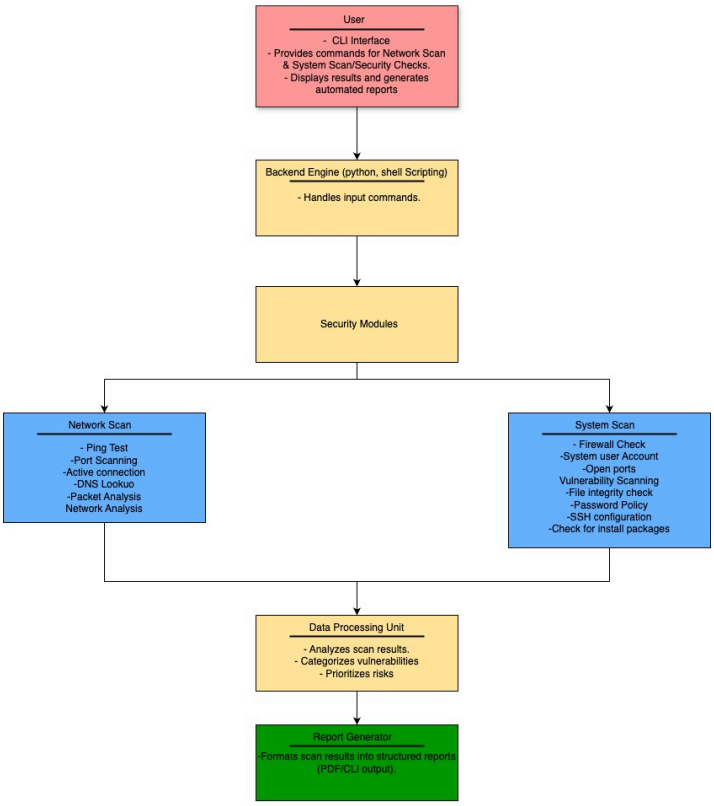


Figure 42 High-level Architectural Diagram

Chapter 06 Development Tools and Technologies

6.1 Development Methodology

The basic security testing application is developed following the Agile software development methodology. Agile software is chosen because of its iterative, flexible, and feedback-driven nature, which means continuous testing and improvement throughout the entire process. If new cybersecurity threats emerge, the application can quickly adapt and incorporate the necessary updates.

6.1.1 why Agile software is chosen

because,

- Continuous testing can identify bugs and improve security.
- Continuous iterations reduce development time due to rapid development cycles
- Improved security allows for frequent updates to allow for continuous security improvements.

6.1.2 How Agile is involved in this project

1. Development process and testing

The project is divided into manageable modules:

- Vulnerability scanning
- WPA/WPA2 security testing
- Automatic report generation etc.

Each module is developed and tested here. This minimizes the likelihood of major system failures. Here, the system is verified to be functional and useful at every stage of development.

2. Regular testing and evaluation

Cybersecurity tools should be tested to ensure accuracy and reliability.

3. Alignment with IT audit processes
IT audits require flexibility in security over the network, infrastructure.
Agile allows the tool to be customized and refined based on feedback from potential users.
If auditors need additional scanning features, they can change the system without planning.

6.1.3 How Agile connects to other aspects of the project

Project aspect Agile development impact
Features using WPA3 in future updates allow for new security testing to be added gradually.
Daily testing helps to quickly identify bugs, improve system stability. Agile allows for easy changes.

The development of the basic security testing application uses the agile development methodology. Since security testing requires flexibility, frequent testing, and incremental improvements, Agile is the best fit for the basic security testing application.

6.2 Programming Languages and Tools

The basic security testing application is developed using a combination of secure programming languages and tools. These tools have been selected to ensure that the application is efficient, scalable, and secure while supporting cross-platform deployment.

6.2.1 Programming languages used

1. Shell Scripting:
Shell scripting is a useful tool for automating routine tasks and system-level operations in Linux-based environments. It is suitable for system and network security audits.
It supports ping tests, firewall status checks, system user account verification, and basic file system analysis, such as finding SUID/SGID files.
Advantages:
Efficiency: Perform repetitive security checks such as active connection detection and firewall rule validation.
Integration and flexibility: Easily integrates with Linux/MacOS command line utilities.
For example, netstat, iptables, ping, ifconfig.

No extensive additional software installations required, lightweight terminal-based application.

- Firewall rule checking (iptables -L)
- Active network connections detection (netstat -tunap)
- System user enumeration (cat /etc/passwd)
- SUID/SGID file checks (find command) etc.

2. Python:
Python is widely used in cybersecurity due to its ease of use, extensive library support, and readability. This makes it ideal for complex, automated security applications and vulnerability scanning.

System Interaction:
Executes shell commands within Python scripts for checks such as SSH configuration, firewall status, SUID/SGID file checks, and system account validations.

6.2.2 How programming languages and tools are involved in the project:

Python:
Quickly and accurately automate complex vulnerability scanning tasks through tools like Nmap and Python scripts.

Shell scripting:
Efficiently performs scans such as ping tests, port checks, and firewall status assessments.

Easy-to-use interface:

Python scripts handle the backend complexity, presenting simple output in an easy-to-read format directly on the terminal. Simple shell commands are used for terminal-based, intuitive interaction.

Simultaneous execution of multiple security tests:
Python's concurrent modules enable the execution of multiple scans in parallel, significantly reducing time.

Automated PDF report generation:
Provides automated PDF report generation directly from Python scripts, saving auditors significant documentation time.

Shell scripting:
It is possible to preprocess logs and generate basic text reports before generating detailed PDFs via Python.

6.2.3 Tools used

Security Testing Tools:
1. Nmap:
Important for the project as it is widely used for network exploration, security scanning, port scanning and vulnerability detection.
Helps auditors quickly identify open ports, services running on those ports, service versions and potential vulnerabilities.

2. Wireshark:
tool for packet analysis and deep packet inspection. Useful for validating network traffic during analysis stages or for detailed packet-level inspection.

System and Network Analysis Tools (Built-in Terminal Commands):

- netstat: Used to check for active network connections and open ports.
- Nmap: Scan the entire network for active devices.
- Tcpdump: Capture and analyze network packets.
- nmap & nectar: Discover open ports on a target host.
- iptables (Linux) / pfctl (MacOS): To check firewall rules and firewall configuration status.
- ping: To check network connectivity and latency.
- nslookup/dig: Performs DNS lookups, supports DNS resolution and domain analysis.
- ifconfig / ip: Provides detailed network interface details such as IP addresses, subnet masks, and interface status.
- sudo, find: To identify vulnerable SUID/SGID files on Unix/Linux systems.

Automated reporting tools:
FPDF: Automates the creation of PDF reports directly in the Python environment, easily producing consistent and audit documentation.

Development tools:
Visual Studio Code: The recommended integrated development environment (IDE) for Python and shell scripting, offering plugins and debugging tools that streamline coding and development workflows.

6.3 Third Party Components and Libraries

Third-party build options help software developers expand product features while making processes easier and more dependable. These core third-party components and libraries need assessment during your security testing process:

1. Nmap

Used for network exploration, port scanning, and vulnerability detection to scan open ports, discover active hosts on the network, identify services running on those ports, and identify vulnerabilities related to network services and configurations.
Integrated using the python-nmap library.

python-nmap:
A Python-based integrated library that allows you to seamlessly execute Nmap commands from within Python scripts.
Simplifies the use and automation of Nmap directly from Python scripts.

3. FPDF (Python Library - PyFPDF)
A Python library used to programmatically generate PDF documents.
Provides simple methods for generating PDF reports directly in Python.
Improves the efficiency of audit documentation, eliminating manual reporting errors and inconsistencies.

4. OpenSSL/OpenSSH
Verifies and tests SSL/TLS and SSH configurations to assess encryption and security practices.
Essential industry-standard tools for validating and analysing secure communication protocols and authentication mechanisms.
Provides reliable methods for ensuring secure configurations and identifying cryptographic vulnerabilities.

6. Telnet
A quick way to quickly assess the status of TCP ports by attempting direct connections and verifying the status of the port and connection (open/closed).
Easily scriptable, integrates directly with Bash and Python scripts.

6.4 Algorithms

The Basic Security Testing platform uses its main algorithm to check the security health of computers through automatic testing and scoring. The scoring system forms the essential part of the security analysis process by delivering accurate results.

How the algorithm works:
1. Startup and data collection
The algorithm starts by running several predefined security tests simultaneously.
among which,
are network scans and system security checks.

2. Risk severity assessment
Each vulnerability discovered is classified according to its severity level:
This scoring is aligned with industry-standard vulnerability scoring systems such as the Common Vulnerability Scoring System (CVSS).

3. Security score calculation
The algorithm calculates an average security score using a weighted average methodology. The mathematical representation is as follows:
The numerator is the severity score added from all identified vulnerabilities.
The denominator is the highest severity score when all tested components are assumed to have critical vulnerabilities.

4. Result Classification
Based on the calculated score, the algorithm classifies the security status.

5. Automatic Report Generation
Post-scoring, the algorithm triggers an automatic report generation process.

- Detailed lists of identified vulnerabilities.
- Severity scores assigned to each vulnerability.
- Overall security assessment and recommendations for mitigation.

This automated PDF reporting feature significantly reduces manual efforts, ensures compliance, and speeds up decision-making processes.

Benefits:

- Time Efficiency: Concurrent security testing reduces audit time.
- Improved accuracy: Eliminates human errors inherent in manual testing.
- User Accessibility: Provides reports that are easy to understand for auditors with varying technical expertise.

Technology Used:

- Python for backend logic
- Integration with security tools
- Automated reporting using Python's FPDF library

Chapter 07 Discussion

Interim Report Overview

The current report shows our work to create the basic security testing system. This initiative creates a total endpoint security test solution that IT auditors and security experts can use. The project uses automation to handle necessary system scans and reporting which enhances network and system security tests. This document explains the project's system analysis techniques while presenting required specifications, feasibility assessments, system design, working tools, and development tools.

Report Summary

This document starts by describing the project's purpose in cybersecurity auditing. It finds key challenges with traditional methods of auditing and lists the difficulties caused by manual testing tools Nmap and OpenVAS. The requirements analysis explains what the tool should do and how it should work plus standard requirements for usability, accuracy and speed. The feasibility study helps determine if an updated cybersecurity tool for auditing purposes can function properly with Python scripting and automated reporting tools during operations. The system design team picked tools that enhance scanning performance plus keep the system lightweight when running Linux OS.

Challenges Faced

- Library Installation Issues: Some devices had problems installing necessary libraries when such installations failed to match existing system specifications or when essential support files were missing.
- Limited Internet Access: When systems did not have Internet connectivity, they could not update their security features nor retrieve digital threat data for time-sensitive virus checks.
- Permission restrictions: The application needs administrative rights to do complete security assessments that involve port scanning and logging view access. The security settings in those systems were too strict and prevented the application from completing its tasks.
- Performance optimization: The program needed to perform several tasks at once without slowing down the system but using resources effectively presented a challenging obstacle.

Future Plane

- User-friendly interface: A better terminal-based software needs a user-friendly interface with clear commands that teaches users how to navigate it.

- Efficient database integration: Our system manages scan results and found flaws through a lightweight database for easier monitoring and evaluation.

- Improved reporting features: The tool now makes better security reports by showing precise vulnerability information along with risk grades and recommended actions.

- Cross-platform compatibility: Although primarily developed for Linux, future iterations are aiming to include compatibility with macOS and Windows for wider use.

- Inclusion of advanced security testing: Expanding testing features to include wireless security assessments (WPA/WPA2 scanning) and malware detection enhances the tool's auditing capabilities.

Reference & Bibiography

[1] S. Kamara, S. Fahmy, E. Schultz, F. Kerschbaum, and M. Frantzen, "Analysis of Vulnerabilities in Internet Firewalls," Mar. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404803003109

[2] E. C. Lo and M. Marchand, "SECURITY AUDIT: A CASE STUDY," Niagara Falls, ON, Canada, May 2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1344989

[3] K. D. Jadhav, "THE ROLE OF CYBER SECURITY AUDITS THE ROLE OF CYBER SECURITY AUDITS IN MANAGING COMPANY SYSTEMS AND APPLICATIONS," Jan. 2023. [Online]. Available: https://www.researchgate.net/publication/367559332_THE_ROLE_OF_CYBER_SECURITY_AUDITS

[4] G. Murali, M. Pranavi, Y. Navateja, and K. Bhargavi, "NETWORK SECURITY SCANNER," 2011. [Online]. Available: https://d1wqtxts1xzle7.cloudfront.net/90636070/2af5cc555cae555660bb4226fab380a43594-libre.pdf?1662272260=&response-content-disposition=inline%3B+filename%3DNetwork_security_scanner.pdf&Expires=1730206193&Signature=JXcwMhMP5aGKSFIRlA03XQmd--AeCJvEeeji9C3Sm76svApm~d-yaAif7MZnAU9SirtkTktlpyJ8ZjUrgmdlzFDyh50YpRfAC2ILLx7suGETBBY0eJhX5Gg2iZ6uwuxYutWHZfUHSJgL1HmpGJfTZAM7rkieuNaZYohC8MZoNLCkdyyt3b1YySaB7TBSzaSUwYLsp33k1u4y~NHyUas~FfU-ry8k4jjEGpFH~pcIusDMOdsBh6OWyFENMjPeXENllyyX0OvWA8P4eC5stKRlHBO1nWQxRA3di94Y1J9y24ck8EAJdgLyVD650rmeOwgbBj3IKacbakgDgFhp1SPluQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[5] A.-M. Suduc, M. Bîzoi, and F. Gheorghe FILIP, "Audit for Information Systems Security," 2010. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8fe27c55f5298a4a67a8204f4596b3609440556e

[6] A. Aarthy Devi, A. K. Mohan, and M. Sethumadhavan, "Wireless Security Auditing: Attack Vectors and Mitigation Strategies," in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 674–682. doi: 10.1016/j.procs.2017.09.153.

[7] Bayu Rima Aditya; Ridi Ferdiana; Paulus Insap Santosa, *Toward Modern IT Audit– Current Issues And Literature Review*. Yogyakarta, Indonesia: IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8528627

[8] P. Boyanov, "A COMPREHENSIVE SCANNING FOR OPEN, CLOSED AND FILTERED PORTS IN THE COMPUTER SYSTEMS AND NETWORKS," *Original Contribution Journal scientific and applied research*, vol. 23, 2022.

[9] Jayant Gadge and Anish Anand Patil, *Port Scan Detection*. New Delhi, India: I E E E, 2008. Accessed: Feb. 02, 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4772622

[10] V. N. Sayankar and V. N. Sayankar, "A Review on Information Systems Audit," *Research J. Engineering and Tech*, vol. 4, no. 3, Sep. 2013, [Online]. Available: https://www.researchgate.net/publication/381653679_A_Review_on_Information_Systems_Audit

Appendixes

Gathering information from those involved in the industry and those not involved.

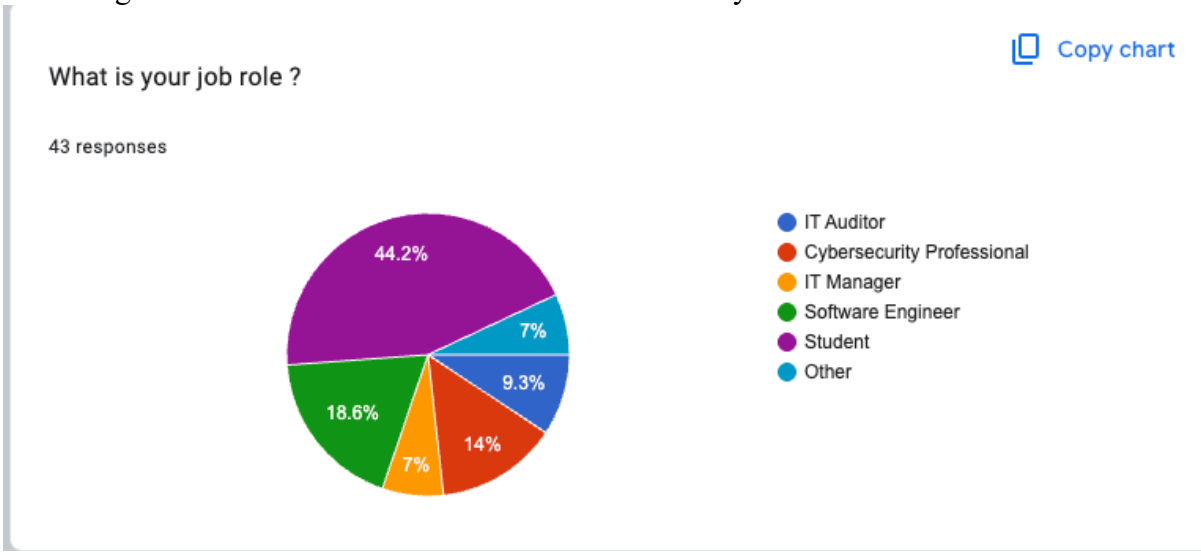


Figure 43 Job Roles in IT Industry

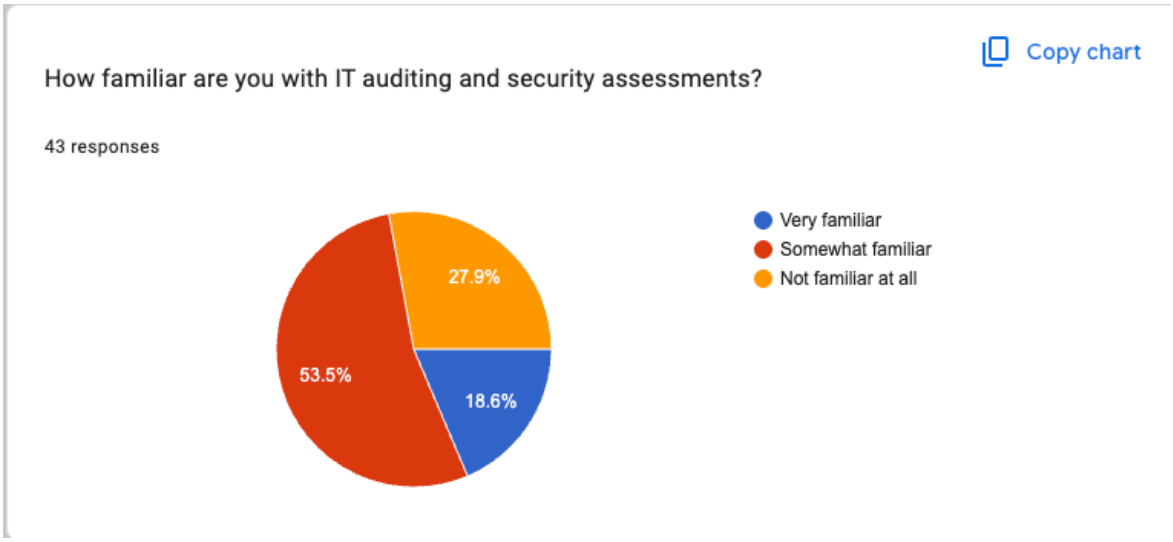


Figure 44 Gather Information of IT Audit

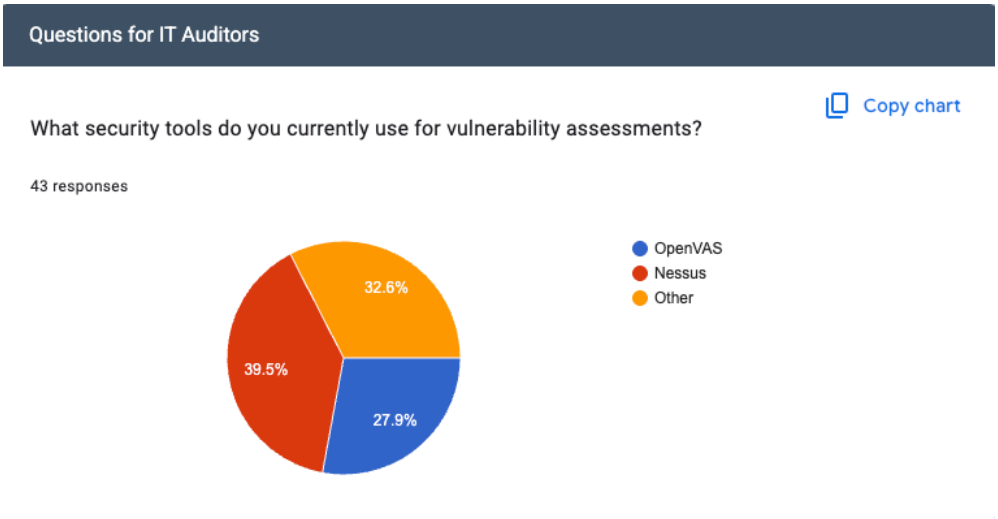


Figure 45 Gather Information of security tools

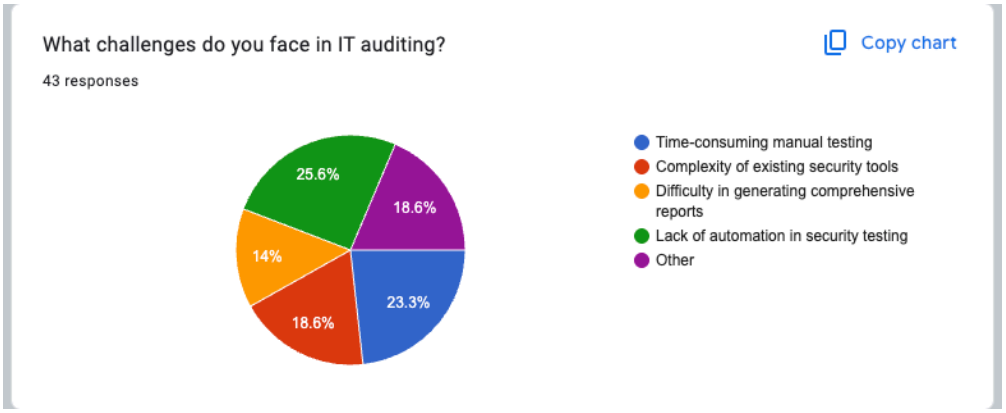


Figure 46 Gather Information of IT audit Challenges

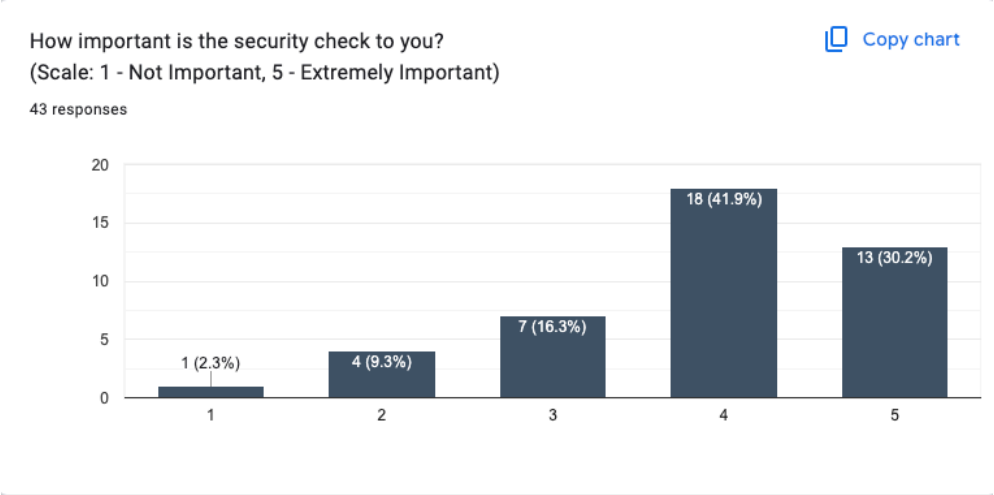


Figure 47 Important of security

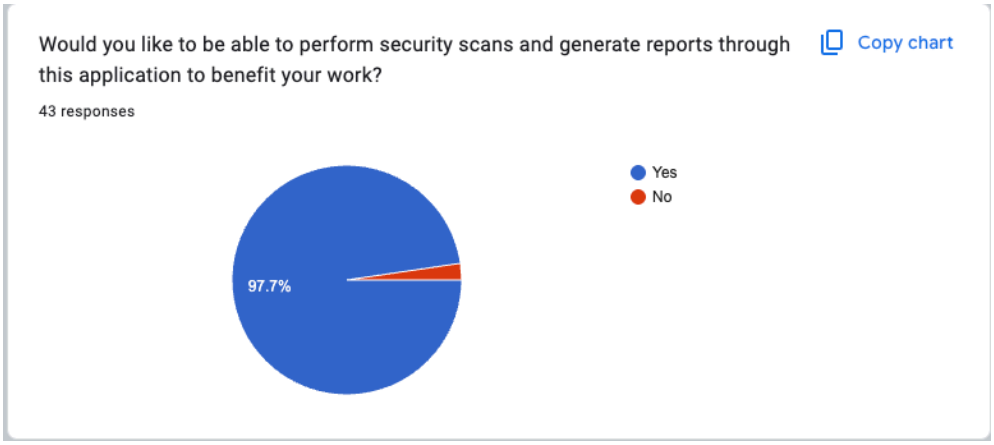


Figure 48 Gather Information of report generate

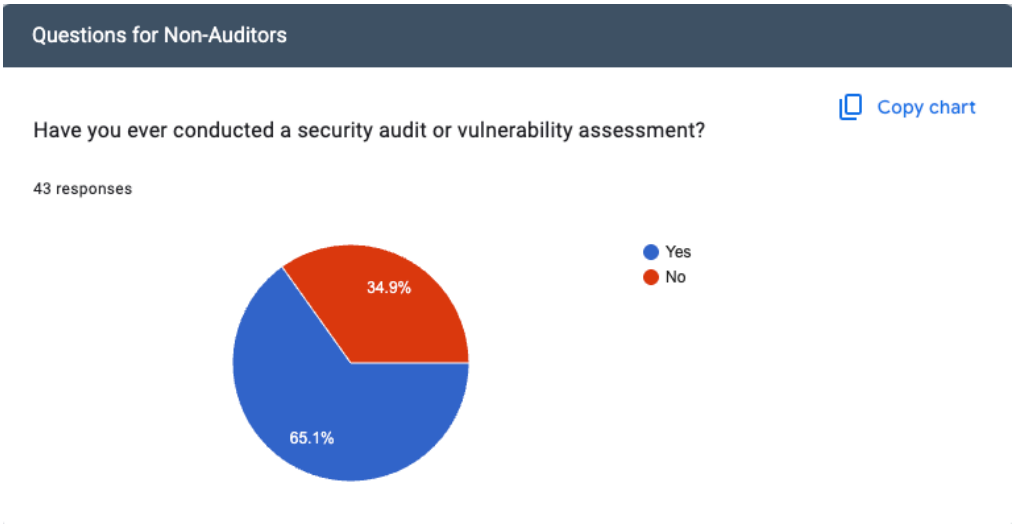


Figure 49 Gather Information about tool usage

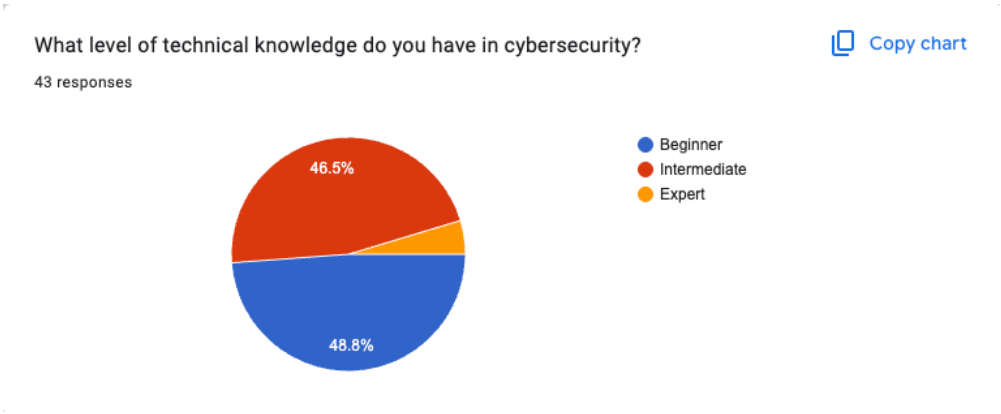


Figure 50 : Gather Information client's technical knowledge

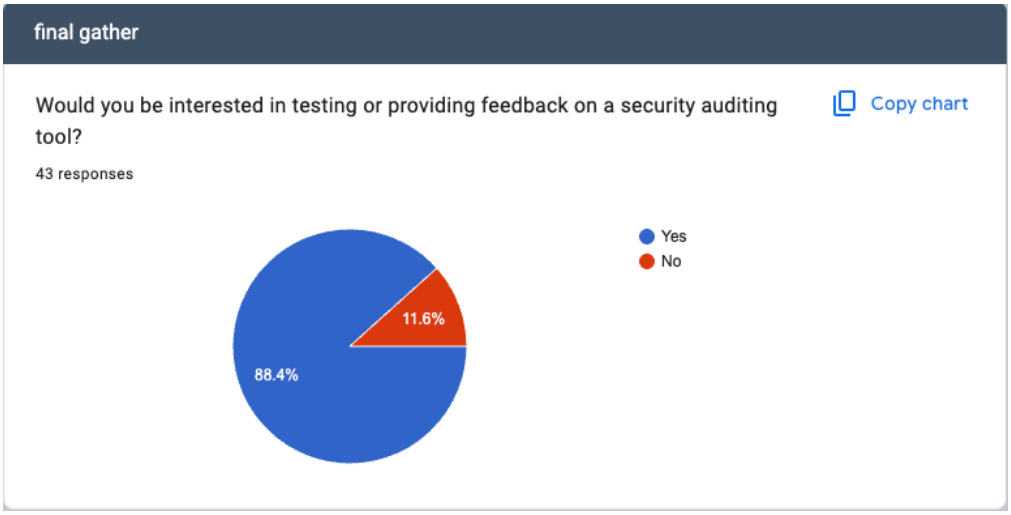


Figure 51 Security Audit Feedback of clients

Do you have any additional suggestions or comments regarding the IT audit tool?

17 responses

Enable scans without requiring continuous internet access.

Make it compatible with windows, Linux

Scheduled Scanning Feature

I have not

Be aware of Security and Privacy Considerations

Multi-Factor Authentication, Audit Logging, Data Retention & Compliance

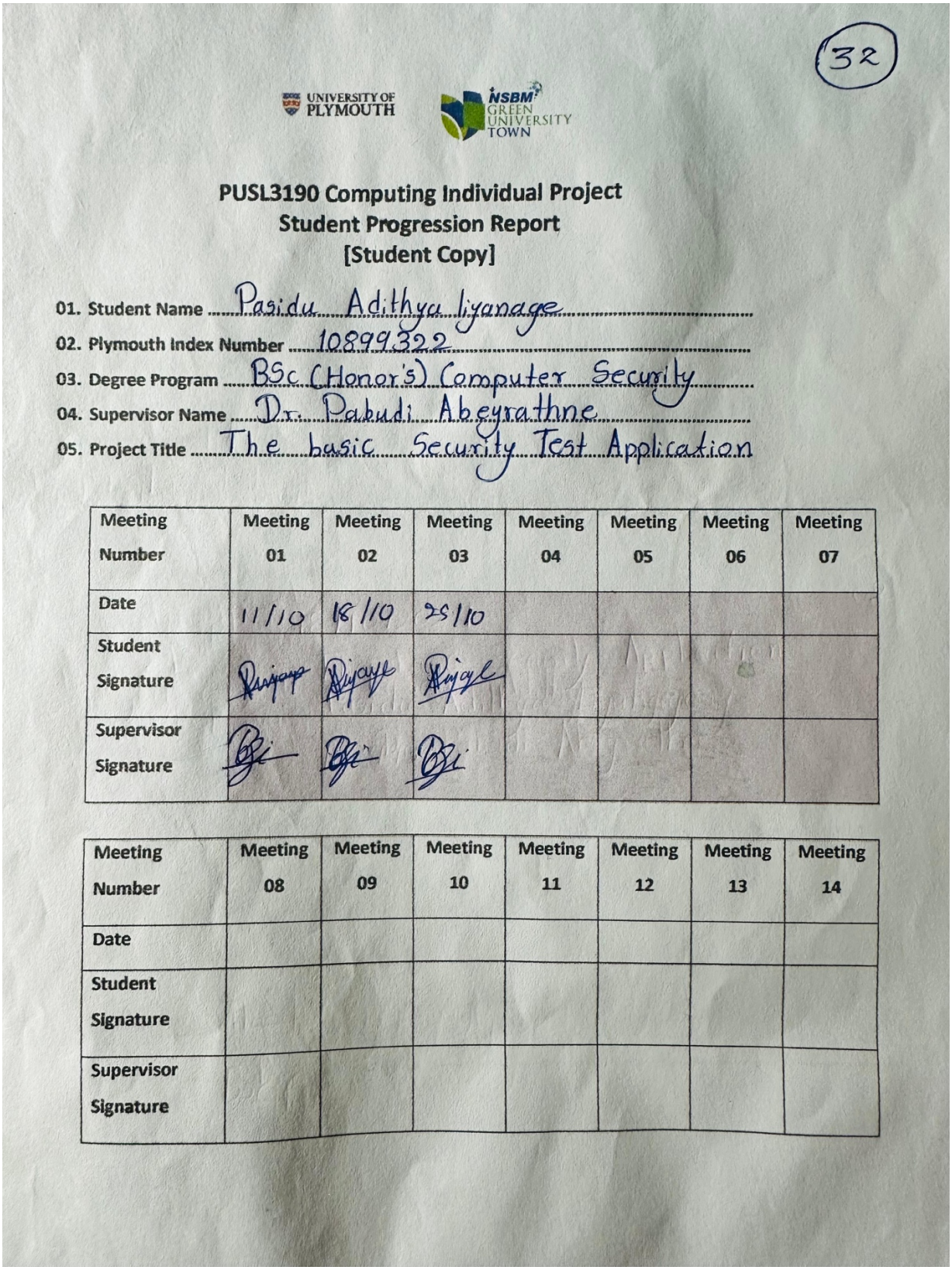
Enable scans without requiring continuous Internet access

create a user-friendly Interface & generate clear report out-put of this project

Ensure an intuitive design that requires minimal technical expertise.

Figure 52 Additional comments

Records of supervisory meetings



32

UNIVERSITY OF PLYMOUTH

NSBM GREEN UNIVERSITY TOWN

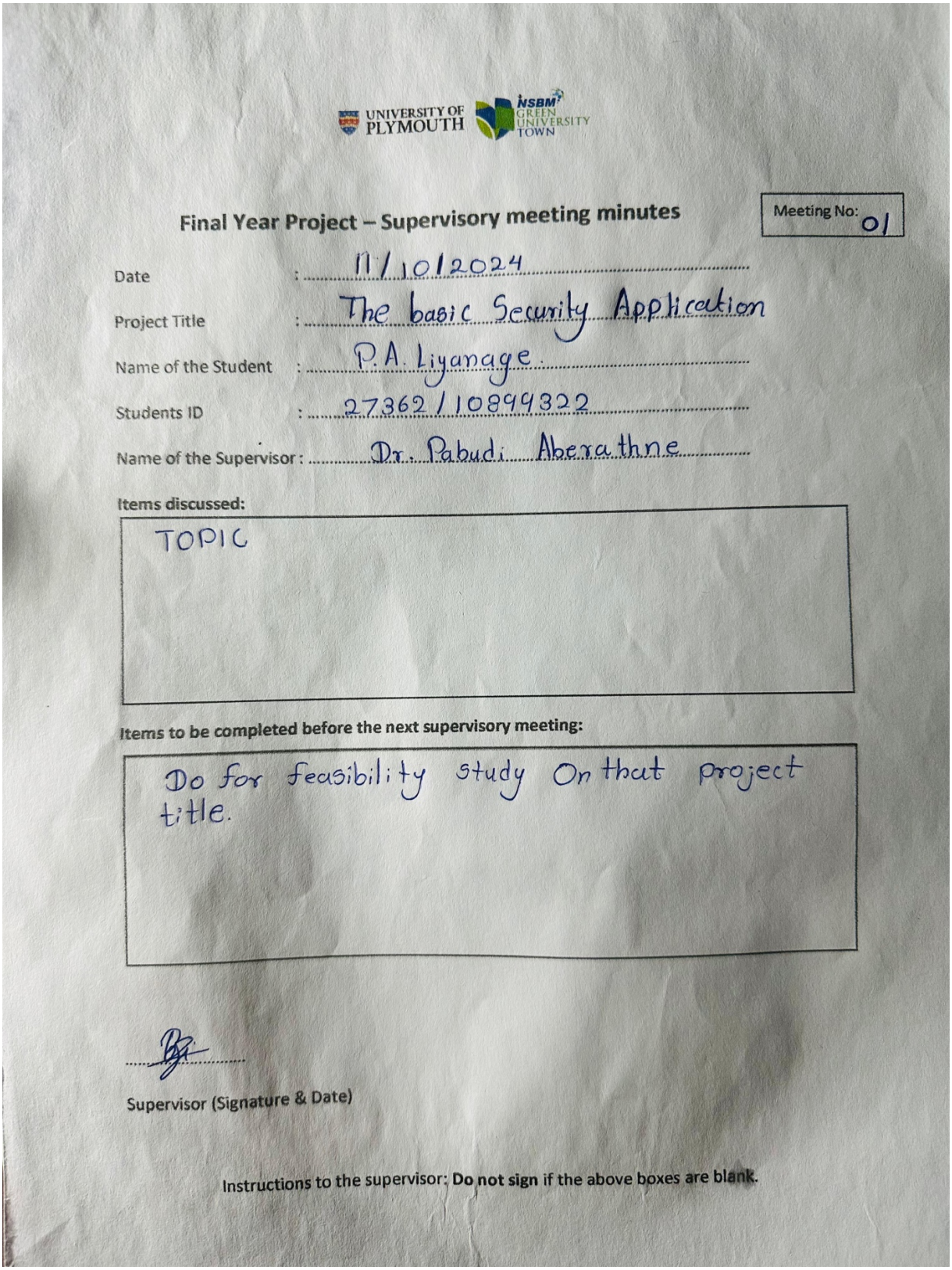
PUSL3190 Computing Individual Project
Student Progression Report
[Student Copy]

01. Student Name ..... Pasidu Adithya Liyanage .....
02. Plymouth Index Number ..... 10899322 .....
03. Degree Program ..... BSc (Honor's) Computer Security .....
04. Supervisor Name ..... Dr. Pabudi Abeyrathne .....
05. Project Title ..... The basic Security Test Application .....

Meeting Number	Meeting 01	Meeting 02	Meeting 03	Meeting 04	Meeting 05	Meeting 06	Meeting 07
Date	11/10	18/10	25/10				
Student Signature							
Supervisor Signature							

Meeting Number	Meeting 08	Meeting 09	Meeting 10	Meeting 11	Meeting 12	Meeting 13	Meeting 14
Date							
Student Signature							
Supervisor Signature							

Figure 53 Records of supervisory meetings



UNIVERSITY OF PLYMOUTH

NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 01

Date : 11/10/2024

Project Title : The basic Security Application

Name of the Student : P.A. Liyanage.

Students ID : 27362/10899322

Name of the Supervisor : Dr. Pabudi Aberathne

Items discussed:

TOPIC

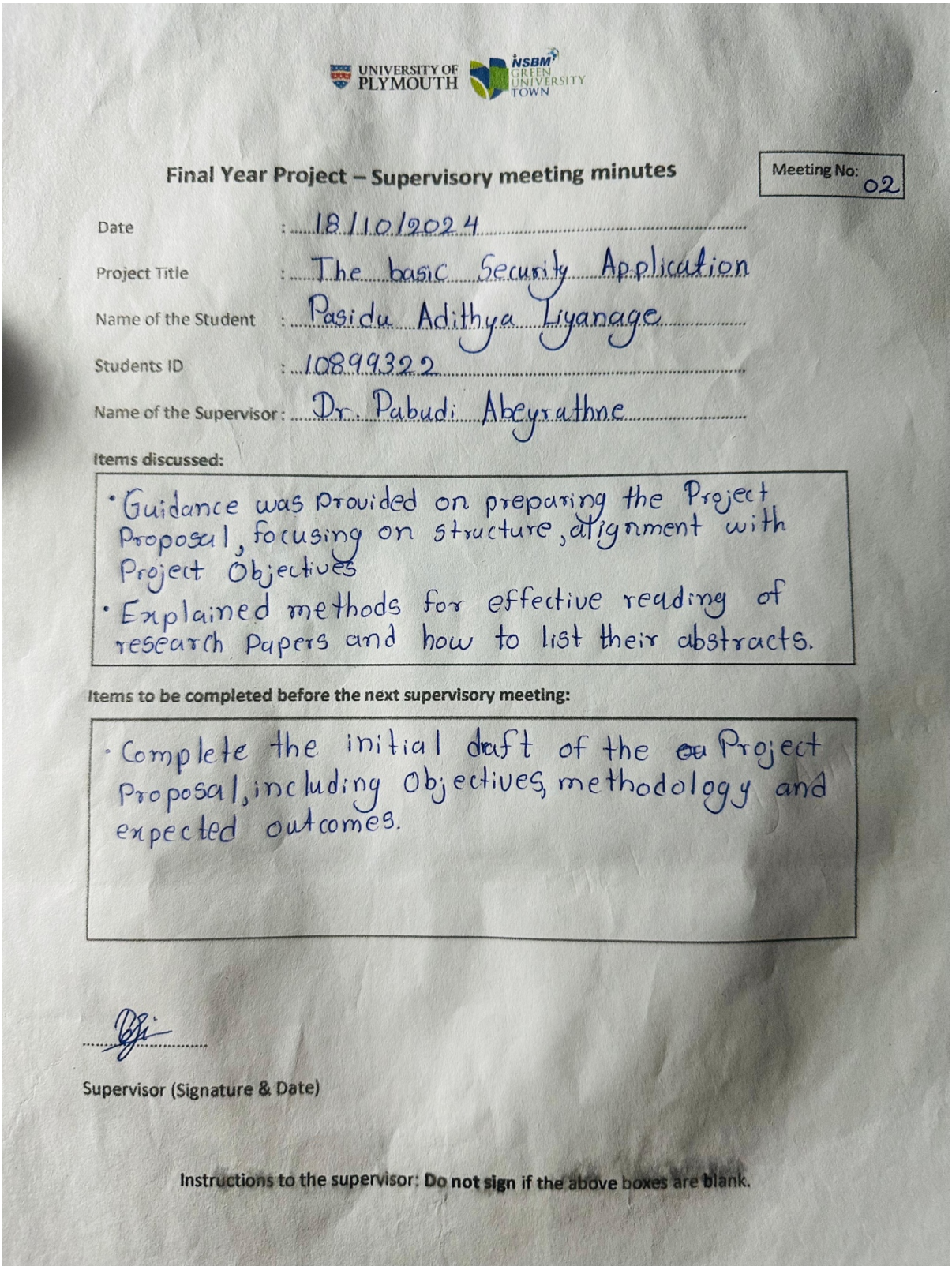
Items to be completed before the next supervisory meeting:

Do for feasibility study On that project title.

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 54 supervisory meeting minutes



UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 02

Date : 18/10/2024

Project Title : The basic Security Application

Name of the Student : Pasidu Adithya Liyanage

Students ID : 10899322

Name of the Supervisor : Dr. Pabudi Abeyrathne

Items discussed:

- Guidance was provided on preparing the Project Proposal, focusing on structure, alignment with Project Objectives
- Explained methods for effective reading of research papers and how to list their abstracts.

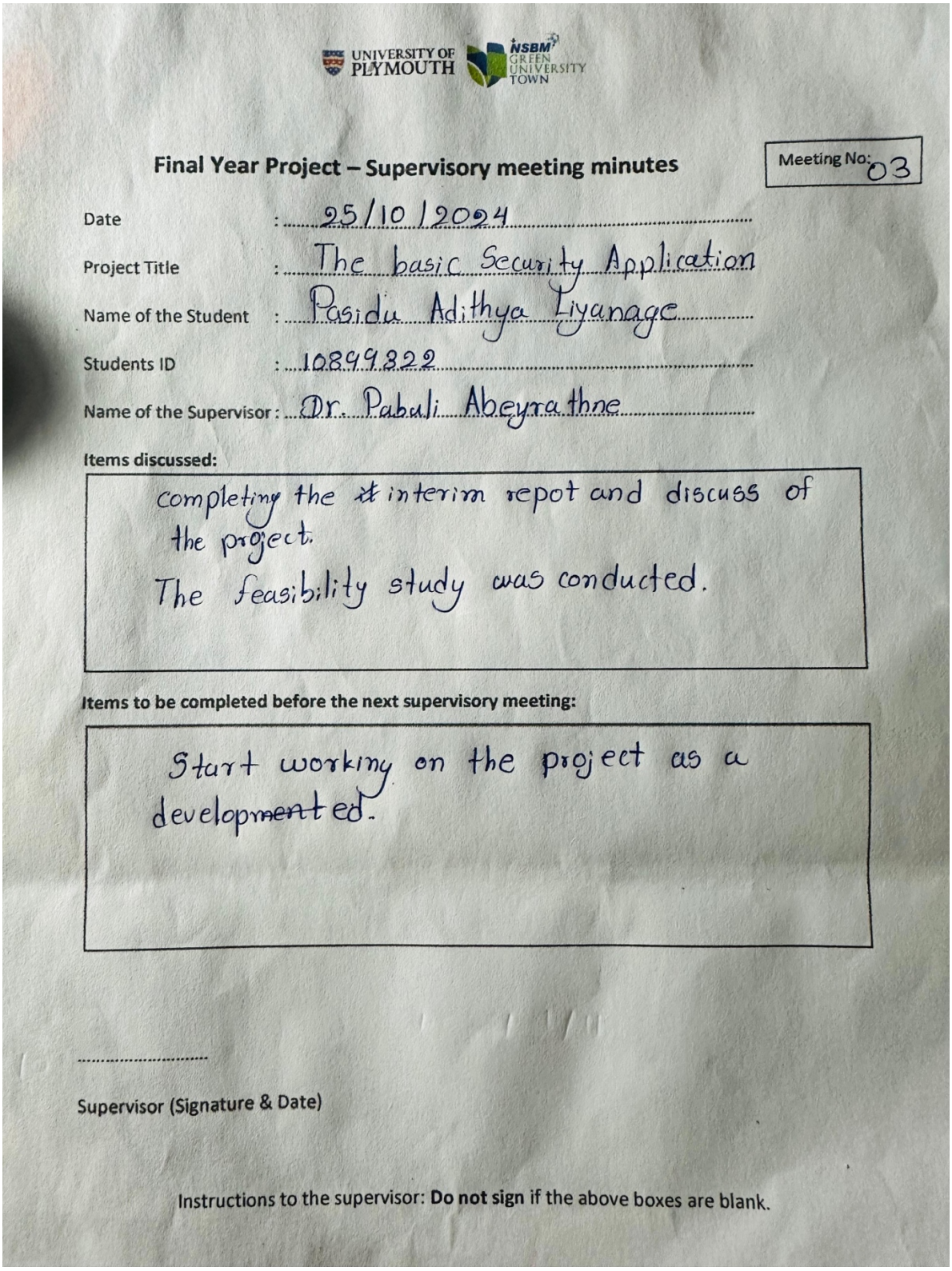
Items to be completed before the next supervisory meeting:

- Complete the initial draft of the ~~ou~~ Project Proposal, including Objectives, methodology and expected outcomes.

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 55 supervisory meeting minutes



UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

Final Year Project – Supervisory meeting minutes

Meeting No: 03

Date : 25/10/2024

Project Title : The basic Security Application

Name of the Student : Pasidu Adithya Liyanage

Students ID : 10899322

Name of the Supervisor : Dr. Pabuli Abeyrathne

Items discussed:

completing the it interim repot and discuss of the project.
The feasibility study was conducted.

Items to be completed before the next supervisory meeting:

Start working on the project as a developmented.

.....

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

Figure 56 supervisory meeting minutes