Unified Vulnerability Scanning Engine and Management System

Final Documentation

Andrew Gilbey C00263656
Supervisor: Richard Butler

Abstract

This report documents the finalised Unified Vulnerability and Management System project. The system is a fully functioning automated vulnerability scanner and management system, which automates the scans of different pre-installed vulnerability scanners through the use of a web application. These scanners can be configured and set to run through the system user interface on the web application; it is capable of running Nmap scans, DNS lookups, OPEN-VAS scans and ZAP web application scanning, saving the results to a database and then creating one uniformed report in PDF format.

The system is built up with four distinct dashboards, each one tailored to cater to the specific user role that interacts with the system; System Administrators who can manage user accounts, Penetration Testers who can configure and run scans, Analysts who triage the results to identify vulnerabilities, and Engineers who focus on the remediation of these vulnerabilities. This role-based dashboard approach is laid out to ensure that each specific user has the tools to perform their specific role effectively.

Table of Contents

Acknowledgements

I would like to thank my supervisor Richard Butler for his guidance and support throughout this project, keeping me level headed and giving me ideas in order to create the best project possible.

I am also grateful for the support of my friends, whose encouragement played a big part in sustaining my motivation and more importantly, my resilience during the duration of the project.

1. Introduction

1.1 Background

In today's digital landscape where vast amounts of data are transferred across and stored using the internet, and with the sophistication of cyberattacks becoming greater, the necessity of cybersecurity cannot be overstated because it stands as the first line of defence against these cyberattacks and so can protect businesses against data breaches, financial losses and damage to company reputation, (Lo-A-Njoe, C, 2023), and can mitigate against the growing harm cyberattacks pose to customers and society, (Bada & Nurse, 2019). Therefore, vulnerability assessments have gained a critical importance as they help organisations identify vulnerabilities in their systems before they can be exploited, (IARM, 2023).

This project’s main aim is to develop an integrated and automated vulnerability scanning system to better meet the cybersecurity needs of organisations and by focusing on both efficiency and accessibility, it seeks to simplify the vulnerability assessment process for organisations of all sizes.

2. Project Overview

2.1 Project Objectives

1. **Develop the Unified Vulnerability Scanning Engine -** This will conduct network and web application scans for identifying security vulnerabilities by utilising open-source tools such as Nmap, OpenVAS, and OWASP ZAP to autonomously scan, detect, and report vulnerabilities within web assets.

2. **Development of the Management System** – The development of four dashboards, each unique for a given role to provide specialised interfaces which are tailored to users' roles. These include- Penetration Testers, Engineers, Analysts, and System Administrators. These dashboards serve as central point for user actions and will display relevant information that is unique to each role, such as initiating new scans for penetration testers, managing users for System Administrators, managing assigned vulnerabilities for Analysts, and viewing assigned vulnerabilities for Engineers.

3. **Develop a Complete Database Solution –** This operates in the backend and is used for storing scan outcomes and facilitating the comparison of new scans with previous ones. This database is a core component of the system as it stores the raw data that is then passed to the web application. The database is vital for storing all the pieces of data that the system processes.

2.2 Technology Stack

2.2.1 Programming Language

The language breakdown is as follows;

Language	Breakdown
Python	51.8%
HTML	35.9%
CSS	9.9%

JavaScript	2.4%

- **Python** was used as the primary programming language for this project for several reasons; it has an easy to write and read syntax when compared to Java and C++, it is open source and provides the advantage of a strong community of contributors, which means it has a number of libraries which can be utilised to simplify solutions.

  Python was used to design the majority of the backend of the project, which includes handling the scanners implementation (Nmap, Open VAS, ZAP), encryption using the cryptography library and security, some of which is also handled by FLASK. Several other Python libraries were also used extensively.

- **HTML** has been used as the core markup language for this project for its simplicity, widespread adoption, and interoperability. It is straightforward enough that it allowed for a rapid development of the applications web pages and its versatility integrates well with other technologies, which helped to reduce the development time.

  Each Dashboard, was coded with HTML and used in conjunction with *Jinja 2* in order to utilise Python functions and display their results.

- **CSS** is utilised in order to style each of the pages, and is responsible for generating the design, layout, and responsiveness of the web application holistically. Two style sheets were used, one for the Login Page in isolation, and another that holds the styling for the core pages as a whole. Each style sheet holds customised properties that define how the HTML elements are styled, and are setup in a way to ensure a visually appealing aesthetics to improve user experience.

- **JavaScript** is used less frequently but plays an important role in some of the application's functionality. The DataTables library, was used to create dynamic tables to display results on several pages. JavaScript also allowed the modification of certain elements within tables, transforming non-editable fields into editable ones, and enabled features such as a password reset. For organisation purposes, one, dedicated JavaScript page, acting as a repository for all the JavaScript code for the application was used, which allowed for easy maintenance across the different pages of the application.

2.2.2 Backend

- **Flask** was used as the web framework due to its Python integration and its efficiency in handling the backend processes. Its simplicity in setting up made it the optimal choice for developing the application.

- **My-SQL** Handles the backend database operations. Given the large size of the database, the database is crucial for the project. The databases were managed using DBbeaver, a Database Management System for Linux, and two were created: one for handling the core project, such as managing user data and scan results, and the other for encryption details such as keys. MySQLs works well with Python and offers a smooth operation for handling data within the system.

2.2.3 Vulnerability Scanners

- **Nmap (Network Mapper**) Nmap has been used for its network discovery capabilities and is an essential module of the system. Its versatility in identifying open ports and services on a network serves as the starting point for the system's security assessment. This provides an initial scan that identifies network assets, and this information is then used in conjunction with other tools in the system.

- **OpenVAS** is utilised for its vulnerability scanning capability and plays the role of the providing deeper vulnerability assessments for the system. This is because OpenVAS specialises in vulnerability assessment and scans network assets for known

vulnerabilities, misconfigurations, and other potential security risks, and then provides a detailed report of its findings, (Das, 2021).

- **OWASP ZAP (Zed Attack Proxy)** has been integrated for its specialised focus on web application security. It is designed to find various types of security vulnerabilities within a given web application and acts as the system's web application scanner. ZAP also offers a web crawl feature which can automatically navigate through a web application and map out its traversal path.

3. What was Achieved

3.1 Web interface

3.1.1 Four Role Based Dashboards

A dashboard for each user role was created, each having their own unique aesthetics and capabilities, and are designed with that role in mind. Each dashboard was created using HTML, styled with CSS and functionality implemented via Python classes while used with Flask. A small break down of the dashboards are as follows:

Administrator Dashboard

Colour Theme - Dark Blue (#252744)

Capabilities –

- **User Management** - Administrators can create new users, modify existing user data, and reset passwords.
- **Audit Log** – Administrators can view the audit log which provides records of system activities, including invalid login attempts and user data changes.

Statistic Display –

- **User Activity -** Total number of users, with breakdowns of active users, (those who have logged in within the last hour), and inactive users, (those who have not logged in for over a year or have never logged in).
- **System Logs** - Total number of log entries, scans conducted this week, and recent logs.

Visual Analytics –

- **User Roles** - Chart visualising the total number of users by their roles.
- **Logs by Action** - Chart showing log activities by type.
- **Scans by User**- Displays the frequency of scans performed by each user.

Figure 1: The Administrator Dashboard

Penetration Tester Dashboard

Colour Theme - Green (#758467)

Capabilities –

- **New Scans** – Penetration Testers can start new scans, and setup the configuration settings like Task Name, Target IP for tools like Nmap, DNS, and OpenVAS scanning.
- **ZAP Scans** – Set up and manage configurations for ZAP scans, including creating exclusion lists.
- **View Current Owned Scans** – View all scans initiated by the user, including ongoing, completed, and those pending VAS scans' results.
- **View Pending VAS Scans** – View the status of a VAS scan and once completed, send the details to the database.
- **View Retests** – Any scan that has been flagged for a retest and sent to the penetration tester can be viewed here.

Statistic Display –

- **Retests Alerts that need Attention –** The total number of retests that have been assigned to the logged in penetration tester.
- **Total Scans on Record –** The total amount of scans from all penetration testers.
- **Your Scans –** Total individual scans initiated by the user, then categorised by type, are displayed; Recon Scans (DNS and NMAP), VAS scans and ZAP scans.

Visual Analytics –

- **Vulnerabilities by Severity** - Chart visualising vulnerabilities identified by severity.
- **ZAP Scan Risks** – Pie chart showing the risk distribution of ZAP scans.
- **Total Scan Type Counts** - Displays a breakdown of scans by type—Recon, VAS, and ZAP.

Figure 2: The Pen Tester Dashboard

Analyst Dashboard

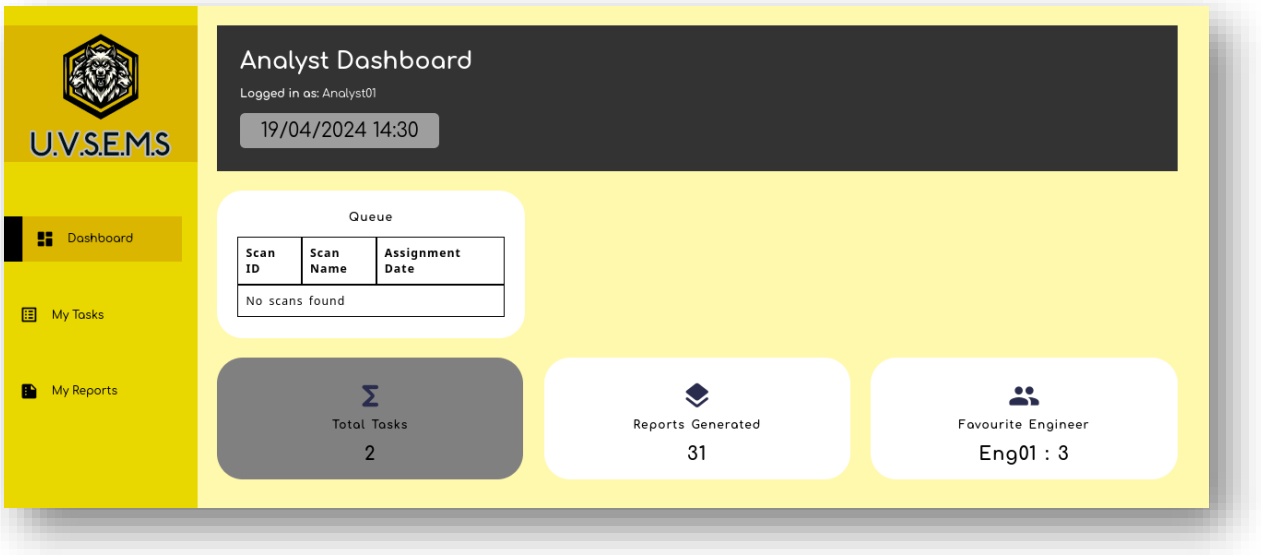
Colour Theme - Yellow (#E9D700)

Capabilities –

- **View Assigned Tasks** – Access to a dedicated page for scans that have been allocated to the user for triage. This feature is used to review scan results, generate result reports, and assign an engineer for remediation.
- **View Reports** – Enables users to access all reports generated by them, which provide a complete view of past report creation.
- **Comparative Analysis** – A detailed comparison between pairs of scans that are linked by retests, highlighting changes between them.

Statistic Display –

- **Total Tasks –** The total number of tasks of for the user
- **Reports Generated –** The total number of reports generated by the Analyst
- **Favourite Engineer –** The Engineer the Analyst works with the most is displayed in this box.



Engineer Dashboard

Colour Theme - Red (#A70000)

Capabilities –

- **View Assigned Jobs** – Access to a specific page dedicated to scans assigned to the user for remediation. This allows users to review scan results, generate detailed reports, and flag scans for retesting.
- **CVE Library** – Can navigate to a comprehensive CVE Library page that displays a searchable list of Common Vulnerabilities and Exposures (CVEs), which can be used by the user in order to enhance the user's ability to address security vulnerabilities efficiently.

- **Completed Jobs –** Review all jobs with status set to “Completed” for the user, facilitating an easy tracking of past activities and their outcomes.

Statistic Display –

- **Total Jobs in Record –** The total number of scans sitting with engineers in total.
- **Your Jobs** – The total number of scans assigned to an Engineer.
- **Favourite Analyst** – The Analyst the Engineer works with the most is displayed in this box.

3.2 Backend Database

The databases are hosted on individual Docker containers, each operating from a different port to ensure security.

- **Core Database**- One database is dedicated to hosting all the core functionalities of the system, providing functionality for operations.
- **Encryption Key Management** - The second database is a specialised database and is used to manage the Encryption Keys. These keys are critical for decrypting data as it moves through the system.

Both databases are using MySQL, which is used an efficient database management system. Management operations are primarily conducted through DBeaver, which is a database management tool that has allowed for easy interaction for maintenance of the database.

3.2.1 Core Database Tables: UVSEMS

Below is the Entity-Relationship (ER) diagram which represents the core database structure of the system. This diagram is used to provide a schematic of the data relationships between all the tables of the database, all of which are important for the system's operation.

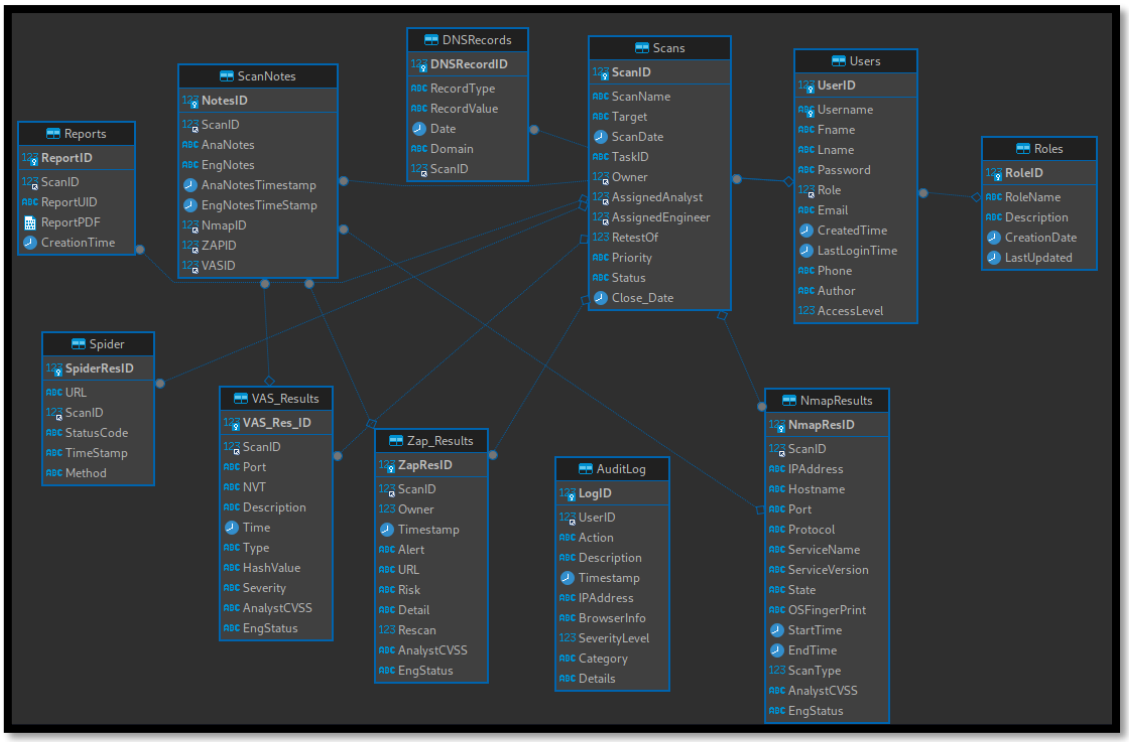


Figure 3: ER Diagram of the Database

A list of all the tables is provided below; these tables are the core of the system's database and are vital for storing all the pieces of data that the system processes. Each table is created with a specific function to ensure the system can run efficiently.

Essential user information, including access control, contact details and the user's role are saved in this table which is significant to enable user management and access control.

Table 1: User Table

Users	
UserID	This is the primary key of the table.
Username	A distinctive username for the user; this must be unique.
Fname	The user's forename.
Lname	The user's surname.
Password	The user's password; this is hashed using bycrypt.
Email	The user's email address.
Role	A foreign key linking to the Roles table; represents the user's role within a range of 1-4, depending on their role.
CreatedTime	The timestamp of when this user was created.
LastLoginTime	The timestamp of the user's last login.
Phone	The user's contact phone number.
Author	The UserID of the administrator who created this user account.
AccessLevel	Applicable to Administrators and determines the extent of access control a user has; Level 1 indicates administrative access with the ability to manage other users, including those with the same access level. By default, users have access Level 2.

Stores information about each of the user roles which can be linked back to the user's table.

Table 2: Roles table

Roles	
RoleID	This is the primary key of the table.
RoleName	The official name of the role.
Description	Description of the role.
LastUpdated	The timestamp of the last update made to the role's information.

Acts as a central repository for all scan activities, and catalogues all the scan detail of every scan along with the allocation to analysts and engineers.

Table 3: Scans table

Scans	
ScanID	This is the primary key of the table.
ScanName	The designated name of the scan; Must be unique.
ScanDate	The date on which the scan was started.
TaskID	The VAS Assigned Task ID if applicable
Owner	The UserID of the individual who initiated the scan.
AssignedAnalyst	The UserID of the analyst assigned to the scan.
AssignedEngineer	The UserID of the engineer assigned for remediation tasks.
RetestOf	The ScanID of the original scan if the current entry is a retest.
Priority	Importance level of the scan.
Close_Date	The date when the scan was closed.
Status	Current state of the scan (e.g., pending, in progress, completed).

All the reports that have been generated are saved to this table, including the PDF that was generated, time it was created and the given ScanID.

Table 4: Reports table

Reports	
ReportID	This is the primary key of the table.
ScanID	The foreign key relating back to a ScanID.
ReportUID	The unique identified assigned to a report based on the time and date it was created.
ReportPDF	Blob; stores the actual PDF.
CreationTime	The time the report was created.

Stores annotations/notes which have been made from an analyst and engineer which effectively acts as collaborative communication tool for the lifecycle of each scan.

Table 5: ScanNotes table

ScanNotes	
NotesD	This is the primary key of the table.
ScanID	The foreign key relating back to a ScanID.
EngNotes	Notes created by an Engineer.
EngNotesTimestamp	Timestamp of the last Engineer Note.
AnaNotesTimestamp	Notes created by an Analyst
AnaNotes	Timestamp of the last Analyst Note

The NmapResults table holds data on from the network scans that have been performed by Nmap, these track each scans specifics, such as targeted IP addresses, operating system identified and forms a big component of a scan's overall makeup.

Table 6: Nmap results table

NmapResults	
NmapResID	This is the primary key of the table.
ScanID	The foreign key relating back to a ScanID.
IPAddress	The IP address that was scanned.
Hostname	The resolved name of the scanned IP address.
Port	The port number that was scanned
Protocol	The communication protocol used by the scanned port.
ServiceName	The name of the service running on the port.
ServiceVersion	The version of the service identified.
State	The current state of the port, (e.g., open, closed).
OSFingerPrint	The operating system identification gleaned from the scan.
StartTime	The timestamp when the Nmap scan was started.
EndTime	The timestamp when the Nmap scan was completed.
ScanType	The type of Nmap scan conducted.
AnalystCVSS	The severity rating assigned by the Analyst.
EngStatus	The status provided by the Engineer.

Saves the outcomes of any web crawling, which includes details of URLs processed, status codes encountered, and timestamps.

Table 7: Spider table

Spider	
SpiderResID	This is the primary key of the table.
URL	The web address that was the subject of the process.
StatusCode	The HTTP status code received when accessing the URL.
TimeStamp	The timestamp when the spider captured the URL information.
Method	The HTTP method used for the spidering process.

Stores the results of vulnerability assessments performed by the Vulnerability Assessment Scanner (VAS), and details severity of vulnerabilities that have been discovered across a scanned network.

Table 8: VAS results table

VAS_Results	
VAS_Res_ID	This is the primary key of the table.
ScanID	The foreign key relating back to a ScanID.
Port	The port number that was assessed during the scan.
NVT	The Network Vulnerability Test identifier used.
Description	A brief description of the findings.
Time	The timestamp when the VAS result was logged.
Type	The type or category of the vulnerability found.
HashValue	A unique hash representing the scan result.
Severity	The severity level of the vulnerability.
AnalystCVSS	The severity rating assigned by the Analyst.

EngStatus	The status provided by the Engineer.

Results from DNS lookups are saved in the DNS_Results table, which details all the record types and associated values as well the domain name.

Table 9: DNS_Results table

DNS_Results	
DNSRecordID	This is the primary key of the table.
RecordType	The type of DNS record, such as A, TXT, etc
RecordValue	The value associated with the DNS record.
Date	The date when the DNS record was retrieved.
Domain	The domain name corresponding to the DNS record.
ScanID	The foreign key relating back to a ScanID.

The systems audit log details are meticulously recorded in this table, stored details from user actions, invalid logins and other errors which may arise across the system.

Table 10: AuditLog table

AuditLog	
LogID	This is the primary key of the table.
UserID	The ID of the user associated with the log entry.
Action	The type of action that triggered the log entry.
Description	A detailed account of the action taken.
Timestamp	The exact time when the log entry was created.
IPAddress	The IP address of the user at the time of the log event.
BrowserInfo	Information about the user's browser during the log event.
SeverityLevel	The level of severity assigned to the log event.
Category	The classification of the log entry (e.g., error, info, warning).
Details	Additional information or context about the action logged.

3.2.2 Key Storage Database Tables: KeyBank

The KeyBank system has no relationships between the tables, the relationships that actually exist are between the KeyBank database and the Core Database tables which are not displayed on an ER diagram.

Because of the database is specialised, each table this database follows an almost uniform template designed to store the encryption elements; this template is applied consistently across the database and contains the following fields:

Table 11: Keybank table template

<Table>Key	
KeyID	This is the primary key of the table.
Key	The encryption key used to encrypt/decrypt the data.
Nonce	The “number used once” for each record. Used in conjunction with the key to encrypt/decrypt the data.
<Table>ID	The given record for a table to ensure it matches.

3.3 Encryption Process

To ensure data security, AES-256 encryption in Galois/Counter Mode (GCM) has been used across the system for the database encryption process. AES-256 offers a high level of cryptographic strength that is difficult for threat actors to breach and so safeguards against unauthorised data access (team, 2023), and GCM further enhances this by combining Counter Mode (CTR) with authentication. This makes it faster, more secure, and better optimised for table-driven field operations as well as supporting both authenticated encryption and authenticated decryption, providing a robust security framework, (Kariyawasam, 2021). Pythons’ cryptographic library, “cryptography” is being used to handle this process as it offers built-in support for AES-256-GCM, (Cryptography Development Team, 2023).

The *KeyBank* database is maintained to securely store the encryption keys, ensuring that even if unauthorised access to the central database occurs the keys will remain protected. This ensures the confidentiality of data but also its integrity making it an ideal solution for securing the database, (OWASP Cheat Sheet Series Team, n.d.).

The below diagram presents a data flow diagram that outlines the encryption and decryption processes within the system.

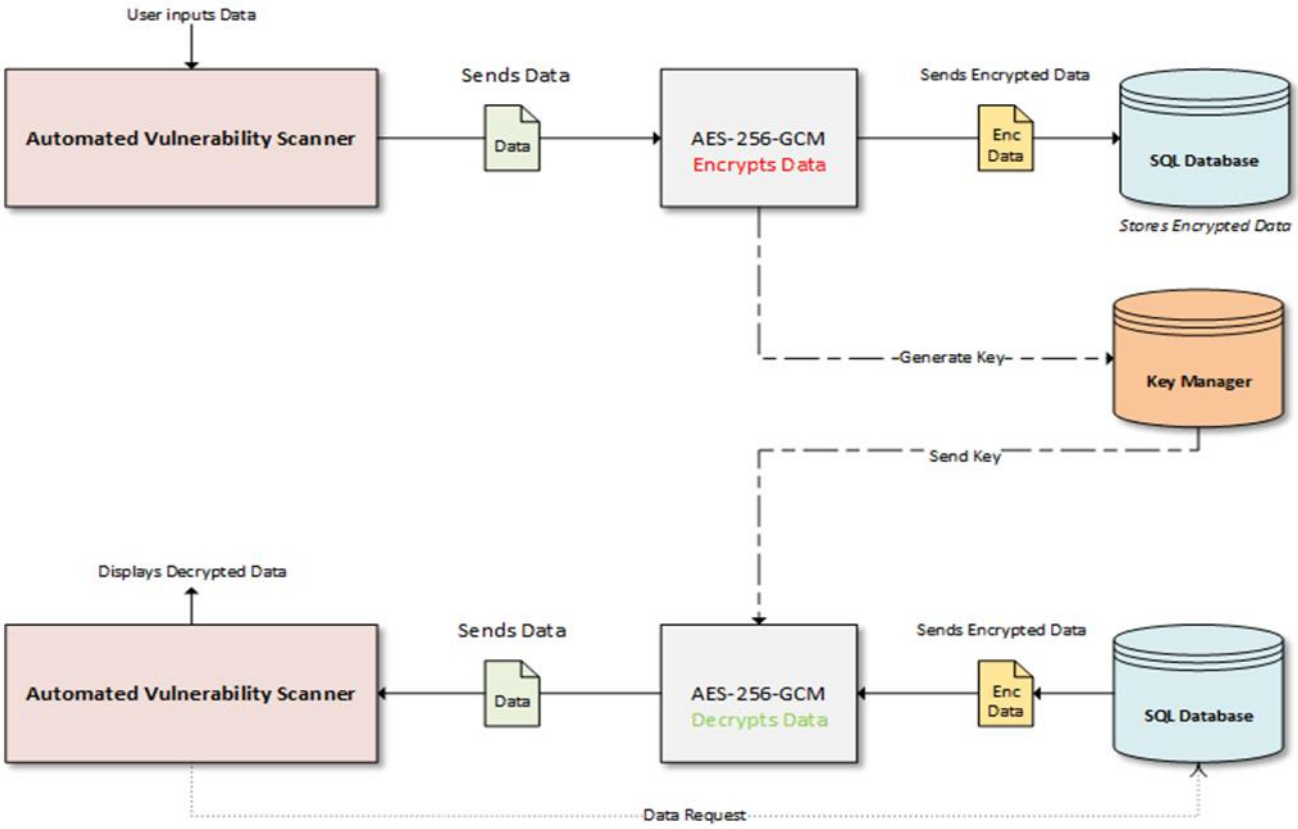


Figure 4: The Encryption Process

3.3.1 Encryption Functions

3.3.1.1 Generating the Key and Nonce

The first function that is part of the encryption process is the *generate_keynonce* function. This function is responsible for generating a pair of cryptographically secure keys; the key itself and the nonce. This is done using the *os.urandom* function, which generates the values and ensures that the key is 32 bytes (256 bits) and the nonce is 12 bytes (96 bits) long, which follows the recommended best practice for AES encryption, (Hinch, 2023).

3.3.1.2 Encrypting the Data

The *encrypt_data* function is responsible for the actual encryption of the data, transforming it from plaintext to cypher text. It uses AES-GCM, which is a symmetric encryption algorithm providing both confidentiality and integrity.

The function itself takes three parameters: the data, which is the data to encrypt, the key, which is generated from the *key_nonce* function and is a 32-byte formatted encryption key, and the nonce, which is a 12-byte formatted value designed to ensure entropy of the encryption process. The data is first encoded into bytes, then encrypted using AESGCM. The resulting encrypted bytes and then encoded into base64 which allows safe transmission into the database.

3.3.1.3 Decrypting the Data

The *decrypt_data* function is designed to securely decrypt data that has been previously encrypted with the *encrypt_data* function. It accepts three parameters: encryptedData, which is the base64-encoded data to be decrypted; key, which is the AES decryption key in raw byte format; and nonce, a unique value used during the encryption process, also in byte format – the key and nonce both being pulled from the database respectively.

The function first decodes the encrypted data from base64 format, then uses the AESGCM class with the given key to decrypt the data using the provided nonce. It then decodes the decrypted bytes back into a UTF-8 string to return it into text.

3.4 Scanner Integration

The three scanners and DNS lookup have been successfully integrated into the system; these scanners are handled by the *ScanUtils* class and all corresponding Python code resides here.

3.4.1 Nmap

The way the system handles Nmap scans is split into two functions.

The *set_nmap_options* function provides a way to retrieve specific Nmap commands based on a given integer ID. It includes seven pre-defined scans for Nmap, which allow for different scans and is setup in a way that they are completely modifiable – so if the commands need to be changed, a developer can enter the code and change the string (or add more), so they can be tailored to a user or businesses requirement easily.

The core function for Nmap, *doNmapScan*, uses these configurations from *set_nmap_options* to execute an Nmap scan on a specific target IP address and is controlled via the Nmap library. The scans are then executed, captured into dictionary and then passed to the database.

3.4.2 DNS-Lookup

The DNSPython library is used to handle the DNS-Lookup scans and is ran through the *do_DNS_scan* function. This function queries different DNS records types, NS, A, TXT and MX, for a given domain. It then formats the results into a dictionary using the record types as keys and then passes them to the database.

The strategy behind this was to mirror the approach of the online tool "*DnsDumpster*", (DNS Dumpster, n,d), from which the inspiration for this particular functionality originates.

3.4.3 Open-VAS

Two functions handle the VAS scanning process and use the GVM libraries to ensure operation. These functions utilise the GVM API, allowing for automated interactions with the GVM platform and provide a programmable approach to managing the VAS scanning process.

The *create_VAS_task* function is called first and it sets up the scanning of a "task". It first initialises the task and then assigns the parameters that are configured on the Penetration Tester's scan configuration page which will be linked to a targeted IP for scanning. Once this has completed, the second function, *start_VAS* will execute, which is responsible for starting the scan under the given configurations.

Both functions require the GVM API to run correctly, using an API key to allow for an authenticated access to the GVM platform, enabling the ability to execute scans. Once, the scan completes, a Penetration Tester can then send the results to the database.

3.4.4 ZAP

ZAP handled by one function, using the ZAP API through the ZAPv2 class. The function will work differently depending on the configuration set by a Penetration Tester user on the ZAP scan page.

ZAP differs to the other scans, as the ZAP scan must be integrated independently from the other two and concatenated into another scan. This means a scan must be made, (this can be blank), before a ZAP scan can be run and integrated with the scan.

The *run_zap_Scan* function is then executed based on the parameters set by the Penetration Tester and then goes through a specific process to start running.

1. A new session in ZAP is initiated first using the *new_session* method from the ZAP API; this will ensure that each scan is managed independently.
2. Exclusion list is setup from the parameters a Penetration Tester sets, and means anything included here will be ignored by both the spider and active scan.
3. If the Use Crawler flag is set, then ZAP will run the spider, which will map out the given application and finds any URLs.
4. If an active scan was initiated then an active scan will be started and will scan for any vulnerabilities on the target page.
5. Once the scan is finished, the alerts are compiled, processed and passed to the database.

3.5 Full Scan Lifecycle

3.5.1 Scan Initialisation

The Penetration Tester (Pen Tester) begins by navigating to the “New Scan” tab located in the sidebar of the dashboard, which opens the scan configuration page. Here, the Pen Tester fills out the details of the form according to requirements and initiates the scan. Once it has been initiated, the status of scan will update first to “Initialising” and then to “Running”.

Nmap and DNS scans are synchronous; and require the system to wait until these scans complete before the results will be sent to the database. In contrast, VAS scans are asynchronous as they require more time to complete. If a VAS scan is configured, the status remains at “Running” until the scan has been completed, after which the results can be manually passed to the database.

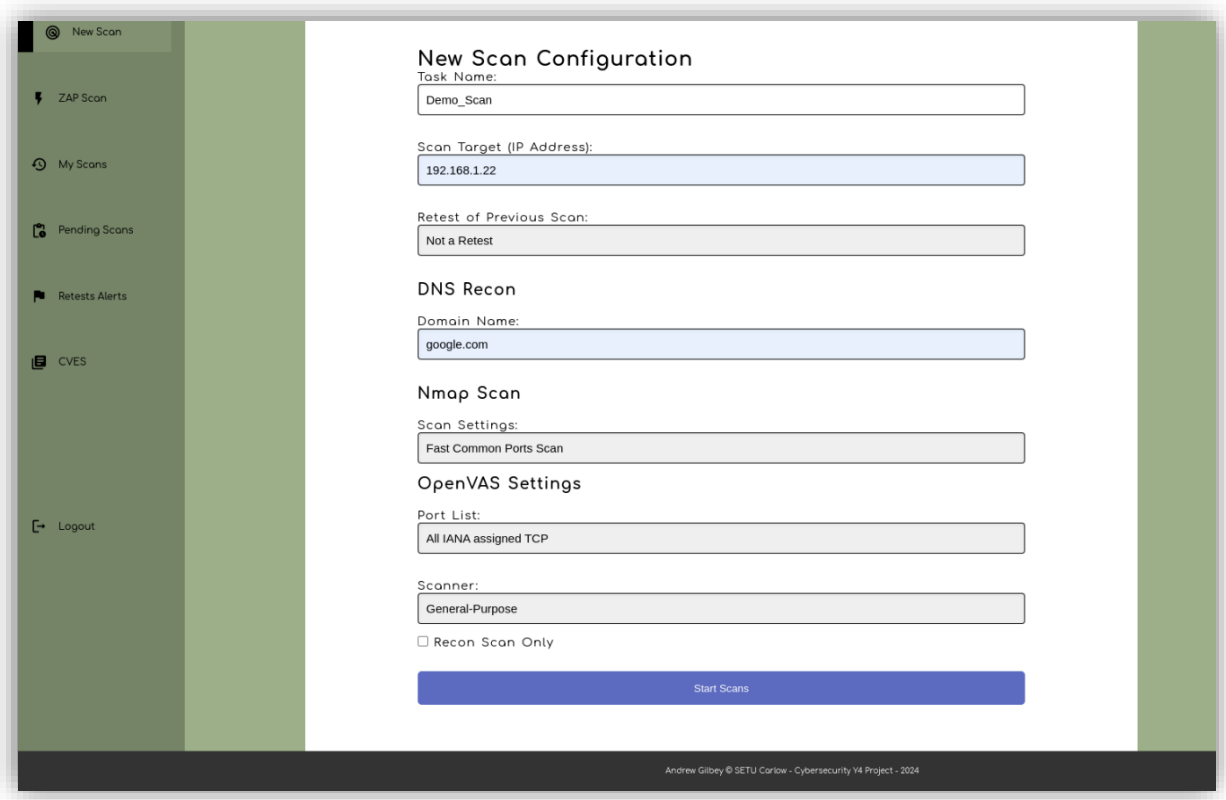


Figure 5: Scan configuration page

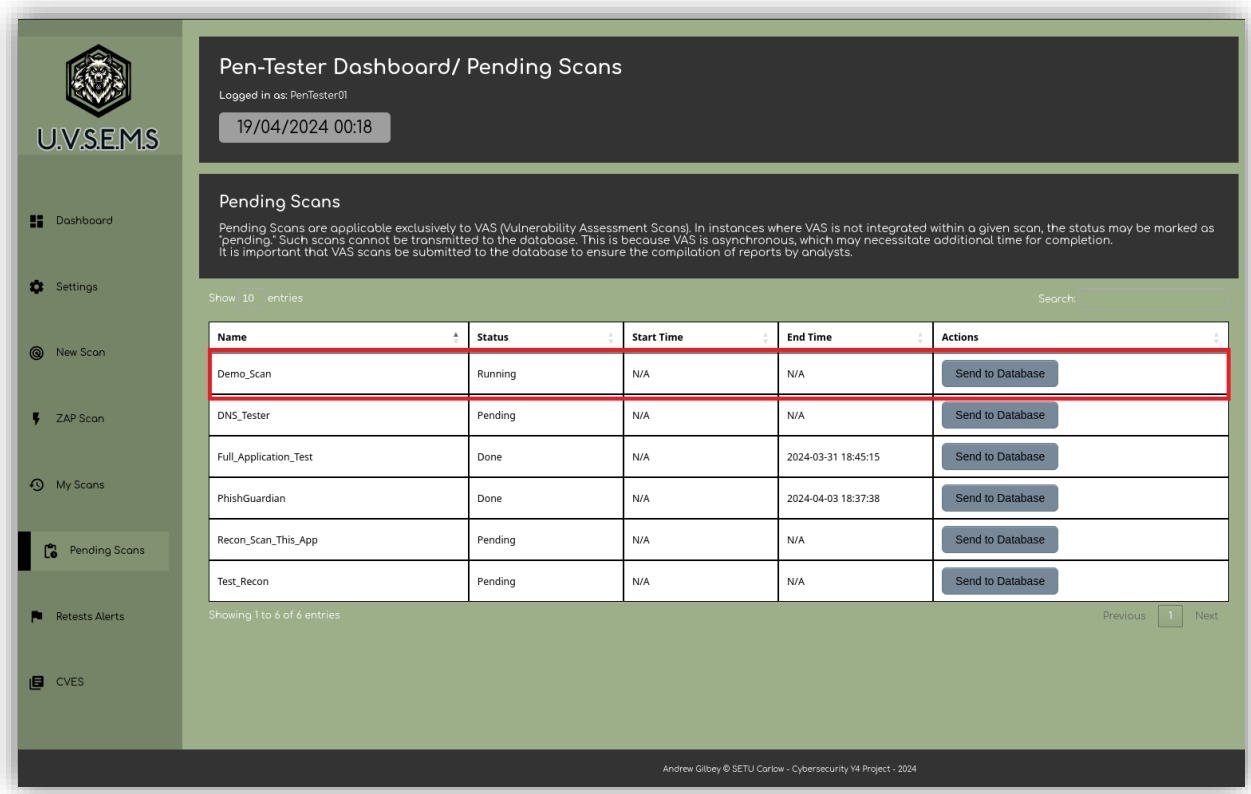


Figure 6: Pending scans page illustrating the status running

Once the scan has completed and the results passed to the database, the Pen Tester is then able to access the results. The status, will be set to “Initiated” at this point, which indicates the scan has finished. After review, the results can be forwarded to an Analyst for further triage. This transition ensures that each review follows the “Four Eyes Principle” to enhance the accuracy of the analysis, (IBM, 2024).

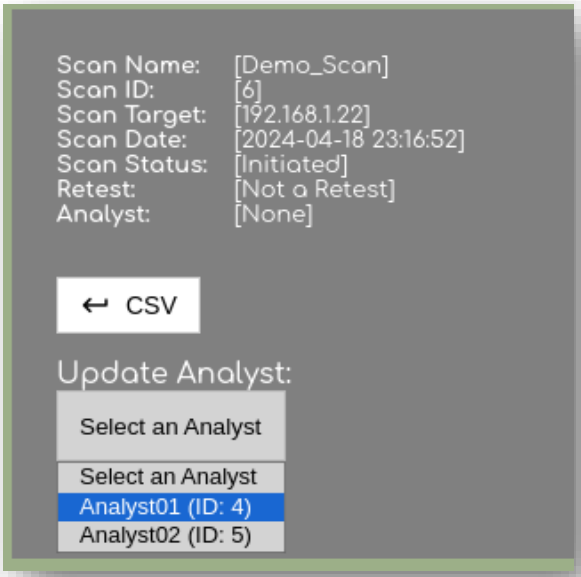
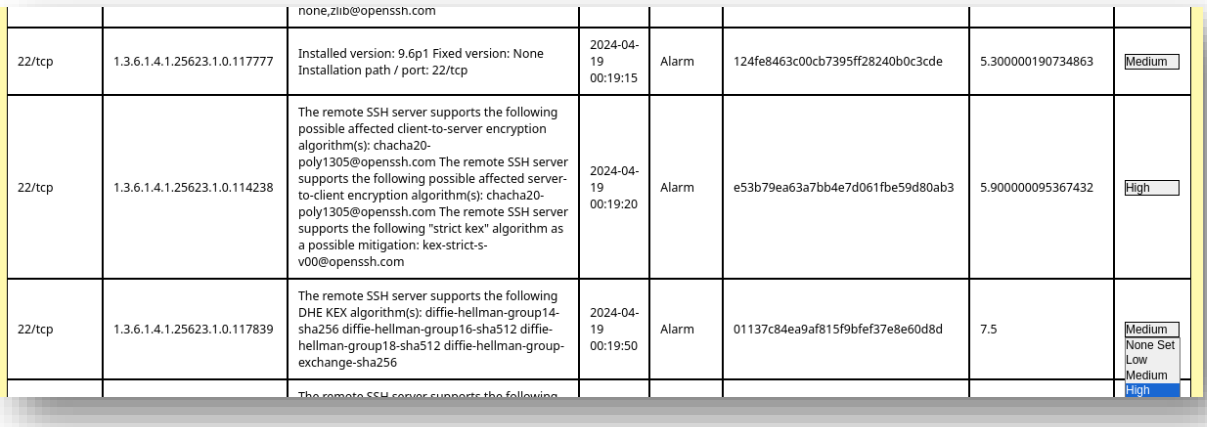


Figure 7: Assignment of Analyst

3.5.2 Scan Triage

Once the scan is passed to the Analyst, the status is updated to “Triage”. The Analyst now assumes responsibility for evaluating the results and assigning their own CSVV scores based on their assessment. These scores are categorised as Low, Medium, High or False Positive, which enables prioritisation and action planning. The overall priority of the scan can also be set.

Once the triage process has been completed, the Analyst assigns an Engineer to address the identified vulnerabilities.



		none,zlib@openssh.com					
22/tcp	1.3.6.1.4.1.25623.1.0.117777	Installed version: 9.6p1 Fixed version: None Installation path / port: 22/tcp	2024-04-19 00:19:15	Alarm	124fe8463c00cb7395ff28240b0c3cde	5.300000190734863	Medium
22/tcp	1.3.6.1.4.1.25623.1.0.114238	The remote SSH server supports the following possible affected client-to-server encryption algorithm(s): chacha20-poly1305@openssh.com The remote SSH server supports the following possible affected server-to-client encryption algorithm(s): chacha20-poly1305@openssh.com The remote SSH server supports the following "strict kex" algorithm as a possible mitigation: kex-strict-s-v00@openssh.com	2024-04-19 00:19:20	Alarm	e53b79ea63a7bb4e7d061fbe59d80ab3	5.900000095367432	High
22/tcp	1.3.6.1.4.1.25623.1.0.117839	The remote SSH server supports the following DHE KEX algorithm(s): diffie-hellman-group14-sha256 diffie-hellman-group16-sha512 diffie-hellman-group18-sha512 diffie-hellman-group-exchange-sha256	2024-04-19 00:19:50	Alarm	01137c84ea9af815f9bfef37e8e60d8d	7.5	Medium None Set Low Medium High
		The remote SSH server supports the following					

Figure 8: The Analyst can change their own CSVV scores

3.5.3 Remediation Process

Once the Engineer receives the scan details, the scan status is updated to “Remediation Pending”. This status change reflects that the Engineer is actively working to address the identified vulnerabilities. The Engineer systematically reviews each vulnerability and implements the necessary fixes based on their severity. After addressing the vulnerabilities, the Engineer flags it for a re-test to ensure that the fix has effectively been resolved and that no other vulnerabilities remain.

Once it has been flagged for re-test is sent back to Pen Tester and this is reflected in their Re-test queue.

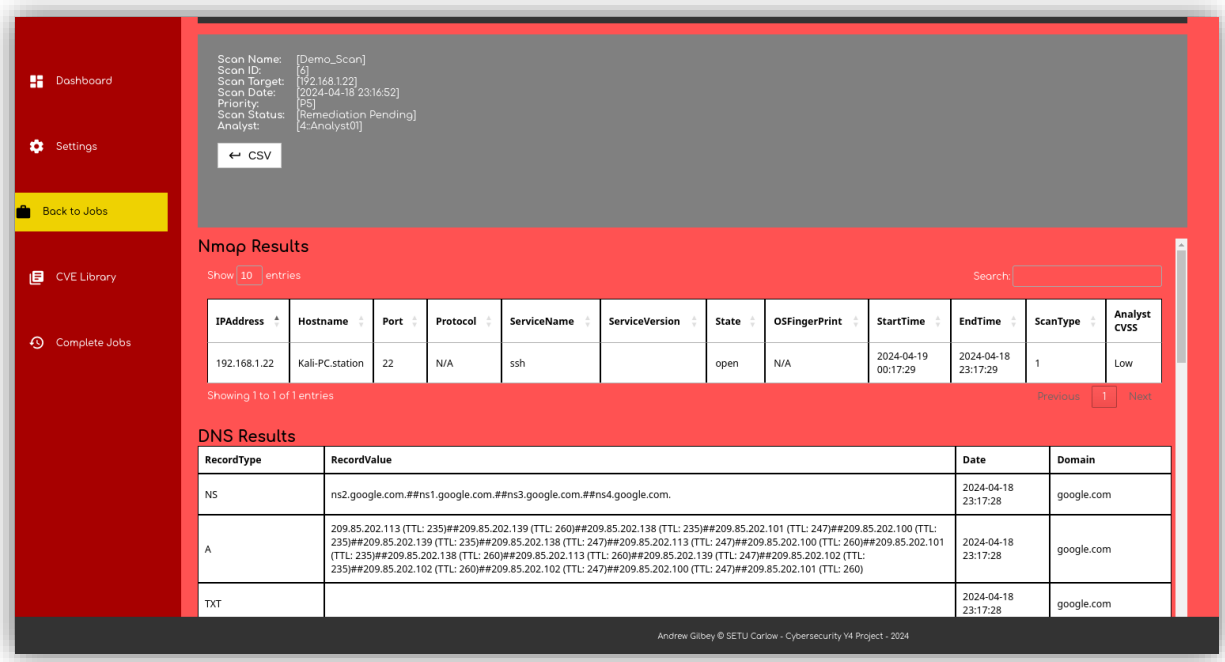


Figure 9: The Engineers view of the Scan Results

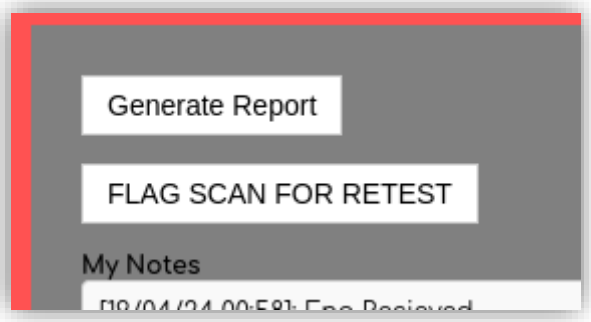


Figure 10: The flag scan for retest button

3.5.4 Retest Process

Once the Engineer has completed their remediations and flagged the scan for re-test, the scan status is updated to “Retest Required”. This scan will then be pushed to the Pen Tester’s retest alerts, and notify them that a follow-up scan is necessary to confirm the remediations.

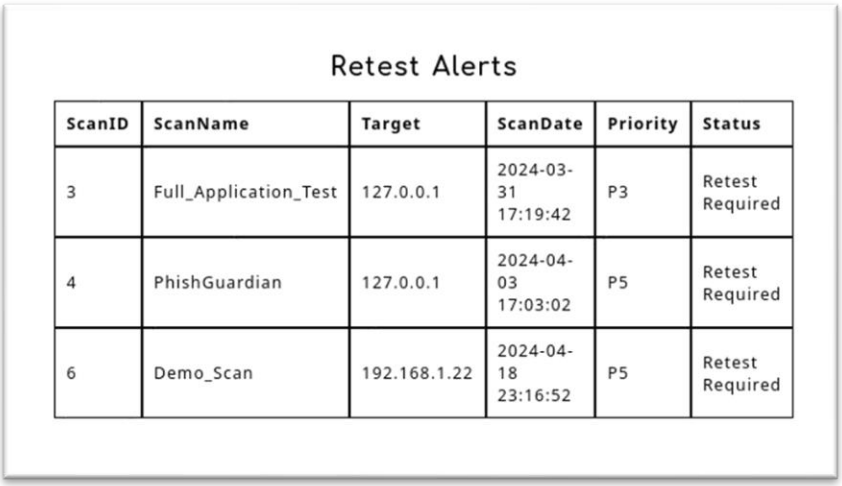


Figure 11: The Pen Testers Retest Alert with Demo_Scan reflected

The Pen Tester then prepares for a retest by configuring a new scan. During the scan configuration, the Pen Tester can "pair" the new scan with the original, ensuring that all re-tests are systematically linked. This pairing facilitates a comparison between the initial findings and the outcomes after remediation, which will streamline the process of verifying fixes effectively, maintaining a continuous loop of assessment.

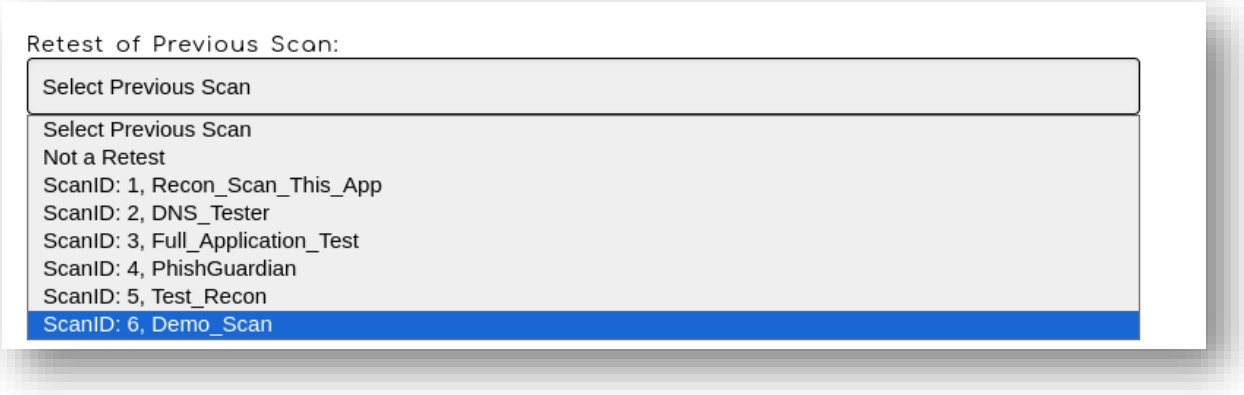


Figure 12: Linking the re-test to the previous Scan

3.5.5 Scan Closure

Once the re-test has been completed, the scan results are returned to the Analyst for another review. The Analyst can evaluate whether the necessary fixes have been successfully

implemented, comparing the original scan results with those from the re-test, and so ensures that each addressed vulnerability meets the standard required.

If the results are satisfactory, the Analyst can pass the scan onto the Engineer for another review; the Engineer can then close the scan, which will officially change its status to “Closed”. This final step will confirm that the vulnerabilities have been adequately addressed and the system’s security has been maintained.

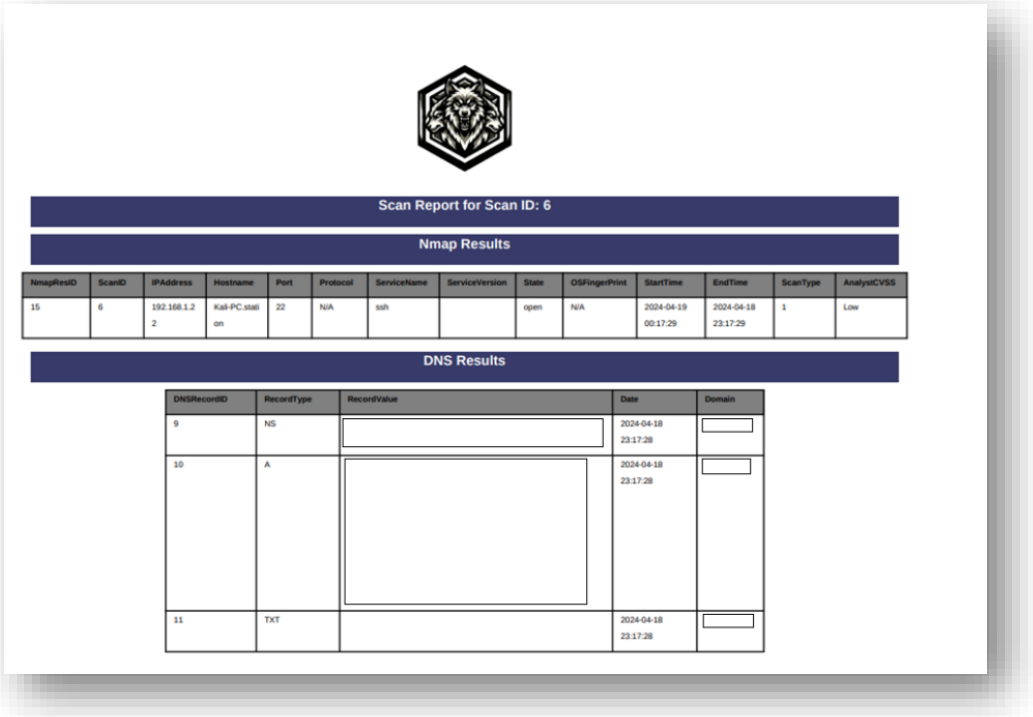
3.6 Report Generation

The system includes the capability to generate and export reports in both CSV and PDF formats, which allows users to share scan results in a format that is accessible. The code was developed using ReportLab, which is a Python library primarily used for creating PDFs from Python, and it allows users to compile reports with a specific layout.

The functionality to generate the PDFs starts by setting up the document template and defines the margins and orientation for the report. The data that will be entered needs to be formatted so the system's sections, NMAP, DNS, VAS and ZAP are configured into tables using another method – *prepare_data*.

The *prepare_data* function formats the data and converts it into a list of paragraph objects which can be rendered by ReportLab. To avoid errors and tedious layout issues, the text needs to be wrapped, so another function - *wrap_text,* was created. This function wraps the text to a specific cut off limit to avoid a *ValueError,* which occurs when a table row is overpopulated and would halt the generation of the PDF.

The report that is generated will detail every scan that was ran against that specific target.



Scan Report for Scan ID: 6

Nmap Results

NmapResID	ScanID	IPAddress	Hostname	Port	Protocol	ServiceName	ServiceVersion	State	OSFingerPrint	StartTime	EndTime	ScanType	AnalystCVSS
15	6	192.168.1.22	Kali-PC.station	22	N/A	ssh		open	N/A	2024-04-19 00:17:29	2024-04-18 23:17:29	1	Low

DNS Results

DNSRecordID	RecordType	RecordValue	Date	Domain
9	NS	[redacted]	2024-04-18 23:17:28	[redacted]
10	A	[redacted]	2024-04-18 23:17:28	[redacted]
11	TXT		2024-04-18 23:17:28	[redacted]

Figure 13: Example report - Redacted

3.7 Security and HTTP Status Codes

The system has several security measures implemented which assures against unauthorised access. These measures are important as to maintain the integrity of the systems data which is sensitive in nature and protect the actual systems as a whole.

3.7.1 Security Headers

The *add_headers(request)* function is at the top of the main app route, this includes the @after_request decorator, which means it automatically appends the security headers included to each route. A breakdown of the headers follows;

response.headers['Cache-Control'] = "no-cache, no-store, must-revalidate"

response.headers['Pragma'] = "no-cache"

response.headers['Expires'] = "0"

- These headers are used to control “caching”, by setting Cache-Control to no cache- no store -must revalidate, this will ensure that no information is stored in the browsers cache.

response.headers['X-Frame-Options'] = "DENY"

- When this is set to Deny, the webpage will be prevented from being framed by another site which effectively protects the web application from Clickjacking attacks

response.headers['X-Content-Type-Options'] = "nosniff"

- The X-content-type being set to 'nosniff', will prevent the browser from trying to MIME-sniff the content type of a response away from the one being declared by the server which will help prevent types of XSS attacks and vulnerabilities where the browser incorrectly guesses the MIME type.

response.headers['Referrer-Policy'] = "no-referrer-when-downgrade"

- Controls how much referrer information is included with each request. This means that a full URL will be provided while staying at the same protocol level which helps to protect sensitive information but allowing functionality for referrer logs.

response.headers['X-XSS-Protection'] = "1 mode block"

- Although not as Widley used as any more, (Jackson, 2023), this enables built-in XSS filters which are available in modern web browsers to protect against XSS attacks.

3.7.2 Automatic Escaping

Jinja2 automatically escapes all output transforming potentially harmful characters into HTML-safe encoded values, which means characters such as < or > become &lt and &gt,

(Delange, n.d.). This operation will neutralise any payload for Cross-Site Scripting attacks and injection attacks, to guard against vulnerabilities.

3.7.3 Password Hashing

When an administrator creates a new user account, they have to a set a password, the system will automatically hash this password using the bcrypt algorithm, which is specifically designed for the hashing of passwords, (Grigutytė, 2023). This ensures that if the database is breached, a threat actor can not see a list of user passwords in plain text.

3.7.4 Brute Force Lockout

The system has uses transience and tracks the number of failed login attempts for every user by username. If a user incorrectly enters the wrong login credentials three times in a row, the system will lock out the user for 20 minutes. This helps prevents unauthorised access from brute force attacks. The conditions of the lockout are configurable within the security class of the system and allow for adjustments to be more strict or lenient depending on security requirements.

3.7.5 Prepared Statements

Prepared statements are used throughout the system, parameterising the SQL commands, which separates data from command. These are used to enhance security for the SQL database, even though users can access the majority of pages, helping to prevent any risk from SQL injection.

3.7.6 HTTP Status Codes

The system uses HTTP status codes in communicating the state of requests between client and server, and an error handling system that uses these codes is implemented. Each status code is associated with a specific error message and an image that helps to visually represent the error. There is a mapping stored in a dictionary called *error_info*, where each code is linked to its own message and its image path.

If a user attempts to access a resource that they are not authenticated for example, (e.g. Pen Tester tries to access the Admin Dashboard), they will receive the error page with the 401-status code message and its corresponding image. This is true also for 404 errors for example if a page cannot be found. The system dynamically handles these errors with the *handle_error* function which retrieves the correct response details from *error_info* based on which error code was triggered.

The complete list of error codes that are setup to function in this way are as follows;

Table 12: Error handler status codes and message

Error handler Status Code	Message
@app.errorhandler(400)	Bad Request
@app.errorhandler(401)	Unauthorised access attempt detected. Please authenticate.
@app.errorhandler(403)	Forbidden. Access to this resource is denied.
@app.errorhandler(404)	Page Not Found
@app.errorhandler(405)	Method not allowed
@app.errorhandler(429)	Too Many Requests
@app.errorhandler(500)	Internal Server Error.
@app.errorhandler(503)	Service Unavailable.

3.7.4 Access Control

Access control is enforced throughout the system using the *role_required* decorator, which checks user roles against permitted roles stored in a session before granting access to specific pages of the system. This ensures that operations are managed and that only authorised users can access certain functionalities, with automatic handling of unauthorised (**401**) and forbidden (**403**) access attempts.

This approach to access control secures the application but also helps improve the user experience by providing clear, concise information and visual confirmation on why certain actions cannot be completed, which guides users appropriately.

4. Relation to Cybersecurity

The system's main objective is to enhance cybersecurity operation within a professional environment by integrating various security scanners, automating their functions and reducing the labour and time required for report generation. Outlined below are the core rationale behind the relation to cybersecurity.

4.1 Integration of Tools used In the Industry

The project employs recognised tools that are actively used in the cybersecurity industry; Nmap, (muhungu, V. j., 2024), OPEN-VAS, (Infosec, n.d.), and ZAP, (HG Insights, n.d), to conduct vulnerability scans across different networks and devices and by combining these tools into one integrated system, could potentially address key gaps in inefficiencies that are often found in the industry, (muhungu, 2024).

4.2 Scans carried out

This can be particular advantageous to environments where the response speed to vulnerabilities can impact on an organisation's security posture.

Streamlining both the scanning and reporting processes, this provides opportunities to identify the insights into the vulnerabilities quicker and more holistically which can deepen the insights into the detected vulnerabilities; which could in turn enable more effective

remediation strategies to be developed to mitigate and prevent recurrence, as well as increasing the response time in addressing the vulnerabilities.

By simplifying the vulnerability scanning process and report generation process, this system would enable a more efficient use and optimisation of technically skilled and proficient resources by reducing the manual involvement associated with such activities, therefore providing capacity to focus on for example, cyber security initiatives and strategies for the future.

This system can lead to a more enhanced cyber security strategy for organisations which can in turn protect users, customers and stakeholders alike.

5. Reflection

Personal Learning

The development of this system has provided multiple opportunities for personal development, from harnessing my technical capabilities and confidence in, for example, integrating Python with the web-based languages of HTML, CSS and JavaScript; using and testing the vulnerability scanners; and UI development by creating and styling multiple web pages. To developing transferable, professional skills such project planning, problem solving and presentation skills.

If I was to start the development of this system again, the below are examples of practices I would sustain going forward, and those I would do differently.

Sustain –

- **Development of a web application** – Building the system in the form of a web application was an approach that I would use again because it provided a simple way to organise the code and web pages, and was effective to style using CSS. The development of a web application enabled multiple web pages and Python classes to be implemented which enabled in the scale and size of the system itself.

- **Project Planning** – The approach to planning all aspects of this sizeable project supported the overall delivery by enabling a wide-ranging deliverable to be divided into tangible steps and sub-steps. This supported maintaining momentum throughout the development as there was clarity of the deliverables required week on week, and where any timelines shifted, enabled me to look forward to determine the impact and how / where to pivot to protect the overall delivery.
- **Resilience in problem solving** – The development of a system of this nature resulted in multiple instances of problem solving, idea generation and a solution focused mindset to be required. By appreciating there would be multiple scenarios in which this was required, and allowing space and time for such activity in my planning, it empowered me to maintain confidence in my ability to overcome obstacles, and my ability to deliver the system in the face of such challenges.

Do Differently -

- **Use of a different PDF library** – The learning curve for Report Lab was quite steep and required a significant time investment to build the functions. Taking time from the outset to source first-hand experience of the most optimal library to use for the purpose of generating PDFs would provide greater opportunity to balance time across wider topics.
- **Establishing a better method for regression testing** – Throughout the development, instances of implementing code fixes resulted in inadvertent knock-on impacts on other functions in other classes in the code. This required time to be dedicated to fixing such knock-on impacts. Establishing a process for fit for purpose regression testing at different stages of the development could enable changes to be implemented in a manner that protected the integrity of the code and minimise the time needed to develop further code fixes.
- **Preparation of supporting documentation** – The readiness of the material and content required to prepare the supporting documentation was impacted by the development of the system itself. By dedicating the primary focus to the actual development, the balance of time to prepare the supporting documentation was

a challenge. Planning for this aspect of the delivery more purposefully, and recognising the sections which could be prepared without the development itself advancing verus those that could not, or indeed conducting the relevant aspects of write ups in real time, could provide a more streamlined approach to delivery.

References

1. Bada, M. and Nurse, J.R.C. (2019) The social and psychological impact of cyberattacks, Emerging Cyber Threats and Cognitive Vulnerabilities. Available at: https://www.sciencedirect.com/science/article/abs/pii/B9780128162033000046 (Accessed: 27 October 2023).

2. Cryptography Development Team, (2023). 'Authenticated Encryption with Associated Data (AEAD)', Cryptography Documentation. Available at: https://cryptography.io/en/latest/hazmat/primitives/aead/#aes-gcm (Accessed: 22 October 2023).

3. Das, A. (2021) Top 10 security assessment tools, IndiumSoftware. Available at: https://www.indiumsoftware.com/blog/top-10-security-assessment-tools/ (Accessed: 22 October 2023).

4. Delange, J. (n.d) Python jinja2: Always Autoescape to avoid XSS attacks, Codiga. Available at: https://www.codiga.io/blog/python-jinja2-autoescape/ (Accessed: 19 April 2024).

5. Grigutytė, M. (2023) What is Bcrypt and how it works?, NordVPN. Available at: https://nordvpn.com/blog/what-is-bcrypt/#:~:text=Bcrypt%20is%20a%20valuable%20tool,to%20break%20the%20bcrypt%20hash. (Accessed: 19 April 2024).

6. HG Insights (no date) Owasp zed attack proxy (ZAP), Companies Using OWASP Zed Attack Proxy (ZAP), Market Share, Customers and Competitors. Available at: https://discovery.hgdata.com/product/owasp-zed-attack-proxy-zap (Accessed: 19 April 2024).

7. Hinch, D. (2023) Understanding nonces and their use in AES-GCM, LinkedIn. Available at: https://www.linkedin.com/pulse/understanding-nonces-use-aes-gcm-derek-hinch (Accessed: 18 April 2024).

8. IARM (2023) Why is a vulnerability assessment critical for your business? IARM Information Security. Available at: https://www.iarminfo.com/why-is-a-vulnerability-assessment-critical-for-your-business/ (Accessed: 20 October 2023).

9. IBM (2024) 4 eyed principle. Available at: https://www.ibm.com/docs/en/b2b-integrator/6.1.1?topic=principle-4-eyed (Accessed: 19 April 2024).

10. infosec (n.d) OpenVAS explained, infosec. Available at: https://infosec-jobs.com/insights/openvas-explained/#:~:text=OpenVAS%20is%20widely%20used%20across,weaknesses%20in%20their%20IT%20infrastructure. (Accessed: 19 April 2024).

11. Jackson, B. (2023) X-XSS-Protection - preventing cross-site scripting attacks, KeyCDN. Available at: https://www.keycdn.com/blog/x-xss-protection (Accessed: 19 April 2024).

12. Kariyawasam, I. (2021) Selecting the best AES block cipher mode (AES-GCM vs AES-CBC), Medium. Available at: https://isuruka.medium.com/selecting-the-best-aes-block-cipher-mode-aes-gcm-vs-aes-cbc-ee3ebae173c (Accessed: 22 October 2023).

13. muhungu, V. j (2024) Nmap: From movies to the most used tool in the industry, HackerNoon. Available at: https://hackernoon.com/nmap-from-movies-to-the-most-used-tool-in-the-industry (Accessed: 19 April 2024).

14. Naik , S. (2022) How has automated testing transformed the World Cyber Security, Times of India Blog. Available at: https://timesofindia.indiatimes.com/readersblog/afour-tech/how-has-automated-testing-transformed-the-world-cyber-security-46939/ (Accessed: 19 April 2024).

15. OWASP Cheat Sheet Series Team (n.d.). Cryptographic Storage Cheat Sheet. OWASP Cheat Sheet Series. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html (Accessed: 22 October 2023).

16. team, K. (2023) Unlocking the Power of AES-256 Encryption: A Comprehensive Guide, Kiteworks. Available at: https://www.kiteworks.com/secure-file-sharing/unlocking-the-power-of-aes-256-encryption-a-comprehensive-guide/#:~:text=The%20key%20strength%20of%20AES,unauthorized%20access%20to%20the%20data. (Accessed: 22 October 2023).