



National
College *of*
Ireland

A Hybrid of Cloud-based and Desktop Vulnerability Security Checker Application: VSCAN

TECHNICAL REPORT DOCUMENT

DANIEL KELLY X14532897

X14532897@STUDENT.NCIRL.IE

SUPERVISOR: DR CATHERINE MULWA

NATIONAL COLLEGE OF IRELAND |

Table of Contents

Table of Figures.....	5
Glossary of Terms.....	7
Project Declaration	8
Executive Summary.....	9
1 Introduction	10
1.1 Motivation and Background.....	10
1.2 Problem Statement	10
1.3 Aims and Objectives.....	11
1.4 Scientific Methodology and Approach Used.....	12
1.5 Contributions.....	13
1.6 Technologies Used	14
1.6.1 Java.....	14
1.6.2 NetBeans.....	14
1.6.3 MySQL Workbench	14
1.6.4 Amazon Web Services.....	15
1.6.5 Tomcat	15
1.6.6 Launch4J.....	15
1.7 Structure of Report	15
2 Literature Review on Computer System Vulnerabilities.....	17
3 Specification, Architectural and Technical Design of Cloud-based and Desktop Vulnerability Security Checker Application	20
3.1 Introductory chapter	20
3.2 User Requirements.....	20
3.2.1 Functional Requirements.....	21
3.2.2 Use Case Diagram	22
3.2.3 User Registration Requirement	23
3.2.4 User Login Requirement	25

3.2.5	Search Vulnerabilities Requirement	27
3.2.6	Requirement Perform Scan.....	28
3.2.7	Reset Forgotten Password	30
3.2.8	Save Scan Result Requirement	32
3.2.9	View And Delete Scan Requirement.....	34
3.2.10	Update The Database Requirement	36
3.3	Abuse Cases.....	38
3.3.1	Abuse Case Tamper with Database Information	38
3.3.2	Abuse Case Network Eavesdropping	40
3.4	Non-Functional Requirements	41
2.3.1	Performance/Response Time Requirement	41
2.3.2	Security Requirement	42
2.3.3	Reliability Requirement	43
2.3.4	Maintainability Requirement.....	43
2.3.5	Usability Requirement	43
2.3.6	Data Requirements	43
3.4.1	Environmental Requirements.....	44
3.4.2	Interface Requirements Specifications For VScan Application.....	44
3.5	System Architecture and Design	48
3.5.1	Presentation Layer	49
3.5.2	Business Logic Layer.....	49
3.5.3	Data Persistence Layer.....	49
4	Implementation and Testing of a Hybrid of Cloud-based and Desktop Vulnerability Security Checker Application	50
4.1	Introduction.....	50
4.1.1	JPA Entities.....	50
4.2	Implementation of the Components of VSCAN Application	52
4.2.1	Implementation and Testing of RESTful API Endpoints	52
4.2.2	Implementation and Testing of the Services performed by the API Endpoints	55
4.2.3	Implementation and Testing of Scanning Component for Vulnerabilities .	56
4.2.4	Implementation and Testing of Automatic Updating of Vulnerabilities	57

4.2.5	Implementation of Amazon Web Services	59
4.3	Implementation and Testing of VSCAN Security Components and Features	60
4.3.1	Implementation and Testing of User Authorisation	60
4.3.2	Implementation and Testing Input validation	61
4.3.3	Implementation and Testing JPE and NamedQueries	62
4.3.4	Implementation and Testing Https / TLS Protocol	62
4.3.5	Implementation and Testing Password Hashing PBKDF2.....	64
4.3.6	Implementation and Testing Error Handling – Generic Warnings	65
4.3.7	Implementation of the Separation of functional layers	67
4.3.8	Implementation Secure coding principles	68
4.3.9	Implementation and testing of the .exe Launcher	68
4.3.10	Implementation and Testing of the Forgotten Password Authentication .	69
4.4	Developed and Evaluated Graphical User Interface of the VSCAN application.	70
4.4.1	Login Screen	70
4.4.2	Forgotten Password Screen	71
4.4.3	Search Screen.....	71
4.4.4	Search Results	72
4.4.5	Search Expand.....	73
4.4.6	Perform Scan Page	74
4.4.7	Scan Results Page.....	75
4.4.8	Saved Page	76
5	User-Based Evaluations and Automated Testing of the Developed VSCAN Application	77
5.1	Introduction.....	77
5.2	Automated System Testing	77
5.3	User-Based Evaluation	78
5.3.1	Testing functionality	78
5.3.2	Survey Results	80
6	Challenges Encountered and Complexity of the Project	83
7	Conclusion and Recommended Further Development	84
7.1	Integrity checker using checksum	84

7.2	Scan Folder Directories	85
	References	85
	Appendix	89
7.3	Project Proposal	89
7.3.1	Objectives.....	90
7.3.2	Background	90
7.3.3	Technical Approach.....	91
6.3.4	Special resources required.....	92
6.3.5	Project Plan	93
6.3.6	Evaluation	93
7.4	Monthly Journals.....	94
7.5	Other Material Used.....	98

Table of Figures

Figure 1 Iterative and Incremental	13
Figure 2 Class Diagram	22
Figure 3 Register Diagram.....	23
Figure 4 Login Diagram	25
Figure 5 Search Diagram	27
Figure 6 Scan Diagram	29
Figure 7 Reset Pass Diagram	31
Figure 8 Save Diagram	33
Figure 9 Delete Diagram	35
Figure 10 Update Diagram	37
Figure 11 Tamper Abuse Case.....	38
Figure 12 Eavesdropping Abuse Case	40
Figure 13 Login Mock-up.....	44
Figure 14 Scan Mock-Up	45
Figure 15 Progress Mock-up	45
Figure 16 Display Mock-Up	46
Figure 17 Delete Mockup.....	47
Figure 18 System Architecture.....	48
Figure 19 Cve Entity	51
Figure 20 Entity Relationships	52
Figure 21 CheckToken endpoint	53
Figure 22 Search Vulnerability Service	56
Figure 23 Scanning Mechanism	57
Figure 24 Servlet Context Listener	58
Figure 25 Timing class	58
Figure 26 CheckSha method	59
Figure 27 AWS Deployment.....	60
Figure 28 Authorization Code	61

Figure 29 Input Validation Code	62
Figure 30 Unsecure Client.....	64
Figure 31 Password Hashing	65
Figure 32 Error Handling Code.....	66
Figure 33 Login Warning	67
Figure 34 Email Token.....	69
Figure 35 Token Email.....	70
Figure 36 Login Screen	70
Figure 37 Search Screen.....	72
Figure 38 Search Results Screen	72
Figure 39 search Expand Page	73
Figure 40 References Page.....	74
Figure 41 Perform Scan Page	74
Figure 42 Progress Bar	75
Figure 43 Scan Results Page.....	75
Figure 44 Saved Page	76
Figure 45 Test User Creation.....	77
Figure 46 test User Login	78
Figure 47 Test Add User Cve	78
Figure 48 Q1 Survey	79
Figure 49 User Experience	80
Figure 50 Future use	80
Figure 51 Frequency Survey.....	81
Figure 52 Future Functionality.....	81

Glossary of Terms

NVD -- National Vulnerability Database

JPA -- Java Persistence API

JPE -- Java Persistence Entity

REST -- Representational State Transfer

CVE -- Common Vulnerabilities and Exposures

BAH – Basic Authentication Header

GUI – Graphical User Interface

Project Declaration

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: DANIEL KELLY
Student ID: X14532897
Supervisor: DR CATHERINE MULWA

SECTION 2 **Confirmation of Authorship**

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarized overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: _____ Daniel Kelly _____ Date: ___ 13/05/2018 ___

Executive Summary

VScan is a Hybrid of a cloud- based API and a desktop-based application which aims to improve the security and integrity of a user's computer. This is being achieved by providing a clear outline of vulnerabilities associated with applications on the user's machine.

Currently there exists rampant vulnerability issues which are encountered by users interacting with computer systems and web-based application on daily basis. This project addresses the problems and challenges encountered by users, and provide supporting information allowing users to make informed decisions on the security of their systems. An investigation of existing literature on computer vulnerabilities was conducted and the results are presented in this report. In addition, during discussions with managers during work experience the topic of vulnerabilities was broached leading the candidate to deciding on the development of VScan.

The candidate has specified, designed and developed VScan which is a java-based desktop application which communicates with a REST styled API which in turn communicates with a MYSQL database. This database contains a subset of information which has been retrieved and formatted from the National vulnerability database. The NVD is an accumulation of vulnerabilities which is maintained by the American Government. It is updated regularly and provided for free.

The developed VScan application is intended for use by any persons who wish to improve the security of their computer. This solution is simple to-use and provides easily understandable descriptions of risks so that users with no technical skills will be able to check and assess the risk of an application in their systems. The solution is also capable of providing information and background discovery information of the problem so users with a technical know-how will be able to fully assess the vulnerabilities.

The developed VScan was also evaluated using customer feedback and a structured user survey the results of which have been summarised and presented in this report.

1 Introduction

1.1 Motivation and Background

Cyber security is a large field in today's economy. Especially with all the recent cyber-attacks such as the apple iCloud leaks and the more recent ransomware attacks which were responsible for locking a user out of their machine until the user payed money to a provided bank account. With all these attacks becoming more publicised in the news etc. people are starting to consider the safety of their data and personal information more seriously. The publication of the Ransomware attacks is where the conception of VScan began. Whilst on work placement I discovered a lack of knowledge of computer system vulnerabilities which I discuss more in Chapter 2 of this report.

With so many possible entry points into a user's machine it is incredibly difficult to properly secure it. VScan hopes to lessen this difficulty by providing users with a prioritised view of all vulnerabilities associated with applications on the user's machine. VScan is a tool with which users can find vulnerabilities, receive information on these vulnerabilities and assess whether these vulnerabilities are a threat to the integrity of their machine or not.

VScan has the potential to alert customers of a range of weaknesses. These range from, applications creating entry points to the user's local machine to simply unencrypted flows of data which provide an attacker with sensitive information that the user is transmitting.

1.2 Problem Statement

Currently there exists rampant vulnerability issues which are encountered by users interacting with computer systems and web-based application on a daily basis. A vulnerability is a weakness in a computer system which exposes a user to the possibility of a malicious entity performing unauthorised actions on their systems. These vulnerabilities range from low priority to high priority. High priority vulnerabilities include, i) exposure of user details, ii) data manipulation, iii) denial of services, iv) system damage, v) buffer

overflow and vi) breach of user privacy. In order to address this problem there is a need to specify, develop and evaluate a desktop application vulnerability security checker (VScan).

1.3 Aims and Objectives

The main goal of this project is to specify, design, develop and evaluate a Hybrid of Cloud-based and Desktop Vulnerability Security Checker Application (VScan) which allows users to heighten the security of their systems and address the problem specified (section 1.2). In order to achieve this goal, the following objectives have been specified.

Objective 1: Conduct an investigation of the literature in computer systems vulnerabilities

Objective 2: Based on the results of this literature and identified gaps the second objective is to develop and evaluate a hybrid cloud and desktop-based application security checker. This objective is further subdivided into several sub-objectives: i) identification of existing computing high and low vulnerabilities, ii) Implementation of RESTful API endpoints, iii) implementation of scanning component for vulnerabilities, iv) implementation of a search component for vulnerabilities, v) implementation automatic updating of vulnerabilities and vi) view saved vulnerabilities.

Objective 3: A documentation of the user specifications, system specifications, design and implementation of both the VScan API and VScan Application. Also present any findings resulting from evaluation and customer feedback.

The desktop-based application (VScan APP) allows users to scan their machines for applications or enter names of applications that they are considering installing on their machines. This then sends this information to the cloud-based application (VScan API) which then performs the required computation and retrieves any vulnerabilities associated with these applications and provides the users with details of the vulnerabilities. The Users can then make an informed decision and evaluate the risk of an application based on the information returned.

All the information sent between the application and the server must be handled securely and appropriately. The application will be retrieving information on the possible

vulnerabilities of a user's machine. In doing so the application is providing any potential attackers information concerning weak points in the user's machine. Therefore, all data exchanged between the server and the application must be encrypted using a secure protocol.

The database which vulnerabilities will be retrieved from must be kept as up-to date as possible. This involves creating a small function which shall run on the REST server every 2 hours. This function shall retrieve the modified data feed from the NVD website¹. It will parse this information withdrawing any required information and update the MYSQL database. This will be done every 2 hours.

1.4 Scientific Methodology and Approach Used

The core methodology used during the implementation of this project was a hybrid of iterative and incremental software development approach (Figure1). Because this project consisted of two separate java applications (VScan app and VScan API), an approach had to be taken to allow the creation of compatible parts of these sections at the same time. When developing each functionality of the VScan APP i.e. User authentication. The appropriate RESTful API endpoint was created, which is used to handle the user request and perform the appropriate action along with the appropriate client, used to send the information gathered from the GUI. This meant that each individual feature of the application could be coded with full attention allowing for a more efficient and productive use of time.

A RESTful approach was taken when developing the API endpoints for the project. This RESTful approach allows for stateless requests to be made to and processed by the API. This means that all the information needed to perform an action is contained within the request, therefore servers need not get overburdened with cookies, session storage etc.

¹ https://nvd.nist.gov/vuln/data-feeds#JSON_FEED

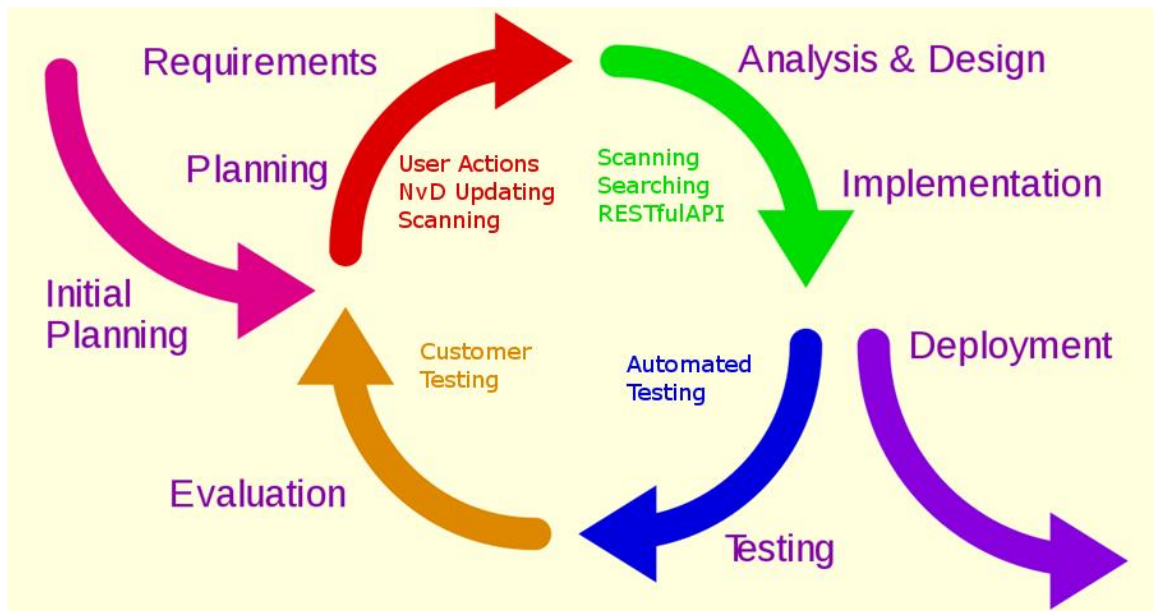


Figure 1 Iterative and Incremental

1.5 Contributions

This project has resulted into the following contributions. The main contribution is a fully developed and evaluated VScan vulnerability security checker.

The minor contributions resulting from this project include : i) identification mechanism of existing computing high and low vulnerabilities, ii) fully developed and evaluated RESTful API endpoints, iii) fully developed and evaluated scanning component, which allows users to scan for system vulnerabilities, iv) fully developed and evaluated search component which allows users to search for system vulnerabilities, v) fully developed and evaluated automatic updating mechanism of known vulnerabilities and vi) ability of users to view saved vulnerabilities. The second minor contribution is the results of the investigated literature in computer system vulnerabilities which are presented in chapter 2 of this technical report. The third minor contribution is the results of the user surveys (Structured Questionnaires) and feedback sessions conducted which are presented in chapter 5 of this report.

1.6 Technologies Used

1.6.1 Java

The Java coding language was used to create this entire project. Both the VScan application and API were created using java. Java provides some helpful implementations such as the Java Persistence API. This provided the RESTful styled VScan API with the portability of Java Persistence Entities. Essentially allowing objects to be directly persisted and stored in the database, for example a User object containing an email, ID and password can be directly manipulated without the appropriate SQL Queries being needed. Java also has direct compatibility with cryptographic procedures such as PBKDF2 which was used in this project².

1.6.2 NetBeans

NetBeans³ is a free and open-source IDE which allows for the developments of java applications. NetBeans is widely used in the industry when it comes to java developments, so it was a clear choice for use during this project. NetBeans also has a very useful design element which allows for the creation of java swing GUI's by way of dragging and dropping elements.

1.6.3 MySQL Workbench

MySQL workbench⁴ was used throughout the creation of this project. Workbench allows for a graphically appealing representation of the database structure and internal data. This was essential during the creation of the project.

² <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

³ <https://netbeans.org/downloads/>

⁴ <https://dev.mysql.com/downloads/workbench/>

1.6.4 Amazon Web Services

Amazon Web Services⁵ is a cloud platform which provides an incredible amount of functionality to users. For this project the VScan API has been deployed on to Amazon's Elastic Beanstalk which essentially creates an instance of a Linux machine running the API which is accessible world-wide. The MySQL database has also been deployed Amazon's RDS Services.

1.6.5 Tomcat

Tomcat⁶ is the server used to implement the VScanAPI. Originally the project began using jetty, but jetty became very difficult when attempting to incorporate HTTPS in the project. The API was then moved over to the tomcat server. This paired with the Amazon Web Services made for an easy implementation of my RESTful API.

1.6.6 Launch4J

Launch4J⁷ was used to wrap the JAR file created by the NetBeans project with a .exe launcher. This provided the functionality of directing the user to the oracle website if they did not have the Java Runtime Environment installed.

1.7 Structure of Report

The rest of the report is structured as follows.

Chapter 2 presents an investigation of the literature of the computer system vulnerabilities.

Chapter 3 specifies, and presents the architecture and technical design of the VScan project

⁵ <https://aws.amazon.com/>

⁶ <http://tomcat.apache.org/>

⁷ <http://launch4j.sourceforge.net/>

Chapter 4 Presents the implementation and testing of components of the VScan API and VScan Application along with Security features implemented.

Chapter 5 presents the results of user-based evaluations and automated testing.

Chapter 6 presents the Challenges Encountered which added further complexity to the VScan Project

Chapter 7 Presents a final Conclusion of the overall project and further development recommendations

Followed by References and the appendix, containing the original report proposal and monthly journals

2 Literature Review on Computer System Vulnerabilities

This literature Review was conducted on recently published academic journals for the purpose of researching the topic of computer vulnerabilities. This review aims to provide background research toward the inception of the project “VScan”, highlighting a need for the project and explaining desirable outcomes.

Computer Vulnerabilities are: any weakness related to software, hardware or networks which could lead the execution of any unauthorized action. These vulnerabilities may range from simple configuration errors, whereby a developer has forgotten to implement an existing component such as input validation in a form or complex design issues such as outdated algorithms (*Rick Kuhn, Mohammad Raunak, Raghu Kacker*⁸). It is incredibly difficult to obtain a 100% vulnerability safe application as vulnerability types and attack vectors are growing day by day along with the measures incorporated to minimize or halt their effect. Vulnerabilities are constantly adapting as clearly visualized by looking at the OWASP Top 10. The OWASP Top 10⁹ is a document portraying the most critical and most prevalent vulnerabilities in web applications which is updated every few years. The most recent version of this document was created in 2017. Looking at a comparison of this document and the 2013 version it is clearly visible that security standards have changed and adapted to include new vulnerabilities and drop some outdated/ less trending vulnerabilities.

There are methods in place to detect and prevent the publication of applications with vulnerabilities namely through active scanning and incorporating overall techniques to the production of applications. Active Scanning software such as Metasploit¹⁰ which attempt to detect vulnerabilities in applications and/or web applications by executing test cases listing successful or partially successful cases against vulnerabilities. The proposed

⁸ Kuhn, R., Raunak, M. and Kacker, R. (2017). An Analysis of Vulnerability Trends, 2008-2016 - IEEE Conference Publication. [online] ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8004385/authors> [Accessed 12 May 2018]

⁹ https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

¹⁰ <https://www.metasploit.com/>

vulnerability assessment and penetration testing approach by *Jai Narayan Goel and B.M. Mehtre*¹¹, includes step-by-step actions which can be taken in the development life cycle to discover and mitigate these vulnerabilities. However, applications such as Metasploit are quite technical and require a knowledge base to use. These active scanners are typically not used by the customers of these software's to protect their individual systems.

Consumers of these applications have a rather limited approach when it comes to protection of their systems. Anti-Virus protection is the leading method most consumers employ however this is not complete. Anti-Virus protection software's such as Bitdefender protect a user's machine against infections such as Viruses, worms and other schemes attackers may present. Which according to *Alan Newson*¹² were the biggest concern of 2005. However, formal code inspection can prevent up to approximately 65% of application errors from being published (*Rick Kuhn, Mohammad Raunak, Raghu Kacker*). Which means that a clear majority of vulnerabilities being published are associated with configuration/development errors and not the effect of virus/worm/malicious code injections. This leaves a gap in the detection of vulnerable systems due to the internal errors and misconfigurations.

This is what lead to the discussion of vulnerability applications during my work placement of 2017. Upon broaching the topic of some final year project ideas with a managing director of the company the topic of vulnerability understanding and detection was instigated. The conclusion drawn from this conversation can be surmised as, a lack of information in the field of system vulnerabilities is found throughout commercial and private sectors. After consulting further with department members, it was found that no

¹¹ Geol, J. and Mehtre, B. (2015). Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. *Procedia Computer Science*, [online] 57, pp.710-715. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050915019870?via%3Dihub> [Accessed 12 May 2018].

¹² Newson, A. (2005). Network threats and vulnerability scanners. *Network Security*, [online] 2005(12), pp.13-15. Available at: <https://www.sciencedirect.com/science/article/pii/S1353485805703147??> [Accessed 12 May 2018].

team members actively researched or knew of the vulnerabilities that were affecting their machine from the installed applications they used for developments or personal use. Out of curiosity the question was asked to multiple other persons in computer studies and non-technical persons, with a consensus of a lack of vulnerability research. The National Vulnerability Database (NVD) was then introduced to me by the same manager. The National Vulnerability Database was used in the study by (*Rick Kuhn, Mohammad Raunak, Raghu Kacker*) and many more like it. It is maintained directly by the American Government causing it to be a source of secure and up-to-date data which is freely available for use. VScan was envisioned as an application which bridges the gap created by active scanning and anti-malware to allow users to make informed decisions based on vulnerabilities found relating to the applications they wish to install or have already installed on their systems.

3 Specification, Architectural and Technical Design of Cloud-based and Desktop Vulnerability Security Checker Application

3.1 *Introductory chapter*

This chapter of the report will begin by defining in detail the user requirements, which are a list of objectives which the system must complete from a user's perspective. These requirements set the foundation of what a user may wish to achieve when using the VScan Application. Abuse Cases shall be shown which shall provide insight into the requirements needed for the security of the project. The non-functional requirements of the project shall then be discussed focusing on specifications which the system must meet in order to judge the operation of the system. Finally, the system architecture of this project shall be presented in great detail, following the three-tiered architecture of presentation layer, business logic layer and data persistence layer.

3.2 *User Requirements*

The User requirements for this product consist of a set of objectives the system must complete from the user's perspective. Below is a list of the core objectives in for this product which will allow the product to meet the user's requirements.

Objectives:

Objective1: The User must be able to securely register/ login to the application.

Objective2: The System must allow users to scan their computer for installed applications. This list of applications shall then be queried against the database, thus returning a list of vulnerabilities associated with the applications on the user's machine.

Objective3: The System must allow the user to search for an application that they have not yet installed. This will again allow the user to make an informed decision in whether to install the application or not.

Objective4: The System must handle the transfer of information securely. When the system is transferring the data pertaining to the vulnerabilities, confidentiality is of an utmost priority, exposure of this list of vulnerabilities may lead to the exploitation of these vulnerabilities.

Objective3: The System must allow the Users to save, view and delete any results they wish to monitor after running the scan or performing the search. This will allow the User to monitor any vulnerabilities they currently have on their machines.

Objective4: The System must provide the most up-to-date vulnerabilities associated with the application. This will be achieved through the constant updating of the database with the Restful interface.

Objective5: The System must prove the vulnerabilities in a prioritized view. This will alert the user to more serious risks first.

3.2.1 Functional Requirements

This section shall be listing a brief overview of the functional requirements of which will be discussed in more depth in the sections following:

1. Register new user

The user shall be able to use their email and a password as login details to create an account with VScan allowing them to login to the system in the future.

2. Login/logout of application

The User will be able to then login to the system (as long as they have an account created) and logout of the system once they have finished using it.

3. Search the Database

The user will be able to enter an application name which will search the database and return any vulnerability information associated with that application.

4. Perform scan

The user will be able to initialize a scan of the system registry which will return a list of applications installed on the Users machine. This will then be queried against the database and thus return a list of vulnerabilities to the User.

5. Save scan result

The user will be able mark any vulnerabilities in this scan that they would like to save for later viewing. The user will then click save saving these marked results.

6. View/Delete saved scan result

The user will be able to navigate to past saved results allowing them to view vulnerability details they have saved.

7. Resetting Password

In the event that a user forgets their password, they may reset this password by way of 2 factor authentication. The user Can enter a 5-character code sent to their email which will grant them access to change their password.

8. Updating the database

The System will update the contents of the database using the provided NVD website database.

3.2.2 Use Case Diagram

The Use Case Diagram provides an overview of all functional requirements.

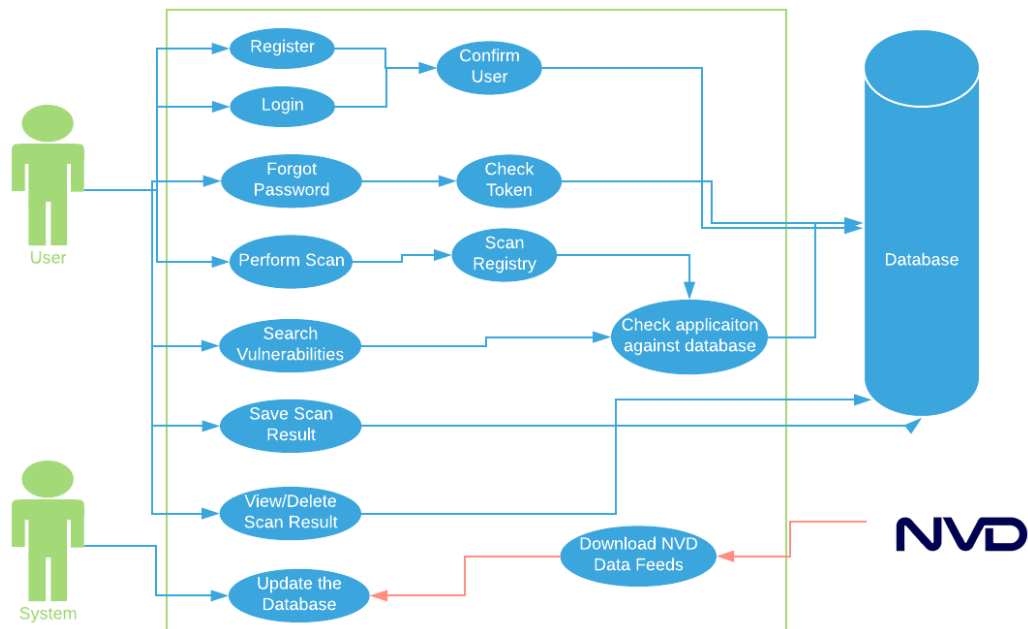


Figure 2 Class Diagram

3.2.3 User Registration Requirement

Description & Priority

This use case will allow the User to register an account with VScan, thus allowing them to use the services the application provides and is essential if a user wishes to use the application.

Use Case

Scope

The scope of this use case is for the user to be able to register to use the java application

Description

After installing the application and opening the program the user will be presented with a screen allowing them to register to use the service

Use Case Diagram



Figure 3 Register Diagram

Flow Description

Precondition

The application has been downloaded and the user is not registered.

Activation

This use case starts when a user loads the application where they will be presented with a login/ register page.

Main flow

1. The System displays a login/register screen.
2. The User enters their email and password details into the registration form.
3. The User clicks create an account.
4. The system confirms that the email format is valid, and the password has at least 2 numbers. (A1)
5. The system confirms that there has been no account created with this email already by checking against the database. (E1)
6. The System confirms registration and logs the user into the application.

Alternate flow

A1 : Invalid user email format

1. The system notifies the user that the email is not valid
2. The User re-enters a valid email
3. The use case continues at stage 5 of the main flow

A2 : Invalid user password

1. The System notifies the user that the password is not valid
2. The User re-enters a valid password
3. The use case continues at stage 5 of the main flow

Exceptional flow

E1 : An account is already associated with this email

1. The System notifies the user that an account has already been created with this email
2. The System asks the User to use a different email or log in using the current one
3. The use case ends.

Termination

User is notified of a successful registration.

Post condition

The user is successfully logged into the application.

3.2.4 User Login Requirement

Description & Priority

A user may login to the application using details they have registered with. This will allow the user to use the services provided by the application. The user must also be able to logout when the wish to.

Use Case

Scope

The scope of this use case is for the user to be able to login/ logout of the java application.

Description

This use case describes the ability of the user to login and logout of the application.

Use Case Diagram

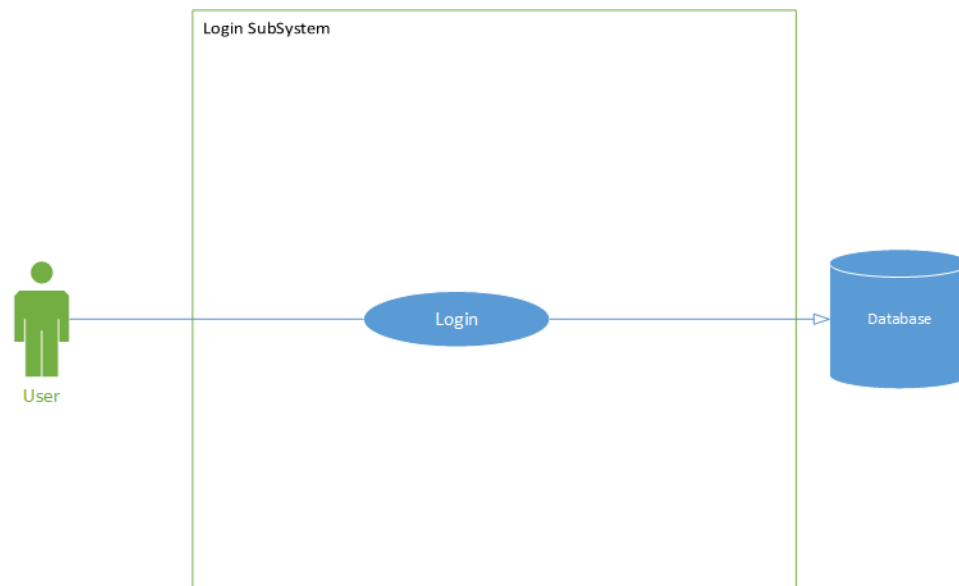


Figure 4 Login Diagram

Flow Description

Precondition

The System has been started. The User has already registered for an account.

Activation

This use case starts when a User starts the application and is presented with the login/registration page.

Main flow

1. The System displays a login/register screen.
2. The User enters their email and password details into the login form.
3. The User clicks login to account.
4. The system confirms that the email format is valid.
5. The system checks the details entered against the database to retrieve the Users details. (E1) (A1)
6. The System advances the User to the home page where they can select a service to use.
7. When finished using the application the User clicks the logout button.
8. The system finished the session and logs the User out.

Alternate flow

A1 : Invalid email format

1. The system notifies the user that the email format is invalid
2. The User re-enters their email in a valid format
3. The use case continues at position 4 of the main flow

Exceptional flow

E1 : The User has not registered

1. The system notifies the User that an account does not exist with these details.
2. The User clicks ok
3. The Use Case ends.

Termination

The System notifies the user of a successful login then goes to the homepage.

Post condition

The system waits for the User to select a service on the home page.

3.2.5 Search Vulnerabilities Requirement

Description & Priority

This use case describes the ability of the user to search for an application and return all the possible vulnerabilities associated with it. The User will enter an application name to search and be provided with a list of vulnerabilities associated with it.

Use Case

Use Case Diagram

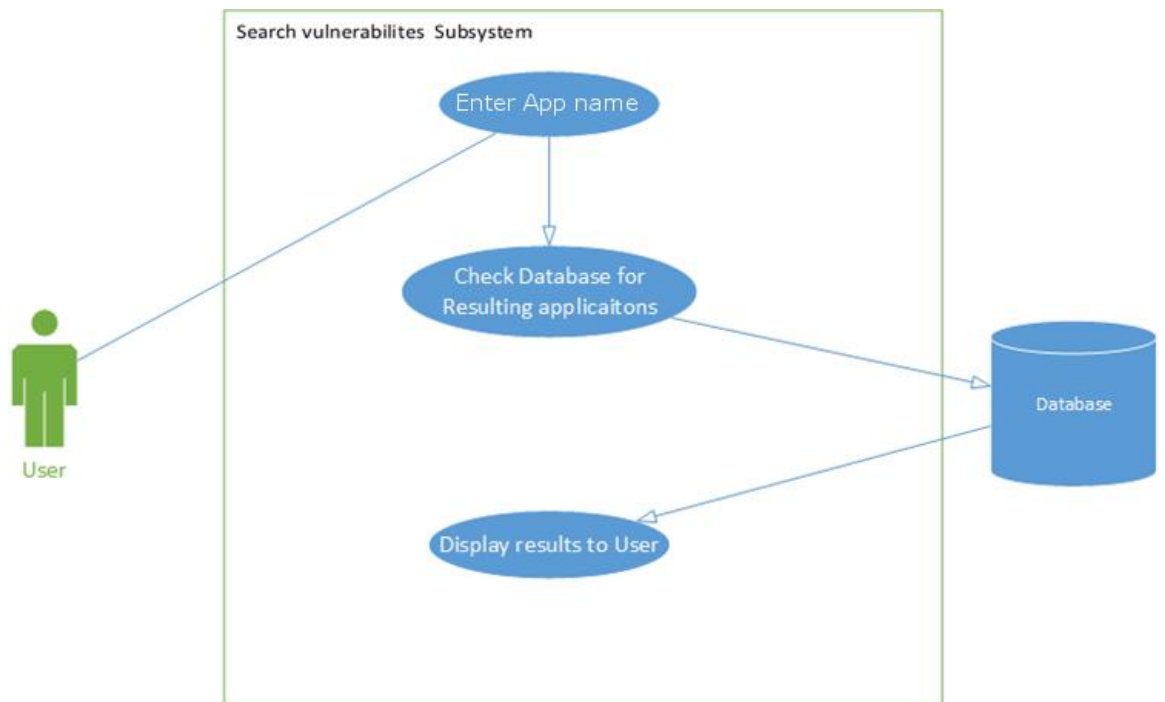


Figure 5 Search Diagram

Flow Description

Precondition

The System has been initialized and the user has logged into the application.

Activation

This use case starts when a User selects the Search option from the home page.

Main flow

1. The User enters in an application name into the form.
2. The User then clicks search.
3. The System will send this information to the database. (A1)
4. The database will return any vulnerabilities associated with the application.
5. The System will display these vulnerabilities in a prioritized view from high to low. (E1)
6. The User can click on a scan and view more details about it.
7. The system will display these details in a pop-up.

Alternate flow

A1: The User cancels the Search

1. The User clicks cancel
2. The System stops searching and returns the user to the previous page.
3. The use case continues at the start position 1 of the main flow

Exceptional flow

E1: The System finds no applications/vulnerabilities.

1. The System returns no files/vulnerabilities.
2. The System notifies the user that there are no files/vulnerabilities.
3. The use case ends.

Termination

The system presents the vulnerabilities to the User.

Post condition

The system waits for the user to either save or discard the information provided.

3.2.6 Requirement Perform Scan

Description & Priority

This use case describes the ability of the User to scan the windows registry which will return application file names and return them against the database. Thus, returning and displaying any vulnerabilities it finds.

Use Case

Scope

The scope of this use case is to search the windows registry and provide a user with any vulnerabilities related to applications installed on the system.

Description

This use case describes the ability of a User to scan a directory which will return any vulnerabilities associated with applications in that directory.

Use Case Diagram

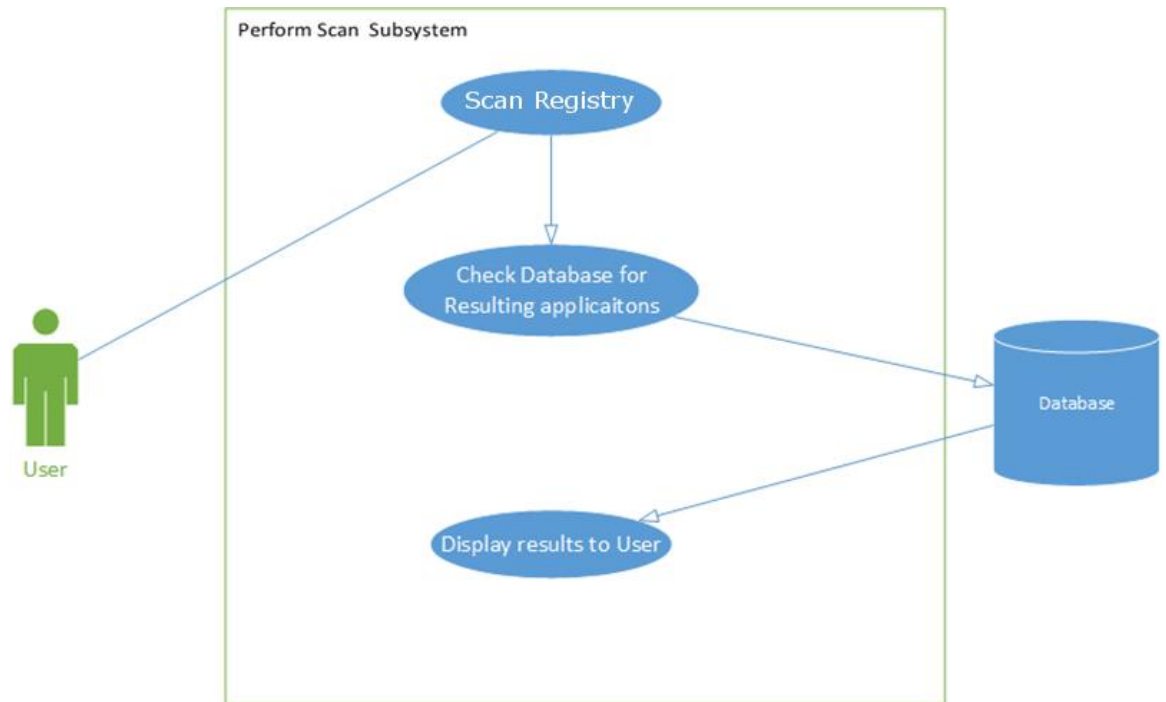


Figure 6 Scan Diagram

Flow Description

Precondition

The System has been initialized and the user has logged into the application.

Activation

This use case starts when a User selects the Scan option from the home page.

Main flow

1. The User enters in a folder location into the form.
2. The User then clicks scan now.
3. The System will run a command to scan the entered directory for any .exe files.
4. The System will take the name and product version from the .exe files.
5. The System will check these details against the database.
6. The System will return any vulnerabilities found for each application found.(E1)
7. The User can click on a scan and view more details about it.

Exceptional flow

E1 : The System finds no applications/vulnerabilities.

1. The System returns no files/vulnerabilities.
2. The System notifies the user that there are no files/vulnerabilities.
3. The use case ends.

Termination

The system presents the vulnerabilities to the User.

Post condition

The system waits for the user to either save or discard the information provided.

3.2.7 Reset Forgotten Password

Description & Priority

This use case describes the ability of the User to reset their password in the event for which they forget their password.

Use Case

Scope

The scope of this use case is to allow a user to recreate a password to be associated with their account, after performing a 2 factor authentication whereby the user receives an email containing a token which acts as proof of identity.

Use Case Diagram

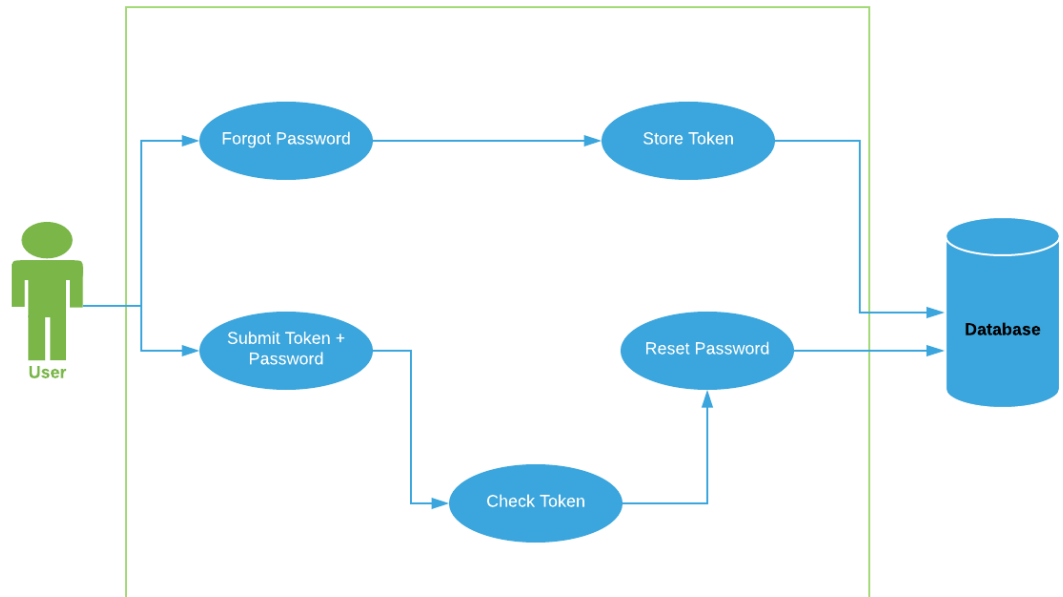


Figure 7 Reset Pass Diagram

Flow Description

Precondition

The User has already registered for an account.

Activation

This use case starts when a User selects the forgotten password section from the login/register page.

Main flow

1. The User clicks the forgotten password link on the login/register page.
2. The User enters their email in the provided form. (A2)
3. The System sends an email containing a newly generated token.
4. The System stores this token in the database.
5. The User provides enters this token into the form.
6. The System verifies the token is correct (A1)
7. The User then enters a new password to be saved. (A3)
8. The System saves the new password. (E1)
9. The System returns to the login page.

10. The system deletes the token from the database.

Alternate flow

A1 : The User enters the wrong token .

1. The System displays an error warning the User.
2. The User enters the correct token.
3. The Use case continues from step 5.

A2 : The User enters the email.

1. The System displays an error warning the User.
2. The User enters the correct email.
3. The Use case continues from step 2.

A2 : The User enters the wrong password .

1. The System displays an error warning the User.
2. The User enters the correct password.
3. The Use case continues from step 7.

Exceptional flow

E1 : The User clicks back

4. The request is cancelled to reset the password.
5. The System returns to the Login page.

Termination

The System returns the user to the Login/Register page.

3.2.8 Save Scan Result Requirement

Description & Priority

This Use case is a medium priority. It is not detrimental to the project but is a nice feature allowing the User to save any vulnerabilities for later viewing / monitoring.

Use Case

Scope

The scope of this use case is to save any vulnerabilities the User wishes to look at again.

Description

This use case describes the ability of a User to click on a vulnerability and save it for later viewing in a tab of the home screen. This allows the User to monitor any vulnerabilities on their machine and possibly delete old ones that are no longer relevant to them.

Use Case Diagram

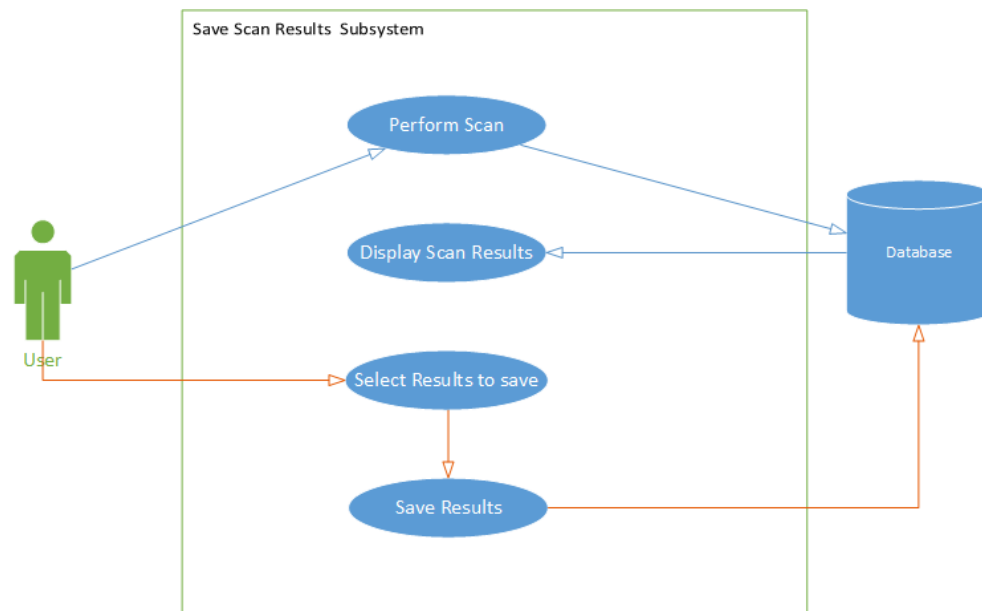


Figure 8 Save Diagram

Flow Description

Precondition

The System has been initialized and the user has logged into the application. The User has performed a scan and is now looking at the results of the scan.

Activation

This use case starts when a User has performed a scan and is looking at the results from this scan.

Main flow

1. The User clicks on which of the results they want to save, highlighting them.
2. The User then clicks the save button.
3. The System stores these vulnerabilities in the database associating them with the user.
4. The System notifies the user that the vulnerabilities were saved successfully.
5. The User clicks ok.
6. The System returns the User to the list of results.

Alternate flow

Exceptional flow

Termination

The System saves the vulnerabilities.

Post condition

The system waits for the user to choose what to do next.

3.2.9 View And Delete Scan Requirement

Description & Priority

This use case will describe the ability of the User to view any saved scans that they have stored. It will also allow them to delete any scans that are no longer relevant .

Use Case

Scope

The scope of this use case is view any saved scans that have been stored previously

Description

This use case will describe the ability of the User to view any saved scans that they have stored.

Use Case Diagram

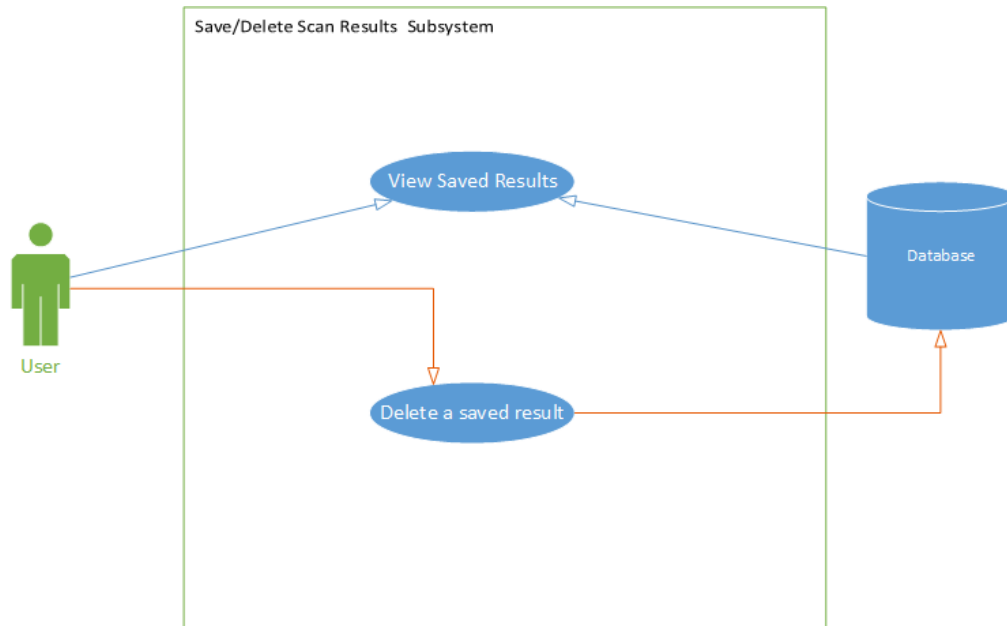


Figure 9 Delete Diagram

Flow Description

Precondition

The System has been initialized and the user has logged into the application.

Activation

This use case starts when a User selects the view saved Scans option from the home page.

Main flow

1. The System displays previously saved scans. (E1)
2. The User can scroll through these scans and select one for more details.
3. The System will display a pop up with more details on the scan results.(A1)
4. The User can exit this pop-up by clicking ok.

Alternate flow

A1 : The User deletes a result

1. The User clicks on delete result
2. The System removes the result form the list of saved results on the database.
3. The use case continues from point 2 in the main flow.

Exceptional flow

E1 : The System finds no saved results .

1. The System tells the User to perform a scan in order to save the Users results.
2. The use case ends.

Termination

The system presents the saved vulnerabilities to the User.

Post condition

The system waits for the user to either view or discard the information provided.

3.2.10 Update The Database Requirement

Description & Priority

This Use case is a big priority as it will allow the vulnerability database to be up to date.

This use case will allow a RESTFUL java app on the server download a file from the NVD website, parse it and upload the relevant data on my database.

Use Case

Scope

The scope of this use case is to update the database for any new vulnerabilities that have been added to the NVD database.

Description

This use case describes the ability of a RESFUL java application to update vulnerabilities in the database.

Use Case Diagram

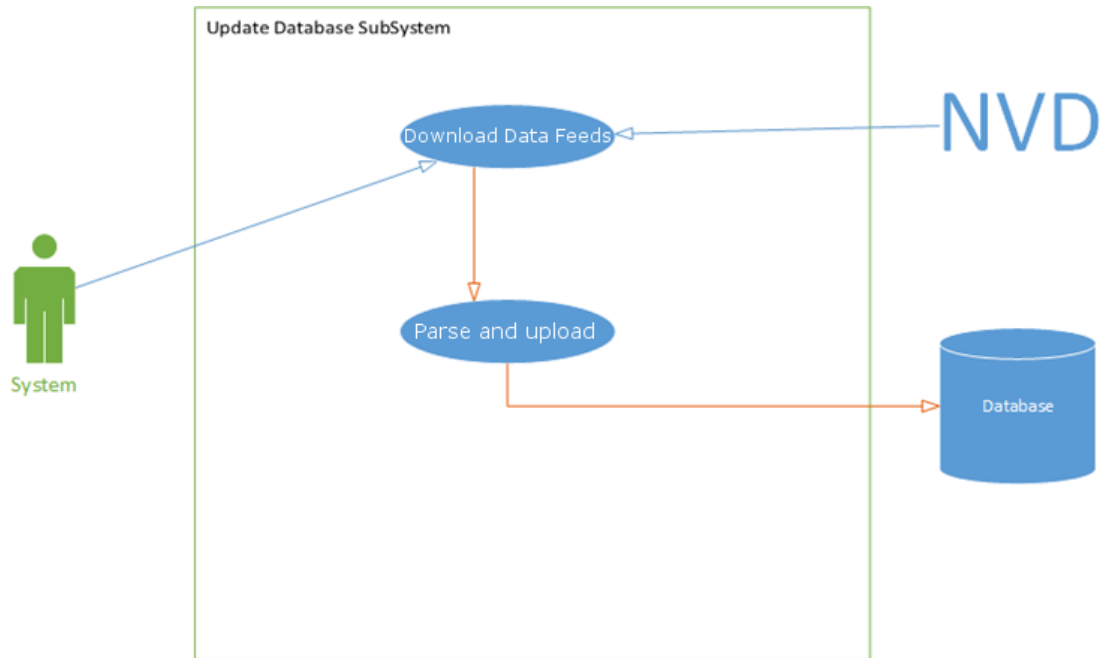


Figure 10 Update Diagram

Flow Description

Precondition

The Java application has been set to run on an interval of 2 hours.

Activation

This use case begins when the restful interface is set to run on the 2 hour mark.

Main flow

7. The System downloads a file from the NVD website.
8. The System then parses the JSON file which is downloaded
9. The System takes relevant information and creates a new JSON file from this.
10. The System then puts each JSON object into the database, replacing entries or deleting entries based on the information in the file.

Termination

The System updates the vulnerability

Post condition

The system waits for the next 2-hour mark to reperform this task.

3.3 Abuse Cases

3.3.1 Abuse Case Tamper with Database Information

Description

This abuse case describes the ability of an attacker to tamper with the information in the database. This can mean entering SQL statements into forms that will be going to the database with an end goal of deleting, updating or reviling confidential information.

Abuse Case Diagram

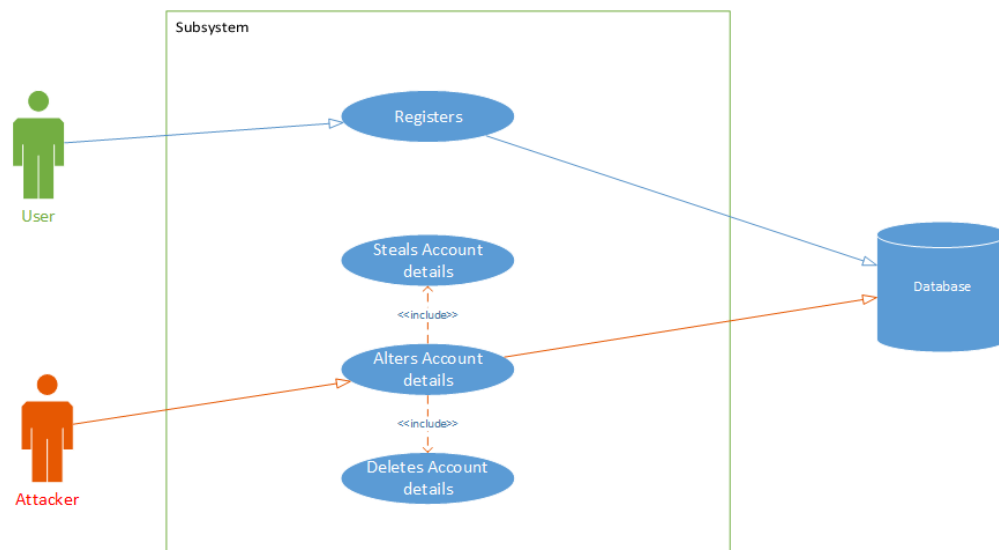


Figure 11 Tamper Abuse Case

Flow Description

Activation

This use case begins whenever an attacker or malicious entity is submitting information on a form in the java application which in turn will communicate to the database.

Main flow

1. The attacker enters information into the form to understand what information is returned. (a1)
2. The attacker then enters an SQL statement in an attempt to return information from the database or alter information in the database.
3. The System then performs the query which either alters the database or provides the attacker with sensitive information. (E1)

Alternate Flow

A1. The attacker introduces a query error which will respond with a database error allowing the attacker to garner more information about the structure of the database. (continues from point 2 of the main flow).

Exceptional Flow

E1. The System never performs the query because the System recognizes that it is an unauthorized query or access to the database.

Harm

1. The Attacker gains login information from clients- leading to the attacker possibly gaining information about the risks of the client's computers.
2. The Attacker deletes client information leading to a loss of accounts.
3. The attacker creates false accounts
4. The Attacker changes account information leading to the Attacker taking control of accounts / denying access to accounts.

Security measures

1. The first step in counteracting this abuse case would be validation of inputs, ensuring that the information entered is of the correct format.

2. The System should also not divulge all information that is available to it, meaning the System should not display internal errors to the public, so they cannot abuse this knowledge.

3.3.2 Abuse Case Network Eavesdropping

Description

Through the use of a network eavesdropping tool(sniffer) the attacker is able to gather the information that is transferred between the User and the server across a network.

Abuse Case Diagram

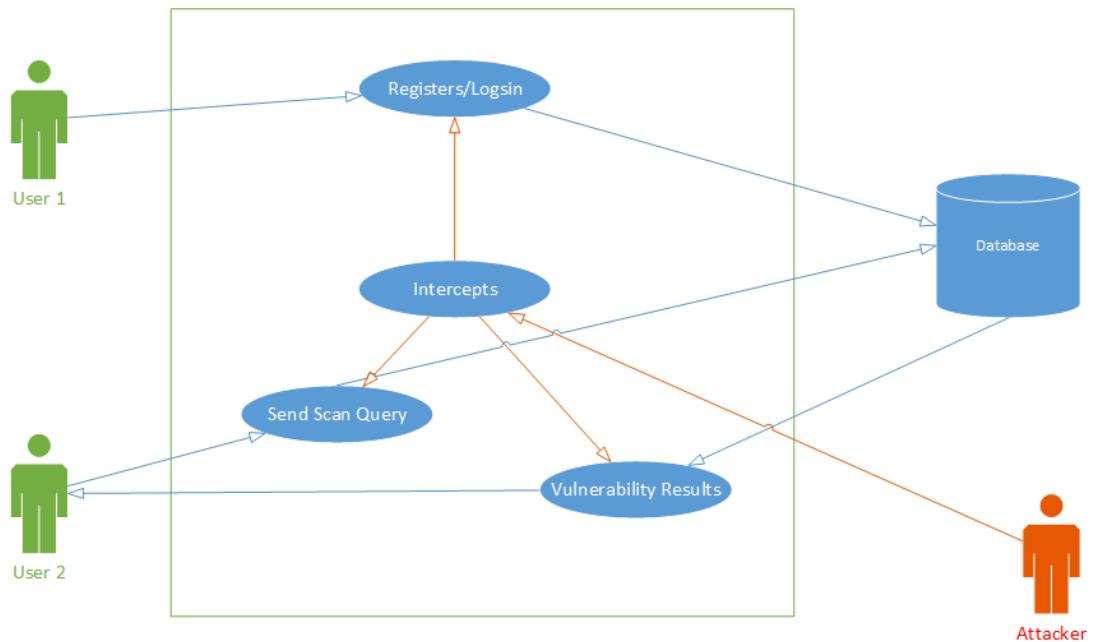


Figure 12 Eavesdropping Abuse Case

Flow Description

Activation

This use case begins whenever a User is sending information from the application to the server across a network.

Main flow

1. The attacker installs a network monitoring software on a computer.

2. This software will intercept the information sent by the user or server.
3. The Attacker then has access to the confidential information.

Exceptional Flow

E1. The System and server encrypt all data that is sent between each other, this ends the abuse case as the attacker cannot de-encrypt the data without the necessary key.

Harm

1. The Attacker can gain account information of the client, allowing them to have access to this account.
2. The Attacker can intercept information based on the vulnerability of the Users computer, allowing them to exploit these vulnerabilities.

Security measures

All information that is exchanged between the Users client and the server should be encrypted, to achieve this VScan will be employing TLS protocol. Which shall be discussed in more detail in the security section below.

3.4 Non-Functional Requirements

2.3.1 Performance/Response Time Requirement

The performance time of the project is crucial to the usability of the application because the application will be scanning the registry which can contain many entries, it will need to perform this scan in a timely manner. A feature of this will be to the total amount of entries it has found along with the current entry it is scanning for vulnerabilities, there shall also be a progress bar presented to the user. The application will need to communicate with a server, so the network speeds of the user and host will need to be considered when attempting to scan. Communicating over the network means that network errors will need to be handled appropriately or this could result in the user not being able to use the application, errors such as a non-existing directory etc. should be displayed to the User so they can rectify any mistakes. Whereas database errors should be handled and hidden to the user allowing the admin to view them only.

2.3.2 Security Requirement

When developing the application, it is clear that there shall need to be a plethora of security precautions implemented. The application is highlighting risks to a person's computer, this information must be secured and protected to protect the integrity of the user's machine. Security aspects of this project are of a major importance and shall be discussed thoroughly in the Implementation section of this report. Below is a summary of the necessary requirements to be met.

This will begin with ensuring the user has a strong password. The password will need to have a capital letter, a number and a special character. This is a small but effective step that will need to be done by the user.

The application will have to have preventative steps for SQL Injection ensuring that unauthorized persons cannot steal information i.e. Emails and passwords of the applications users. It is also important that they cannot steal any information about the current risks of the user on their applications. These steps will include input validation, prepared SQL statements, this way if there is anything out of the ordinary the System will know not to proceed with the Query.

All communications between the user and the database shall be encrypted using the Transport Layer Security protocol. This protocol encrypts the HTTP data sent between the client and the API thus ensuring that data in transit is secured from eavesdropping etc.

When developing the application, it is important that VScan employ secure coding practices such as, making appropriate variables/ classes private and or private whenever possible. It's important that VScan minimize the usage of named inner classes, keeping inner classes anonymous makes them more secure. Secure coding practices will allow my final product to be a more secure application not allowing malicious entities to have an entry point into the application itself.

2.3.3 Reliability Requirement

The reliability of the application will be taken into consideration in the development process. Users may rely on this product to ensure the safety of some of their applications therefore it is important that this application is kept as bug free as possible. The end of this project will be concluded with a user testing approach. This user testing allowed for a survey of the final product but also general feedback and usability feedback which allowed for some small modifications to be made, allowing for less bugs and an overall more reliable product.

2.3.4 Maintainability Requirement

This application lifecycle will be iterative therefore there shall be high motivation to keeping it modular. This will allow for a high level of flexibility and maintainability. It is important to be able to release patches/updates or even hotfixes and keeping the project in an agile development methodology is important to be able to achieve these in short periods of time. Development will include working on different parts of the application during each iteration therefore, it will be easy keep the application in separate modules for this will also allow for better security throughout the application as it allows for more verification checks throughout the program.

2.3.5 Usability Requirement

One of the goals of this application is to be as user friendly as possible and one of the steps taken to achieve this is creating a simple GUI. The GUI has been kept as simple as possible causing the user as minimal effort as possible. This allows the user to devote less time figuring out how to use the product and more time focused on the vulnerabilities of their system.

2.3.6 Data Requirements

As expressed in the security requirements all input will be validated as a security measure for the application. It also alerts the user to any mistakes they may have made. Data is stored in the MySQL database, so the validation helps in ensuring the data is of a correct

format. User data is encrypted when transferring between the database and application to ensure this data is kept private.

3.4.1 Environmental Requirements

The VScan Desktop application runs on a user's machine which will require an internet connection to query the database. The user is also required to install the Java Runtime Environment (JRE)¹³. The application will be compatible with windows 7,8 and 10. To develop the application I required MySQL workbench, Amazon web services and NetBeans. The VScan API runs on the Amazon Web Services, along with the MYSQL database.

3.4.2 Interface Requirements Specifications For VScan Application

The image shows a window titled "VScan" with two main sections: "Login" and "Register".

- Login Section:** Contains two input fields. The first is labeled "Email" and the second is labeled "Password".
- Register Section:** Contains three input fields. The first is labeled "Name", the second is labeled "Email", and the third is labeled "Password".

Figure 13 Login Mock-up

The above Mockup displays the login screen which a user will see as soon as they open the application. This screen will allow the user to login or register.

¹³ <https://java.com/en/download/>

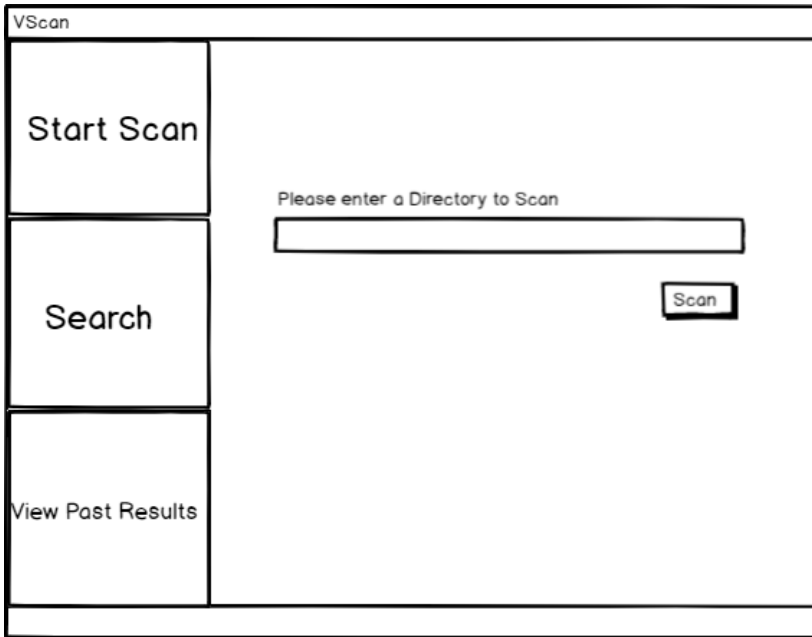


Figure 14 Scan Mock-Up

The above mock-up shows the Scan selection which a user will use to search a directory and provide all the vulnerabilities associated with that application. The search section will look the same except the user will input an application name instead of a directory.

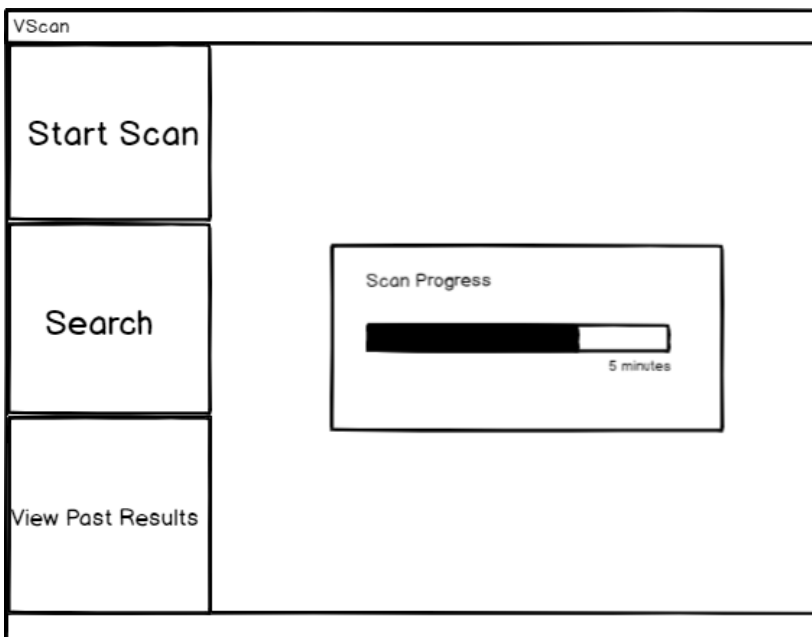


Figure 15 Progress Mock-up

The above mock-up displays an example of the loading bar which will be displayed whilst scanning and searching files and retrieving data from the database.

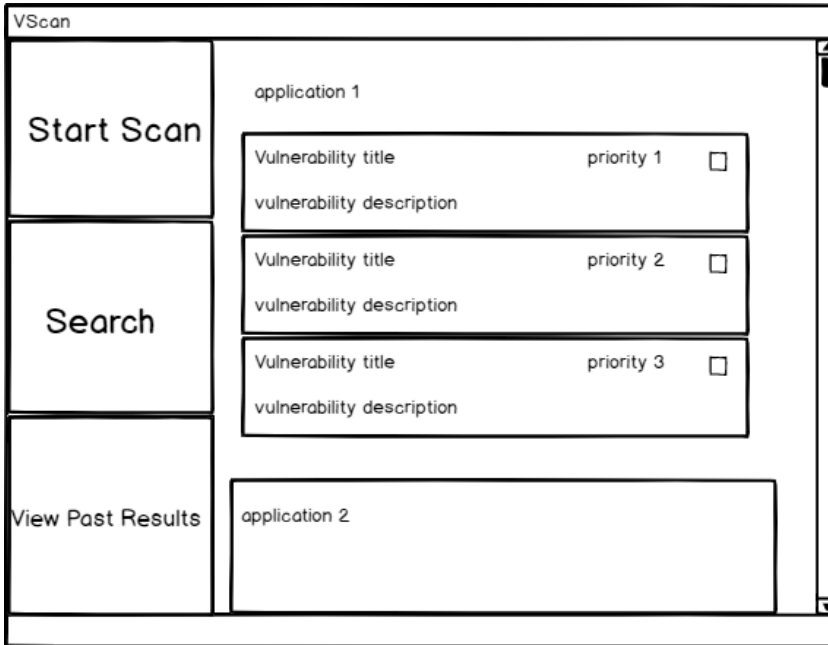


Figure 16 Display Mock-Up

The above mock-up shows the result of a search or scan which shows all the users vulnerabilities in a prioritized view. This screen will contain checkboxes which will allow the user to save or delete the vulnerability. The view past results screen will look the same as the above and shall allow the user to delete the saved result.

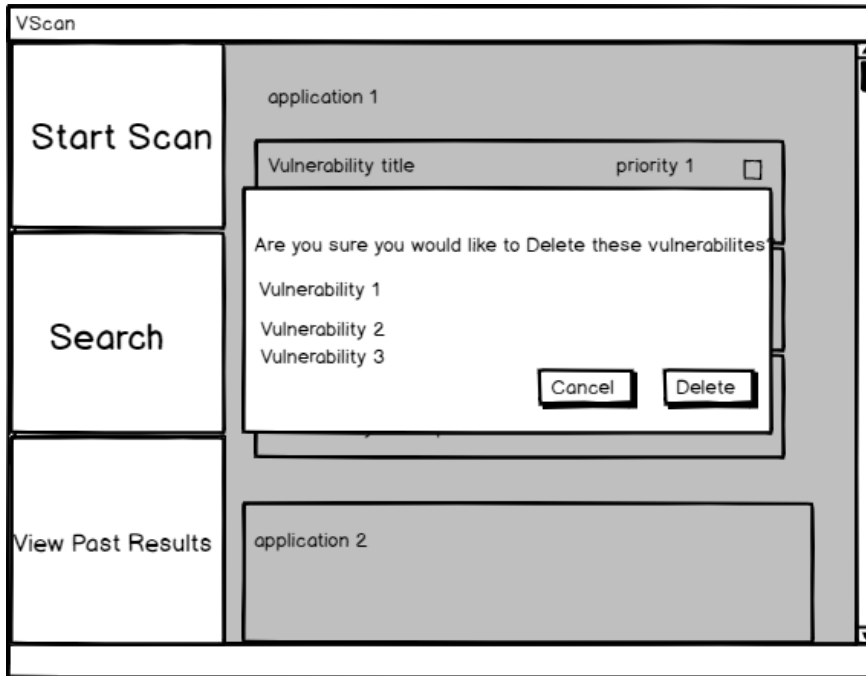


Figure 17 Delete Mockup

The above mock-up shows an example of when the user clicks the delete button after checking a number of the vulnerabilities they wish to delete from the saved list.

3.5 System Architecture and Design

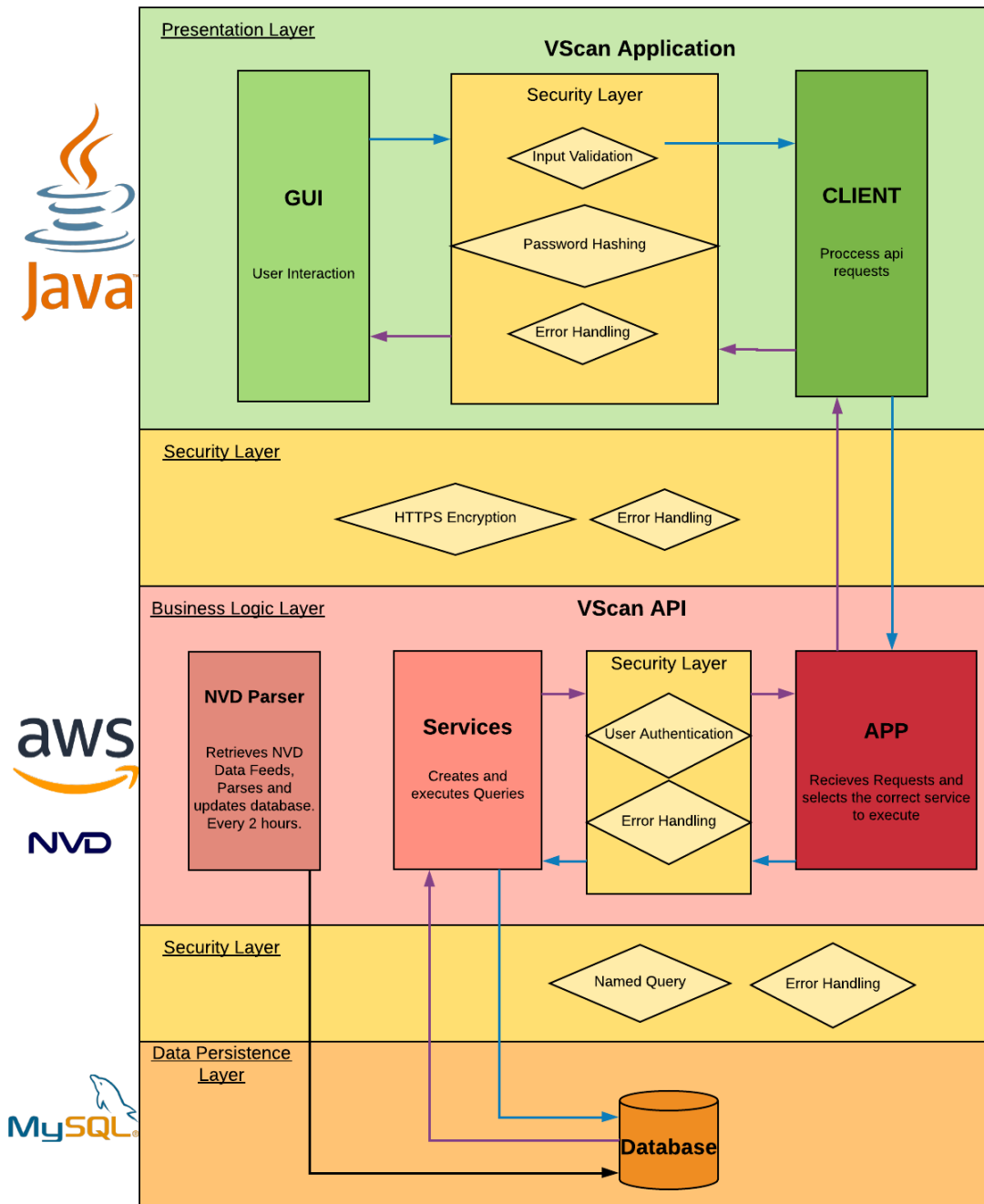


Figure 18 System Architecture

As you can see from the above illustration the overall project has been kept to a three-tiered architecture involving the presentation, business logic and data persistence layers. These layers are explained in detail below with reference to the above figure.

3.5.1 Presentation Layer

The VScan Application has been portrayed as the presentation layer of the overall project. This is due to the fact that this application simply takes user input(**GUI**), creates a request and sends this request to the API which handles all the major functionality of the Project(**CLIENT**). The VScan App then receives the response and displays the output to the user.

3.5.2 Business Logic Layer

The VScan API has been portrayed as the business logic layer of the overall project. The API receives requests from the VScan App(**APP**) and selects the appropriate method to perform the correct action (**SERVICES**), then returns a response to the VScan App(**APP**). This section of the System also is responsible for downloading the NVD data feeds every two hours, parsing these feeds and uploading the resulting data to the database (**NVD PARSER**). This is the section of the project which performs the brunt of the work.

3.5.3 Data Persistence Layer

The data persistence layer is represented by the MYSQL **DATABASE** which is used to store all information and data. The database stores all the user information and all the data parsed from the data feeds.

4 Implementation and Testing of a Hybrid of Cloud-based and Desktop Vulnerability Security Checker Application

4.1 Introduction

This chapter of the report shall describe in detail the implementation of the components mentioned in the contribution chapter of this report (Chapter 1.5). The Order in which these components were implemented is the order in which they are provided in the following chapter. The Security features which have been implemented int this application can also This introduction provides a detailed explanation of the project structure and a background explanation of the Java Persistence API Entities used by the application, as this is an important concept throughout the implementation of VScan and shall be referenced in both the implementation of components and implementation of Security features.

4.1.1 JPA Entities

JPA Entities are used throughout the project to store, transmit and display information. The below Entity is a condensed example of the Cve entity. Cve stands for Common Vulnerabilities and Exposures, this is a standard way of referencing “vulnerabilities”.

```

1. @Entity
2. @XmlElement
3. public class Cve implements Serializable {
4.
5.     @Id
6.     @GeneratedValue(strategy = GenerationType.AUTO)
7.     private int id;
8.     private String cveId;
9.     .....
10.    @OneToMany
11.    private Collection<Product> product = new ArrayList<Product>();
12.
13.    .....
14.
15.    public Collection<Product> getProduct() {
16.        return product;
17.    }
18.
19.    public void setProduct(Collection<Product> product){
20.        this.product = product;
21.    }
22.
23.
24.    public String getCveId(){
25.        return cveId;
26.    }
27.
28.    public void setCveId(String cveId){
29.        this.cveId=cveId;
30.    }

```

Figure 19 Cve Entity

As you can see the above Entity is declared using the @Entity annotation. This Entity contains different pieces of information such as the id of the Cve, and the cveId(String name) it also contains a @OneToMany relationship. This means the CveEntity contains a reference to another Collection of Entities in this case a collection of Product entities.

Entities are used to directly access, create, modify and delete objects from the database. Database tables are created in direct mapping of the Entity class, meaning any attributes created in the entity class i.e.. Id, CveId shall be mapped to a column in the database. Any relationships ie @OneToMany collection of products shall be mapped to a separate table containing the id of the Cve and the id of the related Product. When you retrieve the Cve Entity, the corresponding list of products are retrieved with it by looking at the reference table e.g. Cve_Product, this process is simplified and illustrated below.

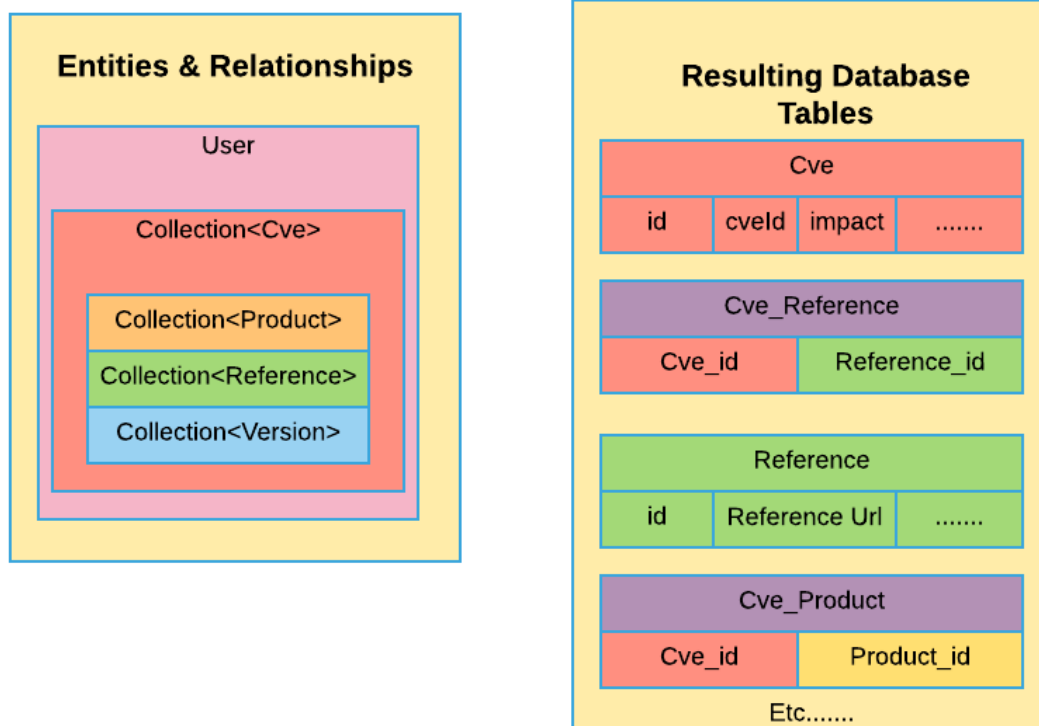


Figure 20 Entity Relationships

4.2 Implementation of the Components of VSCAN Application

4.2.1 Implementation and Testing of RESTful API Endpoints

The implementation of the VScan project began with the creation of the API Endpoints. The API endpoints are in the App.java class of the VScan API. These endpoints are responsible for receiving requests and selecting the appropriate method from the services.java class to enact. This class processes the request and then formats the data and sends the response back to the VScan Desktop Application. Below is an example of the implementation of one of these API Endpoints, followed by a detailed list of the API endpoints implemented in the VScan API.

```

1. @POST
2. @Path("user/checktoken")
3. @Consumes(MediaType.TEXT_PLAIN)
4. public Response checkToken(@Context UriInfo info,String token) throws InterruptedException {
5.     final String email = info.getQueryParameters().getFirst("email");
6.
7.     String status = "{\"ret\":\"false\"}";
8.     if (services.checkToken(email, token)) {
9.         status = "{\"ret\":\"true\"}";
10.        return Response.status(200).entity(status).build();
11.    }
12.    return Response.status(400).entity(status).build();
13.
14. }

```

Figure 21 CheckToken endpoint

The API has an endpoint for each of the declared Requirements. The above pertains to the checking of the token sent in the forgotten email requirement. This endpoint accepts a POST request made to the correct url:

(domain:port/rest/vscan/user/checktoken)

This method then retrieves the email of the user from the query parameters, it validates the email is not false, then selects the method in the services.java called checkToken() passing the email and token to be checked. If the result is true, the API sends a confirmation response to the VScan App. Testing was done throughout the development process for each API endpoint ensuring the correct inputs led to the desired outputs. Automated testing using Junit was also incorporated as shall be described in the User-based evaluations and automated testing section of this report (Chapter 5).

Below is a list of all the endpoints in the App Class:

Path	Type	Consumes	Produces	Input	Output & Process
("user/login")	GET	NA	Plaintext	Basic Authentication Header	<ul style="list-style-type: none"> Retrieves Username and Password from Basic Auth Header Checks against Database Returns true / false

("user/create")	POST	Text_XML	Plaintext	User Entity	<ul style="list-style-type: none"> • Checks if User entity exists in database. • If not, it adds this entity to the database. • Returns true / false
("user/forgotten")	GET	PlainText	application_json	Email String	<ul style="list-style-type: none"> • Checks if Email exists in database • If true, creates a 5-digit token and adds to database • Also sends the token in an email to the user's email • Returns true/false
(user/checktoken)	POST	PlainText	PlainText	Email String	<ul style="list-style-type: none"> • Checks to see if the token is correct • Returns true/false
(user/reset)	POST	PlainText	PlainText	Email String Password String token String Salt String	<ul style="list-style-type: none"> • Checks the token, • Retrieves the user entity from the database using the email • Updates the password field • Removes the token • Returns true/false
(salt/retrieve)	GET	Plaintext	Application_json	Email String	<ul style="list-style-type: none"> • Retrieves the salt from the User found using the email string • Returns salt String
(vulnerability/search)	GET	Plaintext	Application_json	Search String	<ul style="list-style-type: none"> • Retrieves all information pertaining to the search String which contains the product name • Returns List of Cves
(user/cves)	GET	Plaintext	Application_json	Uses username from the BAH	<ul style="list-style-type: none"> • Retrieves all Cve ids referenced by User • Retrieves all Cves from Cve ids found

					<ul style="list-style-type: none"> • Returns a list of Cves
(user/deletecves)	DELETE	Plaintext	Application_Json	Username from BAH cvelid string Cvesize int	<ul style="list-style-type: none"> • Retrieves list of cvesids from username • Deletes the input cvelid from the list • Checks the delete was true by checking the size after is smaller than the input size of the list • Returns list of Cves
(user/updatecve)	POST	Plaintext	Plaintext	cvelid username from BAH	<ul style="list-style-type: none"> • Adds list Cvelid to the Usernames cvelid list in the database • Returns true/false

4.2.2 Implementation and Testing of the Services performed by the API Endpoints

As a continuation of the API endpoints section of this chapter. The Services.java class contains all the functional methods related to each API endpoint. Separating these classes allows for a more secure communication between the API and the Database. Below a code snippet from the method which searches for an application name and returns a list of vulnerabilities associated with it mapped by the (vulnerability/search) endpoint.


```

1. Query query = em.createNamedQuery("vendorSelect");
2. query.setParameter("arg1", "%" + searchSplit[0] + "%");
3. List<Cve> tempCve = query.getResultList();
4. List<Cve> cve = new ArrayList<Cve>();
5. for (Cve c : tempCve) {
6.     for (Product p : c.getProduct()) {
7.         String prodName = p.getProductName().toLowerCase();
8.         String compName = searchSplit[1].toLowerCase();
9.
10.        if (search.toLowerCase().contains("microsoft office")){
11.            try{
12.                compName = searchSplit[2];
13.
14.            }catch(ArrayIndexOutOfBoundsException a){
15.
16.            }
17.        }
18.        if (prodName.contains(compName)) {
19.            cve.add(c);
20.        }
21.    }
22. }

```

Figure 22 Search Vulnerability Service

The above services is a prime example as it makes use of both JPA Entities and Named Queries.

The method above creates a new named Query from a query which is already defined. It then passes input (application name) as the query parameter. The Query searches for any CVE(vulnerability) that contains the application name in its product list. The result is then returned as an entity of type Cve. This entity is then added to a list containing all other Cve entities which are applicable to the application name. This is then returned to the App class which formats the response to the VScan Desktop Application.

Each method in the Services class was developed and tested along with the API endpoints, as they simply provide the functionality to the request of the endpoints. Therefore the Automated tests described in the User-Based Evaluations and Automated testing section of this report are also applicable to these methods.

4.2.3 Implementation and Testing of Scanning Component for Vulnerabilities

The scanning component of the VScan application was incorporated using the Windows Registry. This component can be found under the RegistryScan.java class. The Windows Registry is a set of databases maintained by the Windows Operating System. This registry

contains system information and configurations along with user information and configurations. Using this registry, a list of installed applications can be obtained by firstly searching through the list of uninstallers for 32-bit applications under the path

“HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall”.

If applicable the mechanism also scans through the 64-bit applications by scanning the path

“HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall”. When iterating through the found keys inside of these paths, the VScan App returns the DisplayName of these applications using the below code snippet. These Display Names are then queried against the database returning a list of vulnerabilities associated with each DisplayName (Application Name).

```
1. if (tr.containsKey("DisplayName")) {  
2.     String str = (String) tr.get("DisplayName");  
3.     if (!str.contains("Update")) {  
4.         applications.add(str);  
5.     }  
6. }
```

Figure 23 Scanning Mechanism

This mechanism was tested by attempting to use the above feature on a System with very few installed applications. A list of installed applications was seen using the Control Panel of the System. The scanning mechanism of VScan accurately discovered all the listed Applications under the control panel.

4.2.4 Implementation and Testing of Automatic Updating of Vulnerabilities

The VScan API has a requirement of keeping up-to date this is achieved by using the nvpars2.java class which is responsible for downloading the NVD Data Feeds, it then parses the downloaded json file and updates the MYSQL database. This is achieved every 2 hours. In order to achieve this the VScan API incorporates multithreading to create a new thread which runs the Timing.java class from the MyServletContextListener.java class.

```

1. private final Timing Timer = new Timing();
2.     Thread timerThread = new Thread(Timer);
3.     public void contextInitialized(ServletContextEvent event) {
4.
5.         timerThread.start();
6.
7.
8.     }
9.
10.    public void contextDestroyed(ServletContextEvent event) {
11.
12.        Timer.interrupt();
13.    }

```

Figure 24 Servlet Context Listener

The above code is ran when the tomcat server is started, overriding the serverContextListener contextInitialized method. This method starts the Timer thread created from the Timing.java class.

```

1. Thread.sleep(interval);
2.     if(System.currentTimeMillis() > (time+7200000)){
3.         DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss")
4.         ;
5.         Date date = new Date();
6.         System.out.println("Performed nvdCveParser at: "+dateFormat.format(
7.             date));
8.         nv.file();
9.         time = time+7200000;

```

Figure 25 Timing class

The Timer thread creates the current time of the machine on startup then , it sleeps until the time has elapsed by 2hours. When this happens it calls the file() method in the nvparse2 class. The file() method simple downloads the up-to-date modified meta file¹⁴ which contains the SHA hash value of the actual json file. If this hash value is different to the old downloaded SHA file it then downloads the actual modified json data file¹⁵. As shown below.

¹⁴ <https://nvd.nist.gov/feeds/json/cve/1.0/nvdcve-1.0-modified.meta>

¹⁵ <https://nvd.nist.gov/feeds/json/cve/1.0/nvdcve-1.0-modified.json.zip>

```

1. public String checkSha(File file) throws IOException {
2.     String sha = "";
3.     try (FileReader fileStream = new FileReader(file);
4.         BufferedReader bufferedReader = new BufferedReader(fileStream)) {
5.         String line = null;
6.         while ((line = bufferedReader.readLine()) != null) {
7.             if (line.contains("sha256")) {
8.                 String[] split = line.split(":");
9.                 sha = split[1];
10.            }
11.        }
12.    } catch (FileNotFoundException ex) {
13.    }
14.    }
15.    return sha;
16. }

```

Figure 26 CheckSha method

The resulting json file is parsed using the `nvdDataParser()` method which parses each individual `JSONObject` `cve` and extracts the relevant data to a new `Cve` Entity which it then persists to the database.


The automatic updating feature of the `VScan` API was tested using a local database. This feature was used to download and parse the bulk data files which contained the complete list of vulnerabilities. Once these data files were successfully inserted on the local database it was clear that the parser worked. The Parser then was directed to download the modified feed, which contains a list of new / modified data to be added to the database. This again successfully updated the database with the new entries and modified the old entries. This feature was then incorporated into the final `VScan` API.

4.2.5 Implementation of Amazon Web Services.

Amazon Web Services(AWS) is used to host both the `VScan` API and `MYSQL` database. AWS provides a multitude of softwares with which to host java applications and databases. For this project an elastic beanstalk instance was used to deploy the `VScan` API and an `AWS MYSQL` instance was created to host the data. AWS provides extensive logs and health checks providing the developer with information regarding the status of

deployment. As you can see below the VScan application was deployed healthily and remains in a health state.

Overview Refresh




Health
Ok

Causes

Running Version
vscantest11

Upload and Deploy



Configuration
Tomcat 8 with Java 8 running on
64bit Amazon Linux/2.7.7
Newer version available

Change

Recent Events Show All

Time	Type	Details
2018-05-12 15:11:47 UTC+0100	INFO	Environment health has transitioned from Info to Ok. Application update completed 82 seconds ago and took 22 seconds.

Figure 27 AWS Deployment

4.3 Implementation and Testing of VSCAN Security Components and Features

4.3.1 Implementation and Testing of User Authorisation

In the below code snippet, you can see the checkAuthorization() function. This function retrieves the Basic Authorisation header from the request made to the API, the users username and password are then retrieved from the header and queried against the database. If no result is found the response is false, if a user is found the response is true confirming the login of the user.

```

1. public String checkAuthorization() {
2.     String authHeader = request.getHeader("Authorization");
3.     if (authHeader != null) {
4.         final StringTokenizer st = new StringTokenizer(authHeader);
5.         if (st.hasMoreTokens()) {
6.             final String basic = st.nextToken();
7.             if (basic.equalsIgnoreCase("Basic")) {
8.                 try {
9.                     String credentials = new String(Base64.getDecoder().decode(
10. st.nextToken()), "UTF-8");
11.                     int p = credentials.indexOf(":");
12.                     if (p != -1) {
13.                         final String username = credentials.substring(0, p).trim();
14.                         final String password = credentials.substring(p + 1).trim();
15.                         final User foundUser = services.checkUserLogin(username, password);

```

Figure 28 Authorization Code

This code was tested as it was created ensuring that only found users with the matched credentials returned a state of “true” and alternate inputs returned a state of “false”;

4.3.2 Implementation and Testing Input validation

The below code snippet shows an example of input validation in the project. In line 5 a Regex is declared which is then compared to input from the email text field. This regex only allows for the correct format of string to be accepted (ie. test@test.com). From line 10 to 20 performs a check on the inputted password string ensuring that the password contains a minimum of 6 characters including 2 digits. If both of these conditions are not met, then a generic warning shall be shown which is discussed later in this section.

```

1. try {
2.     final loginClient log = new loginClient();
3.     email = emailLogintf.getText();
4.     final String temppassword = passwordLogintf.getText();
5.     final Pattern VALIDEMAIL = Pattern.compile("^([A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,6})$", Pattern.CASE_INSENSITIVE);
6.     final Matcher matcher = VALIDEMAIL.matcher(email);
7.     final Boolean matched = matcher.find();
8.     int digitNo = 0;
9.     boolean digit = false;
10.    if (email.length() > 1 && temppassword.length() > 6) {
11.        if (temppassword.length() < 16) {
12.            for (int i = 0; i < temppassword.length(); i++) {
13.                char x = temppassword.charAt(i);
14.                if (Character.isDigit(x)) {
15.                    digitNo = digitNo + 1;
16.                    if (digitNo >= 2) {
17.                        digit = true;
18.                    }
19.                }
20.            }
21.            if (digit == true && matched == true) {
22.                // For login we send the retrieved salt to the hasher and then comp
                are the returned hashedpassword
            }
        }
    }
}

```

Figure 29 Input Validation Code

This feature of the code was tested as developed, ensuring that any exceptional characters and wrong email formats were negated by the regex.

4.3.3 Implementation and Testing JPE and NamedQueries

As discussed in the code implementation section of this report JPA Entities were used to communicate with the database along with NamedQueries. Both NamedQueries and JPA Entities nullify the effects of SQL Injection and special characters by escaping the text provided as input. This added a layer of security to the project which was important for the integrity of user privacy.

4.3.4 Implementation and Testing Https / TLS Protocol

Possibly one of the most important security features of the project, and by far the most time consuming to incorporate, is the Encryption of traffic between the VScan desktop Application and the VScan API using TLS. This protocol was implemented in the project by uploading a self-signed certificate and assigning it to the instance running on the Amazon Web Services as seen below. The port 443 is listening for any https requests.

<input type="checkbox"/>	Port	Protocol	Instance port	Instance protocol	SSL certificate	Enabled
<input type="checkbox"/>	80	HTTP	80	HTTP	--	<input type="checkbox"/> OFF
<input type="checkbox"/>	443	HTTPS	80	HTTPS	testtomcat-env.eu-west-1.elasticbeanstalk.com - 9ce5f1cc-1e85-401e-b035-69a28c211ca9	<input checked="" type="checkbox"/> ON

HTTPS typically works by obtaining a certificate from a certificate agency, which contains information verify the server, and the RSA public key of the server, so the responding client can decrypt the traffic. These certificates can cost money, so in this case it was necessary to implement a self-signed certificate using OpenSSL which allowed the developer to generate a certificate. A workaround had to be implemented for the clients in the VScan APP as the java client did not recognise the certificate agency (i.e. Myself) who issued the certificate. The code below is used to force the client to accept the certificate. This would be taken out and the certificate would be replaced with an actual one in a production environment but achieves the necessary outcome in a development environment.


```

1. public Client getUnsecureClient(String email , String password) throws Exception
2.     {
3.         SSLContext sslcontext = SSLContext.getInstance("TLS");
4.         sslcontext.init(null, new TrustManager[]{new X509TrustManager()
5.         {
6.             public void checkClientTrusted(X509Certificate[] arg0, String a
7.             rg1) throws CertificateException{}
8.             public void checkServerTrusted(X509Certificate[] arg0, String a
9.             rg1) throws CertificateException{}
10.            public X509Certificate[] getAcceptedIssuers()
11.            {
12.                return new X509Certificate[0];
13.            }
14.        }}, new java.security.SecureRandom());
15.        HostnameVerifier allowAll = new HostnameVerifier()
16.        {
17.            @Override
18.            public boolean verify(String hostname, SSLSession session) {
19.                return true;
20.            }
21.        };
22.        //this has bee
23.        n put in to force it to work
24.        return ClientBuilder.newBuilder().sslContext(sslcontext).withConfig(get
25.        ClientConfig(email,password)).hostnameVerifier(allowAll).build();

```

16

Figure 30 Unsecure Client

This feature was tested by traversing to the root project url (<https://vscan.eu-west-1.elasticbeanstalk.com/>). As you can see a warning is displayed that the certificate found is not secure, this is because it is self-signed. This also means the HTTPS port is working.

4.3.5 Implementation and Testing Password Hashing PBKDF2

All Passwords entered in the VScan Application are hashed using PBKDF2 with SHA 512. Which is a secure hashing algorithm, this is due to the fact that it is slow, which means that it takes longer to brute force passwords by using payloads containing thousands of possible passwords. The passwords are all hashed before being sent to the client which sends them along with the request. This ensures the password is kept in its original state for as little time as possible. When a user is logging in the salt which is used to hash the

¹⁶ <https://stackoverflow.com/questions/6047996/ignore-self-signed-ssl-cert-using-jersey-client>

password is retrieved then the password is re-hashed, and the resulting hash is checked against the stored one in the database. If they match, then the user is logged in.

```
1. SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA512");
2. PBEKeySpec spec = new PBEKeySpec(password, salt, 10000, 512);
3. SecretKey secretKey = skf.generateSecret( spec );
4. byte[] resultingKey = secretKey.getEncoded();
```

17

Figure 31 Password Hashing

4.3.6 Implementation and Testing Error Handling – Generic Warnings

Error handling is an important feature of any application but is especially important on the backend of this project. The VScan API has error handling through each step, this ensures that if an error is thrown the entire backend of the application does not cease to function, resulting in a denial-of-service to other users. The same can be said for the VScan APP, error handling has been implemented to stop the Client from crashing the application in the event of an unexpected result. Error handling takes place through the project in form of try-catch statements, catching unexpected throws and if and else statements which only allow the correct outputs to be displayed an example of this can be seen in the Figure below. The error handling below catches an exception if the no user is returned when retrieving the salt, or if no salt is found in the user entity.

¹⁷ <https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/>

```

1. //this retrieves the users salt
2.     public String checkUserSalt(String email) {
3.         try {
4.             try {
5.                 final Query query = em.createNamedQuery("RetrieveUserId");
6.                 query.setParameter("arg1", email);
7.
8.                 final User u = (User) query.getSingleResult();
9.
10.                final String salt = u.getSalt();
11.                return salt;
12.            } catch (NoResultException r) {
13.
14.                return null;
15.            }
16.        } catch (NullPointerException n) {
17.
18.            return null;
19.        }
20.    }

```

Figure 32 Error Handling Code

Generic Warnings

An important aspect of error handling is the information displayed to the user. It is important that enough information is returned to the user, so they understand what is needed to rectify their mistake, but not enough information that they can gain some knowledge about the backend of the system or structure of the database. The VScan APP displays generic warnings to the user in the event that an error occurs. Below is an example of this, when a user attempts to login with the wrong email/password the app tells them that an account with these credentials does not exist instead of telling the user that the email is correct, and the password is wrong. This provides a potential attacker with no information about the account provided.

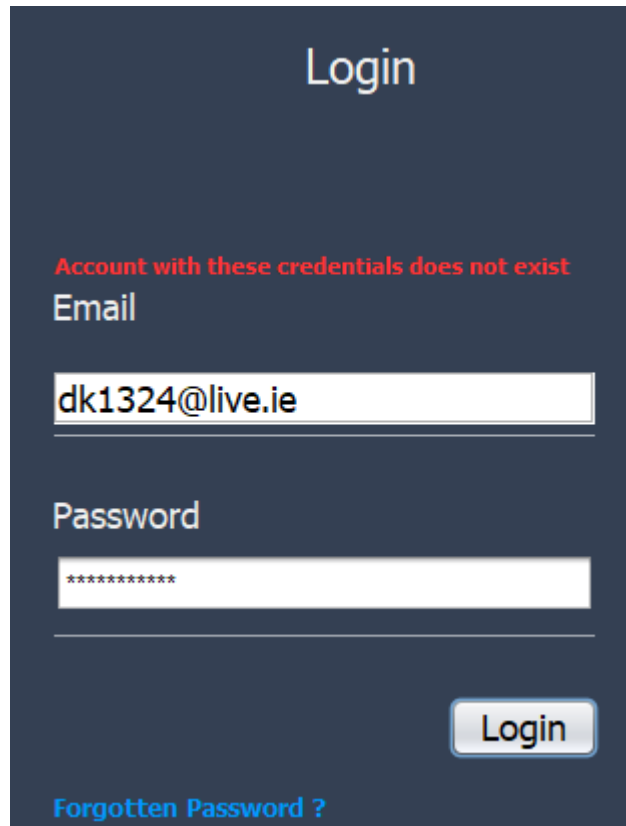


Figure 33 Login Warning

Generic Warnings can be seen throughout the VScan Application. Many of the participants of the User-based Evaluation commented on the warning messages and alerted the developer of any messages not displaying.

4.3.7 Implementation of the Separation of functional layers

It is clear from the system architectural diagram that this project is separated to from 3 layers, the presentation layer (VScan App), the business logic layer, (VScan API) and the persistence layer (MYSQL database). Within each of sections further separation has been designed, allowing for a modular project which can have easy alterations to each section. The VScan APP has been subdivided into the GUI, which handles user interaction, and the CLIENTS, which handle communication between the app and the API. This allows for further separation between the user and the backend system and its associated classes and variables. This has also been applied to the VScan API which is separated between

the APP, which handles requests, and the SERVICES, which handles formatting and creation of Queries. This also allows further separation of contact.

4.3.8 Implementation Secure coding principles

The Entire project has been developed with security in mind, this can be clearly seen from the above-mentioned features. However, all of this would be nullified without basic practices. The major one of these being the setting of attributes for methods and variables. All variables and methods that should not be accessed by outside sources are set to private, and if possible final. This can be clearly seen in the CLIENTS of the VScan API which limits the user from directly accessing the created connections between the VScan APP and API.

```
1. public class loginClient {
2.
3.     private ClientConfig clientConfig = new ClientConfig();
4.     private Client client;
5.     private Response response;
6.     private JSONObject res;
7.     private String ret;
8.     private String userId;
9.     private WebTarget target;
10.    private HttpAuthenticationFeature feature;
11.    private int port;
12.    private String getUrl;
13.    private UnsecureClient unsecure = new UnsecureClient();
```

4.3.9 Implementation and testing of the .exe Launcher

One of the final security features added to the project was the creation of a .exe launcher for the VScan App. This .exe allows for the original JAR file created in NetBeans to be wrapped and encoded. It provides the feature of denying a user from running the application without having the JRE installed. The .exe will alert the user that the application needs the JRE and then direct them to the Oracle website whereby they can obtain the JRE. This is an important feature which prompts the user the install the JRE allowing the VScan App to run on their machine.

This feature was tested by downloading the VScan Application onto a System without the JRE installed. The .exe launcher successfully notified the user, and directed them to the download page for the Java Runtime Environment.

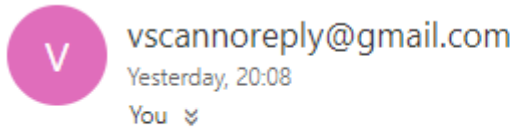
4.3.10 Implementation and Testing of the Forgotten Password Authentication

In the event that a User forgets their password, they must be able to reset the password allowing them access to their account again. This is achieved using a secondary authentication measure. An email is sent to the email account registered. This email contains a 5-digit code which the user must input to the VScan App to confirm their identity. If this code is correct then the User can reset their password.

```
1. String PossibleChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
2.     StringBuilder Token = new StringBuilder();
3.     Random rnd = new Random();
4.     while (Token.length() < 5) { // length of the random string.
5.         int index = (int) (rnd.nextFloat() * PossibleChars.length());
6.         Token.append(PossibleChars.charAt(index));
7.     }
8.     String ranToken = Token.toString();
9.
10.     sendEmail(email.toString(), ranToken);
```

Figure 34 Email Token

The above code is located in the Services.java class of the VScanAPI it selects a random string from the provided possible characters in possible chars, then adds it to the string builder until it reaches a length of 5. It then sends this to the user's email. Whereby the User is prompted to enter the code into the application to reset their password, as seen below.



Dear VScan User,

Here is your 5 digit code : 2Q79R
Please enter this code in the token box of the application to reset your password

Thank you,
The VScan Team

Figure 35 Token Email

4.4 Developed and Evaluated Graphical User Interface of the VSCAN application

4.4.1 Login Screen



Figure 36 Login Screen

The above Screenshot shows the login page whereby users can either register a new account, login to an existing account or click the forgotten password link

4.4.2 *Forgotten Password Screen*

The screenshot displays a three-step process for resetting a password on a dark blue background.
Step 1: Labeled 'Step 1', it features an 'Email' input field, a 'Submit' button, and the instruction: 'To reset your password please enter your email in the above form.'
Step 2: Labeled 'Step 2', it features a 'Token' input field, a 'Submit' button, and the instruction: 'Please enter the 5 digit code sent to your email.'
Step 3: Labeled 'Step 3', it features 'Password' and 'Re-Enter Password' input fields, a 'Submit' button, and the instruction: 'Please enter a new password'.

The above steps are displayed individually on the Forgotten password screen , first the user must input their email ,then the user must input the token sent to their email , then the user must finally input the new password for their account.

4.4.3 *Search Screen*

The screenshot shows a search interface for vulnerabilities. On the left is a sidebar with three menu items: 'Search For a Vulnerability' (highlighted in light blue), 'Scan for a Vulnerability', and 'View Saved Vulnerabilites'. The main content area has a dark blue background with the text 'Enter an applicaiton name to search for the Vulnerabilites associated with it.' Below this is a white input field and a 'Search' button.

Figure 37 Search Screen

The above is the search screen where the user enters an application name to search for vulnerabilities (Cve's) associated with the application.

4.4.4 Search Results

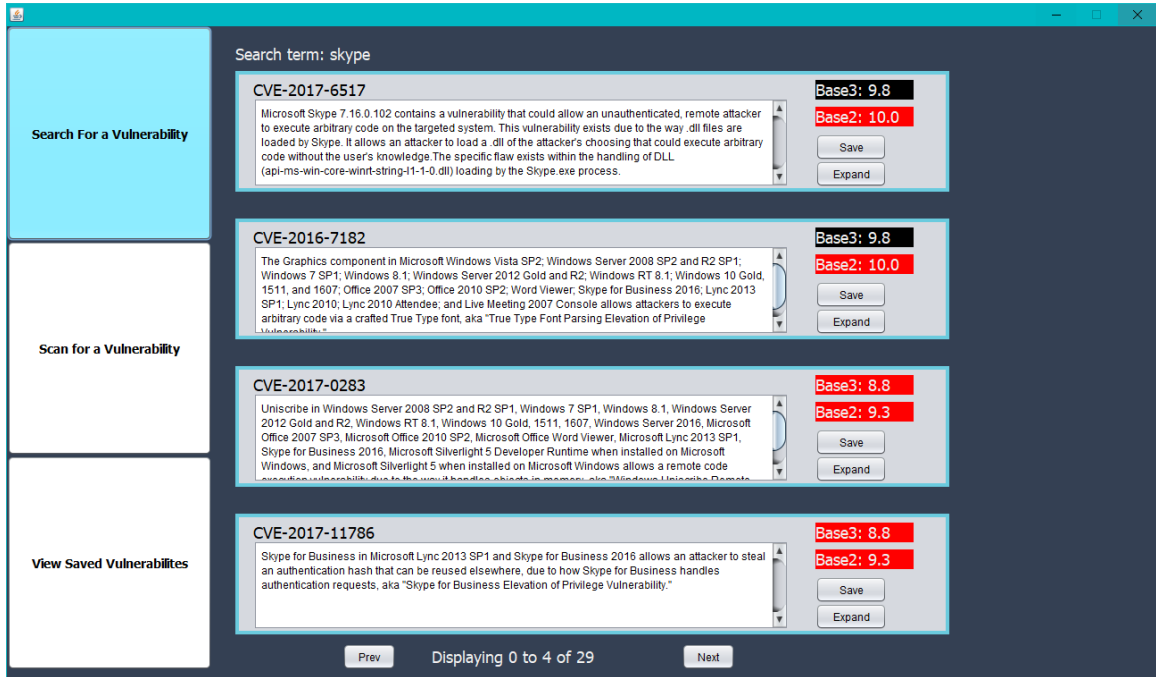
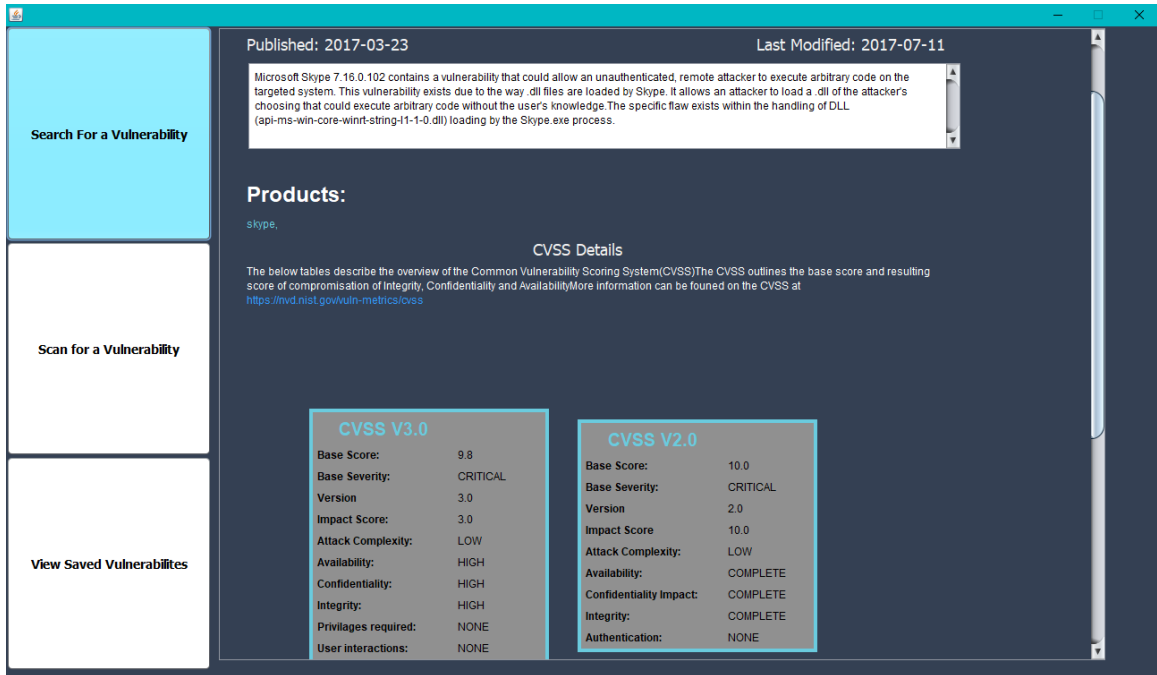


Figure 38 Search Results Screen

The above page displays the results of the search in this instance for application "skype", the resulting list is prioritized by the base3 impact score. Users can save a result to their saved list if they wish or expand to see further information.

4.4.5 Search Expand



Published: 2017-03-23 Last Modified: 2017-07-11

Microsoft Skype 7.16.0.102 contains a vulnerability that could allow an unauthenticated, remote attacker to execute arbitrary code on the targeted system. This vulnerability exists due to the way .dll files are loaded by Skype. It allows an attacker to load a .dll of the attacker's choosing that could execute arbitrary code without the user's knowledge. The specific flaw exists within the handling of DLL (api-ms-win-core-wint-string-l1-1-0.dll) loading by the Skype.exe process.

Products:
skype,

CVSS Details
The below tables describe the overview of the Common Vulnerability Scoring System (CVSS). The CVSS outlines the base score and resulting score of compromise of Integrity, Confidentiality and Availability. More information can be found on the CVSS at <https://nvd.nist.gov/vuln-metrics/cvss>

CVSS V3.0	
Base Score:	9.8
Base Severity:	CRITICAL
Version:	3.0
Impact Score:	3.0
Attack Complexity:	LOW
Availability:	HIGH
Confidentiality:	HIGH
Integrity:	HIGH
Privileges required:	NONE
User interactions:	NONE

CVSS V2.0	
Base Score:	10.0
Base Severity:	CRITICAL
Version:	2.0
Impact Score:	10.0
Attack Complexity:	LOW
Availability:	COMPLETE
Confidentiality Impact:	COMPLETE
Integrity:	COMPLETE
Authentication:	NONE

Figure 39 search Expand Page

The above page is a result of the user clicking the expand button this displays the products effected, the published data and the last modified date. It also displays a blurb with a clickable link explaining what the below tables mean. Both tables explain how critical the resulting vulnerabilities is in terms of its attack complexity etc. As the user scrolls down they will also be greeted with a list of references if they wish to perform more research on the topic as seen below.

References:

Below is a list of references associated with the above vulnerability. These references are provided for your convenience. VScan does not necessarily support the views expressed in these references.

<http://packetstormsecurity.com/files/141650/Sk...>

<http://seclists.org/fulldisclosure/2017/Mar/44>

<http://www.securityfocus.com/bid/96969>

<http://www.securitytracker.com/id/1038209>

<https://technet.microsoft.com/security/cc308575...>

https://twitter.com/tiger_tigerboy/status/755332...

<https://twitter.com/vysecurity/status/845013670...>

Figure 40 References Page

4.4.6 Perform Scan Page

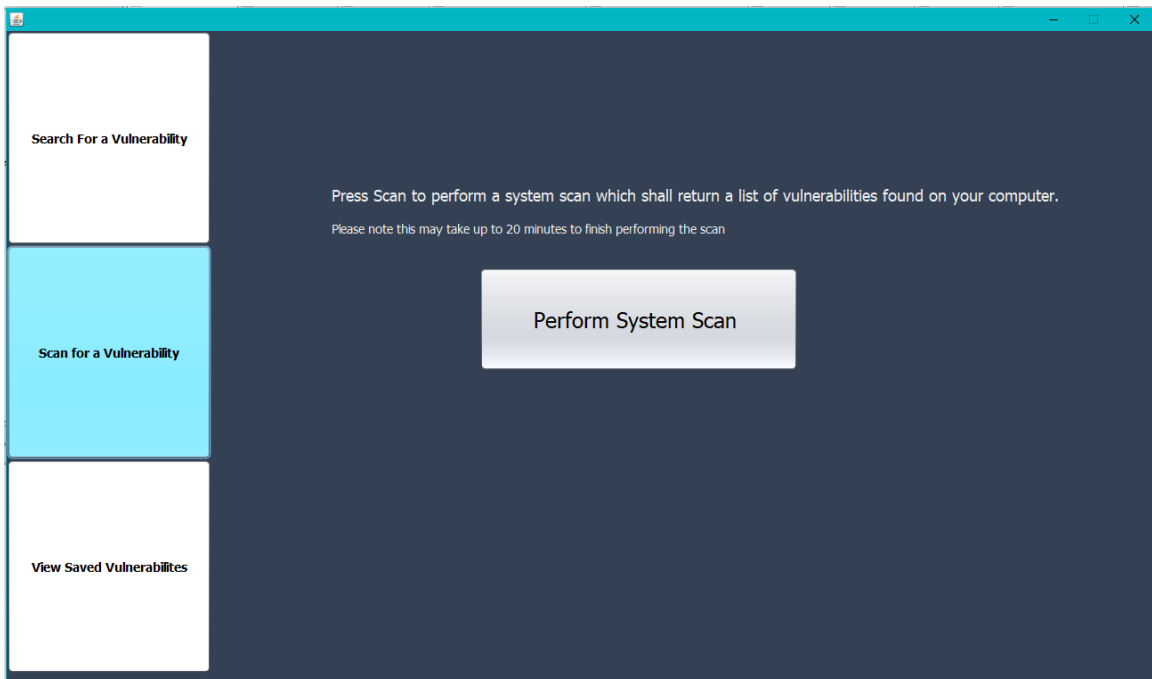


Figure 41 Perform Scan Page

The above is the perform scan page where a user can click the perform system scan to scan the windows registry for installed applications. Whilst the scan is performing the below progress bar is displayed.

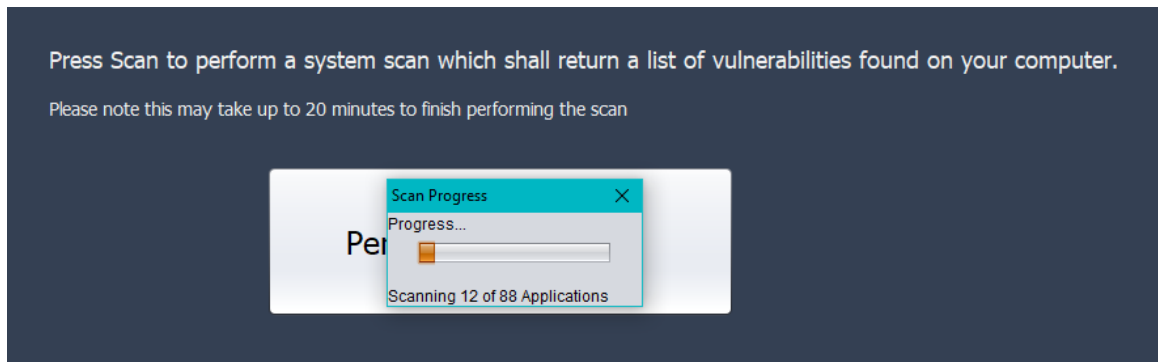


Figure 42 Progress Bar

4.4.7 Scan Results Page

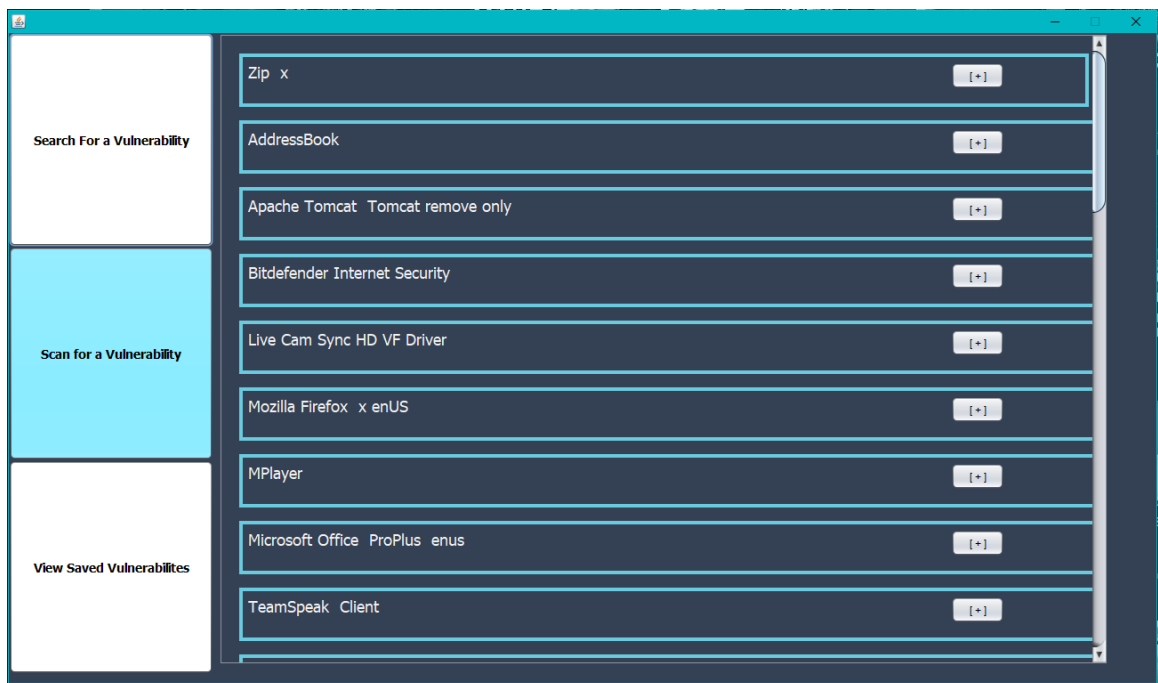


Figure 43 Scan Results Page

The above page displays all the applications found on the users windows registry. When the expand button is clicked the same page shown in the search results section is shown.

4.4.8 Saved Page

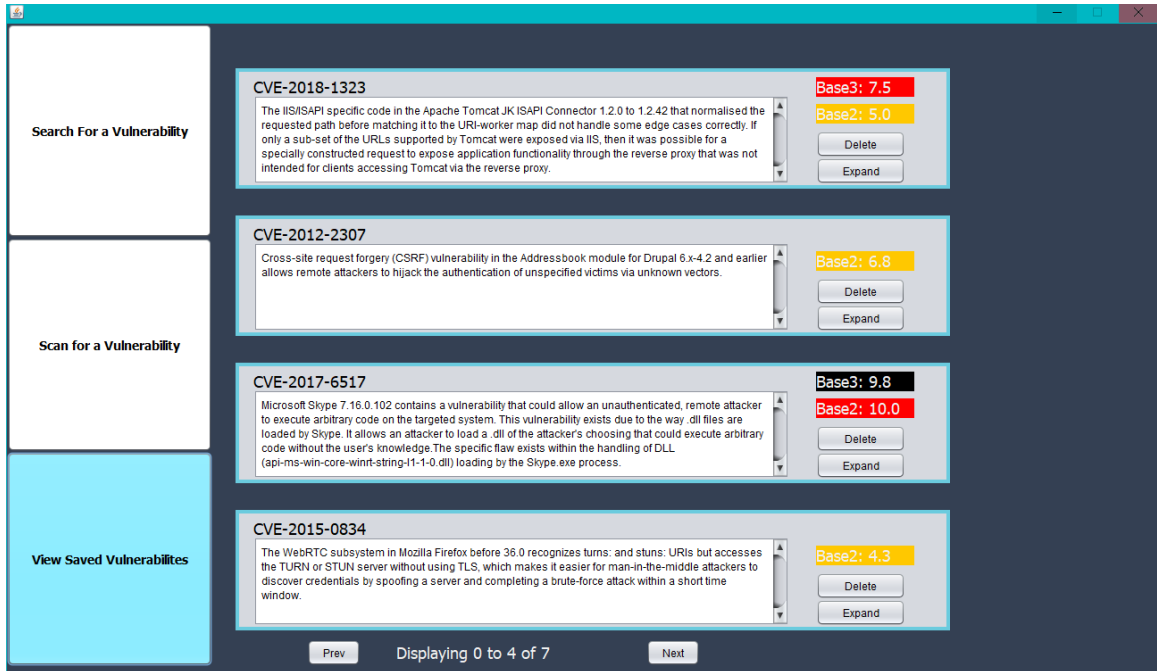


Figure 44 Saved Page

The above page is displayed when the user clicks on the saved vulnerabilities, this page is similar to the Search results page except it allows a user to delete the saved result instead of saving it again.

5 User-Based Evaluations and Automated Testing of the Developed VSCAN Application

5.1 Introduction

The following chapter presents the results of testing applied to the VScan Application. The first approach involved the testing of API Endpoints with Junit test. This was followed at the end of the project development with Structured user Questionnaires relating to the design and functionality of the VScan Project. The Resulting User-Based evaluations were positive and highlighted final bugs/ tweaks that needed to be made to the VScan Project.

5.2 Automated System Testing

Junit test were incorporated on the VScan API to test the server interactions with the database. These tests can be found under the test packages class in the VScan API application. The tests are done in the same order of steps a user would perform the actions i.e. first creation of a user, then login, then searching a vulnerability, then saving the vulnerability, then deleting the saved vulnerability etc. Below are some examples of these Junit tests

```
1. @Test
2.     public void test1UserCreation() {
3.         User user = new User();
4.         user.setEmail(EMAIL);
5.         user.setPassword(PASSWORD);
6.         user.setSalt(SALT);
7.         try {
8.             Response response = app.createUser(user);
9.             String res = (String) response.getEntity();
10.            System.out.println(res);
11.            assertEquals(res, "{\"ret\":\"false\"}");
12.        } catch (InterruptedException e) {
13.
14.        }
15.    }
```

Figure 45 Test User Creation

```

1. @Test
2.     public void test2UserLogin(){
3.         User user = services.checkUserLogin(EMAIL, PASSWORD);
4.         int userid = user.getUserId();
5.         try{
6.             USERID = user.getUserId();
7.         }catch(Exception e){
8.
9.         }
10.        assertTrue(USERID!=0);
11.    }

```

Figure 46 test User Login

```

1. @Test
2.     public void test5AddUserCve(){
3.         Boolean bool = services.updateCveList("CVE-2018-0001", USERID);
4.         assertTrue(bool);
5.     }

```

Figure 47 Test Add User Cve

5.3 User-Based Evaluation

Customer testing was performed at the end of the project creation. This testing was performed by asking participants to test the overall functionality of the application. Along with this the participants were asked to fill out a survey of the application¹⁸.

5.3.1 Testing functionality

After testing the functionality of the application many of the users discussed some of the problems, bugs they encountered when using the application, below is a list of some of the helpful suggestions and small bugs found by the users.

Colour Scheme

The original colour scheme for VScan was black and green, many of the users were happy with how the application looked using this colour scheme.

¹⁸ <https://docs.google.com/forms/d/1cq4NLeEkwqSQOu1eZojrmlpjwhKSQTOx-iNCaClug/edit>

Was the application graphically appealing

8 responses

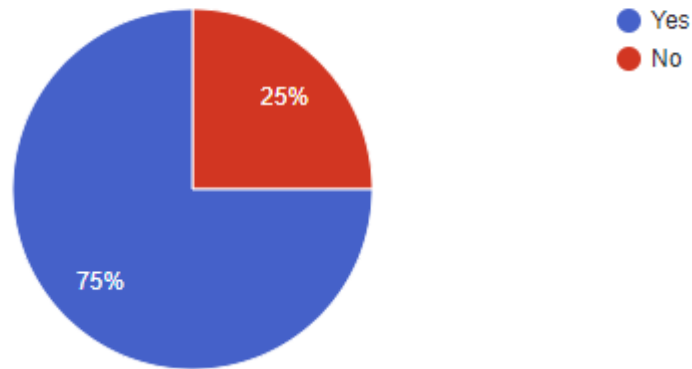


Figure 48 Q1 Survey

However, a few participants commented on how the green contrasted with the black was very harsh and made for some difficult reading. The colour scheme was then changed to the blue seen above.

Warning messages

One of the warning messages for the forgotten password screen was not displaying if the token was incorrect. This was a simple fix, by setting the warning messages visibility to true in an if statement.

Broken Client

Whilst performing the testing one of the first participants stumbled across a broken client, when they attempted to submit their email for a password reset the entire application crashed. This was due to the fact that there was both a POST and a GET request declared in the client, which were not meant to be there. This was fixed by simply removing the wrong request.

5.3.2 Survey Results

I performed the survey using google forms which allows for easy creation of sharing of the survey between participants, below are some of the other questions along with their responses and appropriate graphs

Please Rate the user experience of this Application

8 responses

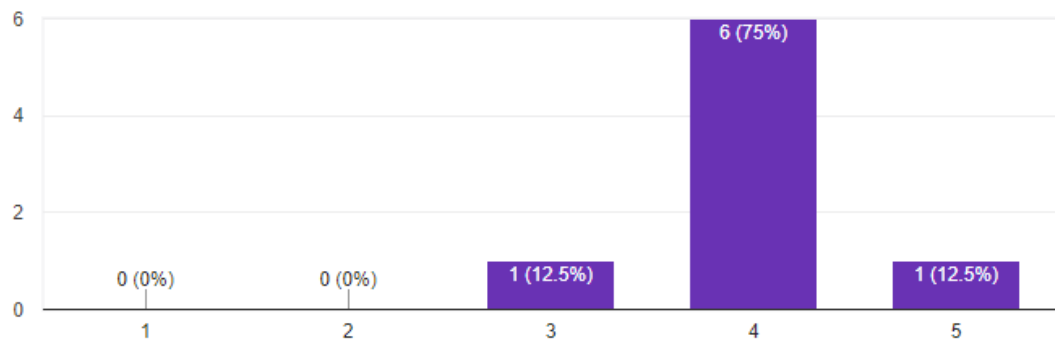


Figure 49 User Experience

When asked how the overall user experience felt all the users were quite happy with the application. One user did mention again about the harsh color scheme which has since been rectified. Another user who was less technically inclined commented on how easy it was to perform all the tasks. The GUI is simplistic in nature, so this was a great outcome.

Would you use this application in the future

7 responses

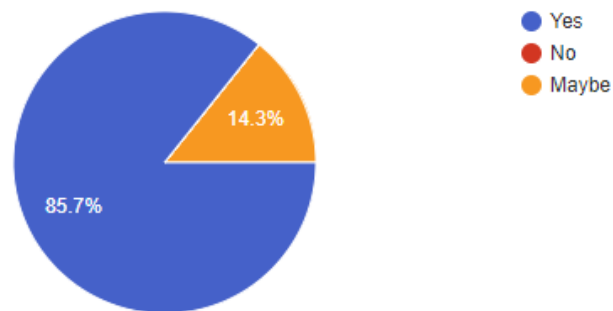


Figure 50 Future use

A majority of the users said they would be happy to use the application again. One of which was extremely interested in the concept as they are in the field of cyber security. Users frequency of use also suggests that this application could be used on a semi-regular frequency, as seen below:

Rate the frequency you could foresee yourself using this application

8 responses

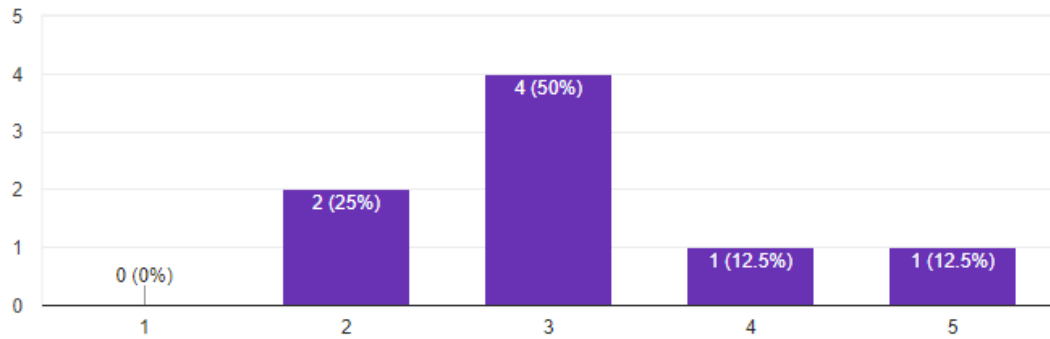


Figure 51 Frequency Survey

Check any of the following functionality you would like to see in the future

8 responses

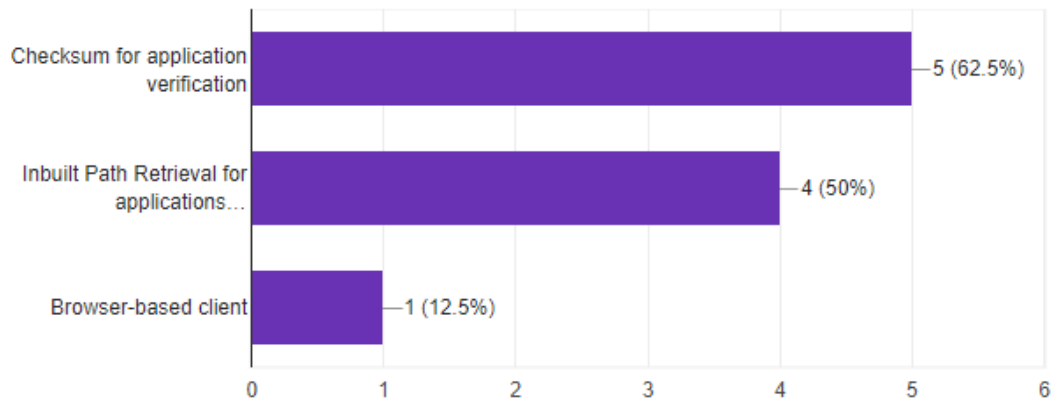


Figure 52 Future Functionality

The last question of the survey was put in to see what functionality users may wish to see in the future. The highest voted was for a checksum which would verify the integrity of the application files located on the system. The next voted was a simple inbuild path retrieval for applications so the users could see themselves what folder the applications were installed under. The feature with the least votes was the browser-based client which users all agreed they would rather use a desktop client. The main argument here when the users were asked is that they would much rather have a security style application installed on their machine and it would make them more inclined to use it more frequently.

6 Challenges Encountered and Complexity of the Project

The development of VScan presented some challenges along the way, below are some of these challenges and how they were faced

Design Challenge:

VScan was designed using java swing which originally presented itself as the opportune language to code in, but as development carried on I found myself struggling to create a graphically appealing GUI. The constraints of swing and my own inexperience with developing using swing led to a difficult creation of the design. However, after some time getting use to the environment I am extremely satisfied with the resulting application.

Scanning the System Challenge:

Originally the VScan App was intended to scan a system directory for .exe files which were then checked against the database. However, after developing this feature, It quickly became apparent that it would not be a suitable solution. Applications can contain multiple .exe files which ended up resulting in 1000 + results being queried, this created a huge bottleneck in the application. After studying the windows registry as a digital forensics tool in our digital forensics module, it was decided that I would attempt to scan the registry for a list of applications instead. This was resulted in a very simple method which can access the registry and return the list of applications

Updating the Database Challenge:

Downloading the feeds from NVD was a simple step to incorporate in the project but getting this action to occur alongside the running of the VScan API was a difficult process. Multithreading was the final solution which was used in the project which was a giant learning experience as I had no prior knowledge of multi-threading.

HTTPS Encryption Challenge:

HTTPS Encryption was the final hurdle when developing this project. Luckily Amazon Web Services allows for very simple activation of HTTPS ports along with Self-Signed-Certificates. Creating a java client which accepted the self-signed certificate was a difficult

task. There is very little official literature published by the jersey/tomcat community on enabling https and using jaxrs clients to consume the https services. This was the most time consuming single feature to employ in the project.

7 Conclusion and Recommended Further Development

Overall the development of VScan has been a tremendous success. The idea for this project was conceptualised in the summer of 2016 and since then it has been a tremendously informative and educational experience in developing this project. The approach used to develop the project I believe to be a great success, it allowed for improvements or changes to be made to individual components without effect the complete project. This modular design is something that would be beneficial in future project. Some of the Techniques may be changed if this project was to be developed in the future. The use of a different language would be used for the GUI simply because of the challenge it presented when designing.

Overall, I Believe the application was a success and the goals set out in the beginning have been achieved. Customer feedback and testing was positive with interest as to whether I would be further developing the project which is intriguing.

After reviewing customer feedback and multiple brainstorm below are some of the functionalities that would be beneficial to the overall functionality of the application and allow a user to obtain better security of their overall system.

7.1 Integrity checker using checksum

I would like to create an integrity check for the applications. Using information about the application such as its intended number of files and the actual details of the file. For this it may be possible to use a checksum check. A check sum is a data value that is associated when installing a program which will allow you to know if the program has been installed intact or not. Using this we may be able to see if the application has malware or is missing some of the files it needs.

7.2 Scan Folder Directories

VScan would benefit from the introduction of a folder examiner, which would allow users to scan individual directories/ folders on the user's computer. This would allow them to target certain groups of applications which they are using. It may be useful for companies who store applications across different directories when installing company wide.

References

Amazon Web Services, Inc. (n.d.). Amazon Web Services (AWS) - Cloud Computing Services. [online] Available at: <https://aws.amazon.com/> [Accessed 12 May 2018].

Baeldung. (2018). Zipping and Unzipping in Java | Baeldung. [online] Available at: <http://www.baeldung.com/java-compress-and-uncompress> [Accessed 12 May 2018].

Beach, J. (n.d.). PlanetB | Syntax Highlight Code in Word Documents. [online] Planetb.ca. Available at: <http://www.planetb.ca/syntax-highlight-word> [Accessed 12 May 2018].

Dev.mysql.com. (n.d.). MySQL :: Download MySQL Workbench. [online] Available at: <https://dev.mysql.com/downloads/workbench/> [Accessed 12 May 2018].

Dixon, J. (2016). Installing a self-signed SSL certificate on Elastic Beanstalk | Joe Dixon. [online] Joedixon.co.uk. Available at: <https://joedixon.co.uk/installing-a-self-signed-ssl-certificate-on-elastic-beanstalk> [Accessed 12 May 2018].

Fisher, T. (2018). What Is the Windows Registry & What's It Used For?. [online] Lifewire. Available at: <https://www.lifewire.com/windows-registry-2625992> [Accessed 12 May 2018].

Kowal, G. (n.d.). Launch4j - Cross-platform Java executable wrapper. [online] Launch4j.sourceforge.net. Available at: <http://launch4j.sourceforge.net/> [Accessed 12 May 2018].

Kuhn, R., Raunak, M. and Kacker, R. (2017). An Analysis of Vulnerability Trends, 2008-2016 - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8004385/authors> [Accessed 12 May 2018].

McMillan, R. (2003). A RESTful approach to Web services. [online] Network World. Available at: <https://www.networkworld.com/article/2339954/software/a-restful-approach-to-web-services.html> [Accessed 12 May 2018].

Netbeans.org. (n.d.). [online] Available at: <https://netbeans.org/downloads/> [Accessed 12 May 2018].

Nvd.nist.gov. (2018). NVD - Data Feeds. [online] Available at: https://nvd.nist.gov/vuln/data-feeds#JSON_FEED [Accessed 12 May 2018].

Oracle.com. (n.d.). Java SE - Downloads | Oracle Technology Network | Oracle. [online] Available at: <http://www.oracle.com/technetwork/java/javase/downloads/index.html> [Accessed 12 May 2018].

Owasp.org. (n.d.). Category:OWASP Top Ten Project - OWASP. [online] Available at: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project [Accessed 12 May 2018].

Project, A. (n.d.). Apache Tomcat® - Welcome!. [online] Tomcat.apache.org. Available at: <http://tomcat.apache.org/> [Accessed 12 May 2018].

Stack Overflow. (2017). Ignore self-signed ssl cert using Jersey Client. [online] Available at: <https://stackoverflow.com/questions/6047996/ignore-self-signed-ssl-cert-using-jersey-client> [Accessed 12 May 2018].

Geol, J. and Mehtre, B. (2015). Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. Procedia Computer Science, [online] 57, pp.710-715. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050915019870?via%3Dihub> [Accessed 12 May 2018].

Gupta, L. (2013). Generate Secure Password Hash : MD5, SHA, PBKDF2, BCrypt Examples - HowToDoInJava. [online] HowToDoInJava. Available at: <https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/> [Accessed 12 May 2018].

Metasploit. (n.d.). Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit. [online] Available at: <https://www.metasploit.com/> [Accessed 12 May 2018].

Newson, A. (2005). Network threats and vulnerability scanners. *Network Security*, [online] 2005(12), pp.13-15. Available at: <https://www.sciencedirect.com/science/article/pii/S1353485805703147??> [Accessed 12 May 2018].

Appendix

7.3 *Project Proposal*

Project Proposal

Client based application Vulnerability Security Checker

Daniel Kelly

X14532897

X14532897@student.ncirl.ie

BSc (Hons) Computing

Specialisation Cyber Security

25/10/2017

7.3.1 Objectives

The objective of this project is to create a Java application that can allow a user to scan a directory in their computer, return the application that the scan finds and check this against a database to find any vulnerabilities associated with that application. The java application will allow a user to register and login, allowing the user to perform the scan. The user will be able to review any past vulnerabilities they have found in order to verify if they are still a threat on their machine. The application will provide a prioritized view of vulnerabilities for each application in order of serious to trivial allowing the user to see what they are at risk of.

I will first have to create a backend for my java application using MySQL this will allow me to store details on applications and the vulnerabilities associated with them. This database will have steps to prevent SQL injection.

I will need to create a RESTFUL API which will run every two hours and download the JSON data feed from (nvd.nist.gov.). This will allow me to populate my database with up to date and relevant vulnerabilities.

The Java application itself will be coded using secure coding practices in order to minimize its own vulnerabilities. It will communicate with the database in an encoded manner, allowing the applications on the user's computer to remain unknown and thus non-targets.

7.3.2 Background

I came up with the idea for this project whilst on my internship last semester. Myself and my then manager were discussing the topic of my 4th year project when he broached the idea of a vulnerability scanner. The reason we thought of the idea is because as software developers we were using many programs and applications to complete our tasks. Some of the applications we used, were out of the current patch. Because an older version of the application was more compatible with other applications we were using, whether that

be the companies own application, an open-source or a paid for application. Whilst it may have been necessary this is a dangerous thing to do. By using applications without knowing their associated vulnerabilities, we are leaving our computers open to a multitude of risks and by extension the network of the company.

The idea particularly caught my attention because this was around the time of the ransomware attacks. Which sent a lot of businesses scrambling to make sure their security was up to standard.

After gaining a few leads to research from my manager I discovered that there were already vulnerability scanners on the market. After researching further, I found that most of these scanners were marketed for big companies allowing them to scan their networks for network weak points. The other kind of scanner I came across were scanners which would assess the vulnerabilities of web applications. I believed there was a market for a scanner that could scan applications which were on the client's machine and provide information on an individual level.

Upon researching the topic, I decided I needed to find a reliable source of vulnerabilities. I decided to use The National Vulnerability Database which is a repository maintained by the U.S Government. This organization provides a data feed which is updated every two hours free for use. This will allow me to have a constant and up-to-date source for the vulnerability data.

7.3.3 Technical Approach

In order to develop this project, I will need to first research the requirements for the project. I have learned in past projects that setting clearly defined goals for the short and long-term are incredibly important for having a successful finished product. So before setting out to develop the product I will need to get the documentation for the project finished. This includes the project proposal, research requirements and specifications and the project plan. I will then begin creating my prototype. Which can be used for my mid-point presentation and further developed into my final product.

For documentation and research, I will predominantly research how I can most effectively use the eclipse IDE for my development of the Java application. I will also be taking a course in MYSQL on udemy.com, this will act as a refresher for SQL in general and allow me to refine my skills for use in the creation of my database. I have also found some informative articles written by Martin Fowler that discuss the iterative development methodology, Martin is one of the leading talk givers on the agile process and other methodologies to use when developing, so reading his articles will allow more of an understanding of the iterative approach.

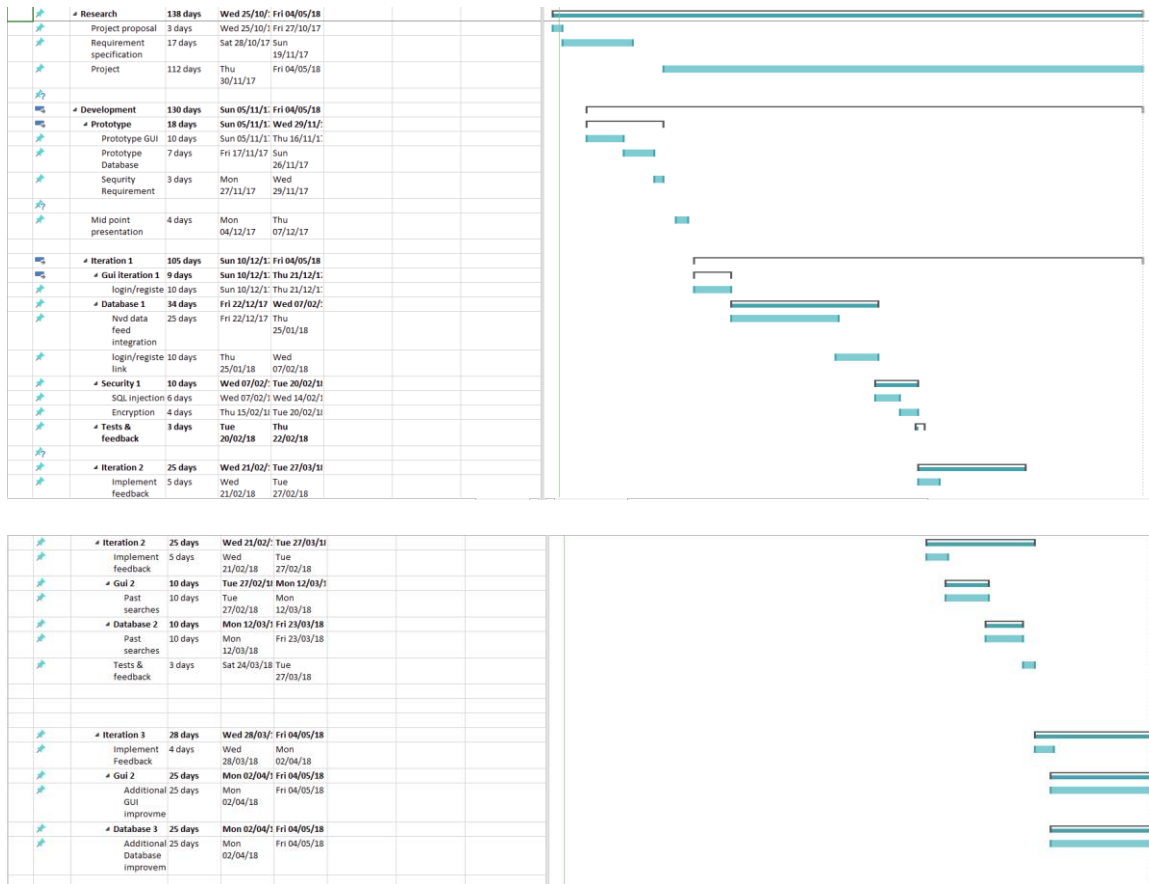
I have decided to go with an iterative methodology when developing the product. I think this will be the best route to take when developing as it will allow me to create some of the application, then obtain feedback (through my supervisor and possible testers) and then apply this feedback in the next iteration of the development. I believe this approach will allow me to benefit from the use of a prototype the most as I will be able to create a base application with the intent to learn from the feedback at the mid-point presentation. I can then continue using this approach next semester with my supervisor.

The application will be created using Java and I will be using MongoDB as my backend for the server. MongoDB is a collection based database which will be perfect for storing the data feeds which are provided in JSON or XML format from NVD. These data feeds will be fetched by having a batch file run every 2 hours to download the files. I need not download the whole database every time as I can download meta files which will just give the most recent changes to any entry's already existing in the database.

6.3.4 Special resources required

The main special resource required will be the database from NVD. This database will be downloaded using the RESTFUL API, parsed and formatted. It will then be used to update my own database with the information I have selected to take from it.

6.3.5 Project Plan



As you can see from the project plan above I have taken the iterative development methodology into mind. I have also kept some time in my plan for me to incorporate security features in the project. I expect that I will have 3 iterations of development, but I kept the last cycle long so that if needed I can add in a fourth.

6.3.6 Evaluation

I will be attempting to incorporate Junit tests as I develop the application. I plan on using some of the techniques I learned whilst on my internship last semester. I found that incorporating the tests as I move along with the project allows for a bug free application which will cause less stress as I come closer to deadlines. I will also be asking my supervisor and fellow classmates to test the application after each iteration this will allow

me to fix/add any corrections my focus group comments on at the beginning of the next iteration. When my product is complete I plan on asking friends to test the application, these will include people currently working in the tech industry and outside of the tech industry, so that I have opinions from both people with a technical background and without.

7.4 Monthly Journals

September

This month I focused on preparing for the project pitch and improving on my project idea until it was at a standard that I believed presentable. I originally thought of my project idea on my work placement in late June. Myself and my supervisor in Openet were discussing my final year project ideas when he mentioned a vulnerability checker for application on a computer. We thought this could be a good project, as from working in the company for 6 months I knew that there were certain applications we ran in older patches for compatibility issues. We did this without knowing the risks these applications were open too. After a few minutes of discussing this project idea I decided to look into it and see if it was possible. I decided to create an application that would scan a directory in a computer for application/software files. It would then check these software's/applications and the patch they are using against my database. My database will contain information about applications/software's and their vulnerabilities on each patch or version. This information is taken from the National Vulnerability Database, which is an American government run organization. They provide their database in JSON or XML formats which is updated every hour.

I decided this was a good base idea for my project, but wanted to think of more possibilities to add it. I am planning on adding a login to the application which will allow the user to store previous applications they have found and what vulnerabilities they have. This will allow users to know what applications and risks they are taking on their

computer. It will also allow them to rerun the scans in the locations they have already found to check and see if their applications have updated.

In preparation for the pitch I made sure I understood the idea and I wanted to be able to add in security features, such as preventative steps against SQL injection and possibly encryption of the user details. The pitch went well with all three judges agreeing that they liked the idea. The next step now is to receive my supervisor and begin working on the project documentation. Especially the project proposal which is due soon.

October

This month was focused on preparing my requirements specification documentation and further developing my idea for the mid-point presentation. I created 7 use cases and 2 abuse cases for the project to show the main functionalities of the project. The users will be able to register, login, search for vulnerabilities by entering an application name, scan a directory and search for vulnerabilities and save and edit these results. The report also goes into detail about the security aspects of the project which is important seeing as I will be handling sensitive information about the vulnerabilities of the user's machine which we do not want an attacker getting hold of. I have begun to make mock-ups of the GUI which are helping me visualize what the prototype will look like for the mid-point presentation. I also met with my supervisor for the second time and she helped me greatly in preparing for the requirement specification sheet, mainly by helping me outline the idea of the project in a better way.

November

During November I met with Catherine again to further discuss the requirements for the technical report upload. During this time, we broached the topic of a web application. After further research I have decided to go with the desktop application. Due to web applications being sandboxed they are limited on what they can do to a User's machine. I feel as though creating a desktop application will allow me to have a wider array of possibilities for the future of the project. Catherine has been very helpful in the reviewing

of my requirement specification. I have begun building a prototype with the aim of displaying a simple GUI example of the final project. I intend on providing the examiners with a brief understanding of how the application will be laid out and what the architecture of the final project will be during the mid-term presentation.

January

During January I focused primarily on further researching the creation of the project. What I mean by this is the creation of the GUI. I was unsure whether or not I would incorporate HTML into the java application using NetBeans or stick with a Swing application. After considering both, I decided I would attempt to create the application using Swing, as in the past all my projects have consisted of HTML applications. I wanted to spread myself out of my comfort zone slightly. I began creation of the base application using the swing toolkit in NetBeans. For now, I have decided to create the GUI and the VScan API component and feature by feature. This means creating the front end on the VScan APP for the login and the backend on the VScan API at the same time allowing me to get features done as a whole each time. This month I managed to get the login/register pages done. I also achieved getting the search functionality finished also. Which will allow me to incorporate the same backend functionality when I get the scan working. I have also created the parser which will parse through the downloaded json file and extracted information to the database.

February

During February I encountered a few challenges which were necessary to overcome. The first challenge consisted of the creation of the scanning feature of the application. This would scan a directory on the user's computer and return any vulnerabilities found associated with applications in the users machine. This feature was not as difficult to create as I thought it would be however it had an undesired output. The result of the scan typically led to more than 1000 queries as each application could have multiple .exe files

which were redundant for the use I needed them. This resulted in me needing to change how the application scanned a user's machine. After studying the windows registry in digital forensics, I had the idea of scanning the registry for a list of installed applications. After some research I found that this was indeed possible and was in fact simple to implement. The next hurdle came when attempting to perform the update of the database upon start-up of the server. After some research I found that I could implement multithreading to run a separate thread on the VScan API which allowed me to run a timer. This timer checked if the system clock had progressed by 2 hours. If this condition had been met, then the parser would download the NVD data feed from the appropriate link and update the information to the database.

March

During March I encountered my last big hurdle. I still did not know how I was going to host the application and incorporate the HTTPS encryption. Luckily this was solved on the same platform. I found amazon web services and decided to try hosting the tomcat server on this. It worked wonderfully, needing no further configurations. Next, I attacked the https challenge. This initially seemed to be going well with amazon providing clear steps on how to incorporate self-signed certificates. The problem was with the clients themselves, they would not recognise the self-signed certificates as they had no agency associated with them. This led to me incorporating a workaround which could easily be removed in a development environment. I began the arduous process of writing my report during this month and finished off the creation of the report. Catherine was extremely pleased with my progress again and we decided the report was the final part of the project I needed to get right.

7.5 Other Material Used

Any other reference material used in the project for example evaluation surveys etc.

Below are links to the Survey I created and my GitHub repository

Survey:

<https://docs.google.com/forms/d/1cq4NLeLEkwqSQOu1eZojrmlpjwhKSQTOxiNCaRClug/edit>