Database Vulnerability Tester

Jakub Lus

18018563

Word Count: 9139

UXCFXK-30-3

Digital Systems Project



# Abstract

The base idea of the project is for the tool/system to perform most common penetration tests against the database to see how the database reacts and behaves after the attack has occurred and if the attack was successful or not. Principally the purpose of the attack is test database against the most common attacks that are occurring on databases. In this project I have research many aspects of databases from the concepts of them to encryption of databases. As far as the research goes, I have researched database technologies so when I was in the designing and prototyping stage, I know how to design the system as I learned that there are two kinds of database technologies; web-based and download and install database technology. When I was thinking about the methodology, I thought about programming environment and programming language so that when the implementation stage began, I knew where I must start as I chosen the python language and the default environment to use during this stage. The tool is meant to test databases against three most common database attacks which include Denial of service attack, weak authentication attack and SQL Injection attack. After the implementation was done, I performed the tests on the system next so that I know which parts of the system work, and which doesn’t. The next stage of the project was the evaluation of my progress on the project and overall, I believe that I have done with the project okay like around 3/5 as I believe I did very well in certain areas and not so well in others.

# Acknowledgements

I would like to thank everyone who has helped and/or guided me through the process of creating and progressing though the Digital System project no matter how big or small their contribution was towards the Digital Systems Project. First, I would like to Thank Dr Phil Legg who has been my mentor though out the entire process of research, design, creation and testing the Project that I have created. Also, I would like to thank other lecturers who have given me feedback when I asked them about my project idea, before I was assigned to Dr Phil Legg. Also, I would like to Thank my peer Abdihamed Mohammed who gave me feedback on my progress and what he had read in my Report of the Digital Systems Project. This is all the People that I believe helped me in my progress throughout the project and I would thank them all very much, through the challenges I had to go through in this Project and their Help which helped me to shape and structure the project professionally.

# Table of Contents

[Abstract 1](#_Toc97284906)

[Acknowledgements 2](#_Toc97284907)

[Table of Contents 3](#_Toc97284908)

[Table of Figures 5](#_Toc97284909)

[1. Introduction 6](#_Toc97284910)

[2. Literature Review 7](#_Toc97284911)

[2.1 – Databases 7](#_Toc97284912)

[2.2 – Concept of Database Security 7](#_Toc97284913)

[2.3 – Database Security Challenges 8](#_Toc97284914)

[2.4 – Penetration testing tools 8](#_Toc97284915)

[2.5 – Database Security Threats 9](#_Toc97284916)

[2.6 – Use of Encryption in Databases 10](#_Toc97284917)

[2.7 – Attacks on Databases and Prevention 10](#_Toc97284918)

[2.7 – Database Technology 11](#_Toc97284919)

[2.8 – Database Vulnerabilities 13](#_Toc97284920)

[3. Requirements 14](#_Toc97284921)

[4. Methodology 16](#_Toc97284922)

[4.1 – Language and IDE 16](#_Toc97284923)

[4.2 – Platform 16](#_Toc97284924)

[5. Design 17](#_Toc97284925)

[5.1 – High-Level Design 17](#_Toc97284926)

[5.1.1 – System Architecture 17](#_Toc97284927)

[5.1.2 – User Interface Design 18](#_Toc97284928)

[5.1.3 – Environment Design 18](#_Toc97284929)

[5.2 – Low-Level Design 19](#_Toc97284930)

[5.2.1 – Design Structure 19](#_Toc97284931)

[5.2.2 – Design Interactions 19](#_Toc97284932)

[5.3 – Design of Tests 19](#_Toc97284933)

[6. Implementation 21](#_Toc97284934)

[6.1 – Coding 21](#_Toc97284935)

[6.2 – Testing 23](#_Toc97284936)

[7. Project Evaluation 25](#_Toc97284937)

[7.1 – Evaluation of Research 25](#_Toc97284938)

[7.2 – Evaluation of Requirements 25](#_Toc97284939)

[7.3 – Evaluation of Methodology 25](#_Toc97284940)

[7.4 – Evaluation of Design 25](#_Toc97284941)

[7.5 – Evaluation of Implementation 26](#_Toc97284942)

[8. Further Work and Conclusions 27](#_Toc97284943)

[Glossary 28](#_Toc97284944)

[References / Bibliography 29](#_Toc97284945)

[Appendix A: Project Proposal 31](#_Toc97284946)

[Appendix B: Meeting with Supervisor 32](#_Toc97284947)

[Appendix C: Source Code 34](#_Toc97284948)

# Table of Figures

[**Figure 1 - Diagram of Database** 7](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206916)

[**Figure 2 - Intrusion Detection System** 7](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206917)

[**Figure 3 - CIA Diagram** 8](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206918)

[**Figure 4** - **Screenshot of SQLMAP** 9](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206919)

[**Figure 5 - Denial of Service attack Diagram** 9](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206920)

[**Figure 6 - Attacks on Database Diagram** 10](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206921)

[**Figure 7 - List of Database Vulnerabilities[16]** 13](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206922)

[**Figure 8 – System Architecture** 17](#_Toc97206923)

[**Figure 9 - System Design(Labelled) Version 1.0** 18](#_Toc97206924)

[**Figure 10 - Version 1.0 of the System** 21](https://d.docs.live.net/108ef6df5e057cd7/Desktop/Digital%20System%20Project/Final/Word%20Template.docx#_Toc97206925)

**Figure 11 - Database**……………………………………………………………………………………………………………………….21 **Figure 12 - Version 2.0 of the System**……………………………………………………………………………………………..21 **Figure 13 - Procedure of the Application**………………………………………………………………………………………..22 **Figure 14 - Final Process with Demonstration Route**………………………………………………………………………23

# 1. Introduction

The Databases are very important in nowadays and that’s why security and securing the databases is very important aspect, when designing and creating a database. Finding flaws in the databases is also quite difficult when you don’t know what you need to look for and where you could find it. Some people may not know how important security is, but the truth is that if the security has flaws then the entire Database is at risk, because there is always someone that will figure out a way into the database that has no authorization to get in.

The main reason for this project is to develop a tool that prevents malicious attackers as well as leakers from gaining access to the database without authorization of the people who oversee the database server and/or specific database file. The Flaws are the vulnerabilities that need to be removed in order to make the databases more secure and safer, so that confidential data won’t be stolen or used by the wrong people.

Databases are sets of Data stored in one place so that they can be easily accessed by the company, especially the management staff to add, alter and delete data, when needed as for example if the companies employee leave the management staff needs to delete the row in database for that employee in the company, or if the company recruits someone new, then they need to add another row with the confidential data of the employee so if anything will go wrong they know where to look for their contact details etc. Therefore, the project that I’m doing is very important as if someone unauthorized will gain access to the database then bad things could happen if the attackers steal confidential data like home address, bank details etc.

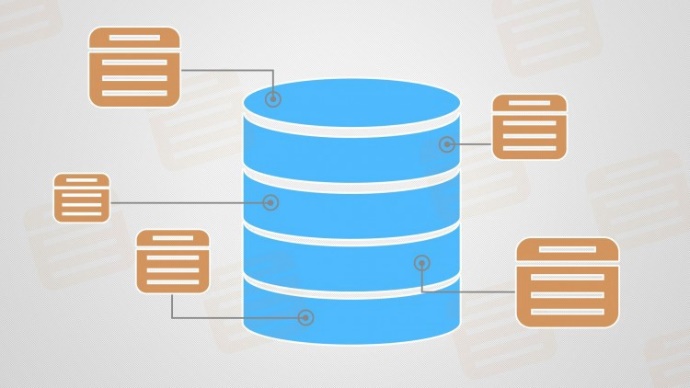
The tool that will be designed for this project is meant to generate a database and run the database with the tool to test how secure the database is and at the very end of the test say whether or not the database was protected successfully and multiple tests could be performed to the same database, but order of tests will affect the result of test, so for example if the first attack will be attack that deletes data and then use attack to steal data then nothing will be stolen, if attacks done in opposite order it will be much effective as it will not only steal the data but also delete the data from the database.

**Project Goals:**

* The main goal of my project is straight forward, and the goal is that it can analyse databases and test against various dangerous database attacks that could be very impactful.
* The secondary goal is to create a database if no database is provided to test if the tool works and to show to users with no knowledge about on how this tool is supposed to work.
* The third goal of the project is to solve the problem that the tool was designed to solve, the problem is that databases are being breached, and this tool will test the databases so that the companies would know in advance how they could prevent data breaches from happening in the future, obviously this won’t guarantee database full protection as in future more ways of breaching database will be discovered to steal and delete valuable data from databases.

# 2. Literature Review

## 2.1 – Databases

Databases are collections of Data stored in one location. “*A database is an organized collection of data*”. (Zaw, T.M., Thant, M. and Bezzateev, S.V., 2019, pp. 1-6) [1]. This shows that databases as we know them are basically locations where big collections of data are stored in one place which means that they can be simply accessed if an individual has the knowledge of how to access these collections of data which as we know is known as Database. There are many ways that the databases can be stored securely on companies’ network in the buildings. “*Though several techniques, such as encryption and electronic signatures, are currently available for the protection of data when transmitted across sites*”. (Zaw, T.M., Thant, M. and Bezzateev, S.V., 2019, pp. 1-6) [1]. This shows that there are multiple ways of securing the databases meaning that if the attacker wants to infiltrate a database, he will have to do some research first as for example if the database is encrypted through AES Encryption method and the attacker thinks he should use symmetric encryption method without any research or reconnaissance then the attack would be a failure. However, if the attacker first makes some reconnaissance for the targeted database, he would have known how to prepare for the attack so that the attack would have been successful.

**Figure 1 - Diagram of Database**

## 2.2 – Concept of Database Security

Diagram

Description automatically generatedThe Concept of databases is that databases are created to store collections of data. “*Relational database systems mathematically define arbitrary sets of stored and derived data, they have been proposed as a way of handling context-and content-dependant classification, dynamic classification, inference, aggregation, and sanitization in multilevel database systems*”. (Denning, D.E., Akl, S.G., Heckman, M., Lunt, T.F., Morgenstern, M., Neumann, P.G. and Schell, R.R., 1987. Pp.129-140) [2]. Even though this article is old it clearly states the basics of database concepts, meaning that it clearly explains what databases are made from. It also tells you that databases are ways of handling of data so that the data could be classified into categories meaning that mathematics is a strong aspect when creating a database system. One of the concepts in Database security is anomaly detection in systems, which basically a system that detects mistakes in databases which is very important in terms of database security. “*It provides a guarantee that the known attack is detected, but it cannot detect an unknown attack*” (Khan, M.I., Foley, S.N. and O'Sullivan, B., 2020. P. 6) [3]. This shows that this kind of concept is good, however if the attack that is performed is unknown to the detection system, that is when the attack could be successful as the detection system doesn’t know how to prevent and even detect the system. This also means that if the attack is successfully especially when it gets into the database server, the attacker could steal confidential as well as data that is not ready to be published by the company that owns that database. Therefore, vulnerability tester for database could be very useful in these types of cases, such as prevention of data breaches, as when the company knows the weakness of the database, they can work on improving and even eliminating this vulnerability. In the Figure 2 you can see how intrusion detection system is designed to provide the necessary precautions especially if the attack is known to the system, same idea goes for intrusion detection system designed for databases as this will help in protecting the databases and their content whether they confidential data, or data that isn’t ready to be publicly published.

**Figure 2 - Intrusion Detection System**

## 2.3 – Database Security Challenges

There are many different database security challenges, mainly because they are not taught in traditional database courses as well as the solutions for them are not simple solutions. “*The reason for focusing on these topics is because they are new Database security issues (at least not taught in traditional database courses*” (Guimaraes, M., 2006, pp. 64 – 67) [4]. This old article and now these concerns are more often taught as the cyber world is on the rise and more and more people are using online tools and big companies use databases and the concerns such as SQL injections don’t have simple solutions but there are solutions to protect databases from the attacks such as SQL injections as these attacks happen more often now, than back when this article was written namely in 2006, back then there was no such thing as Internet of Things and cyber security wasn’t much required compared to now. In nowadays is much easier to overcome Database Security challenges such as SQL injections.

Diagram

Description automatically generatedAs some challenges are eliminated in database security new challenges appear in the world of cyber security. “*Despite such advances, the database security area faces several new challenges. Factors such as the database security area faces several new challenges. Factors such as the evolution of security concerns, the “disintermediation” of access of data, new computing paradigms and applications, such as grid-based computing and on-demand business*” (Bertino, E. and Sandhu, R., 2005, p. 2) [5]. This shows that even when we manage to find solutions in database security, new challenges appear, such as introduction of new computing paradigms as when new paradigms are introduced then more issues could be discovered in certain areas such as database security as these issues may not be noticeable in previously known computing paradigms and the issues are just being introduced as new paradigms are introduced such as the grid-based paradigm mentioned in the article. As Database Security as any other security is based around the CIA acronym (Confidentiality, Integrity and Availability) and these three keywords that are seen in diagram are always challenged.

**Figure 3 - CIA Diagram**

## 2.4 – Penetration testing tools

There are many different penetration tools but not all of them could be used to test the security level of databases, such as testing for SQL injections. “*SQL injection vulnerabilities are particularly relevant, as Web services frequently access a relational database using SQL command*” (Antunes, N. and Vieira, M., 2009, pp. 301-306) [6]. This shows that even though there aren’t many databases relevant tools to test the security level of database, they are in need as many web services access databases using SQL commands meaning that easily SQL injection can be performed if the database isn’t correctly protected against malicious users on the web. Also, implementing same approach on database multiple times doesn’t give the same result. “*Implementing the same detection approach frequently detect different vulnerabilities”* (Antunes, N. and Vieira, M., 2009, pp. 301-306) [6]. This shows that for the test to be successfully the detection approach will have to be repeated to gather as much vulnerabilities as possible as there is a possibility like in this paper that first time round a set of vulnerabilities could be found and second time round using same approach could give me complete set of different lists of vulnerabilities.

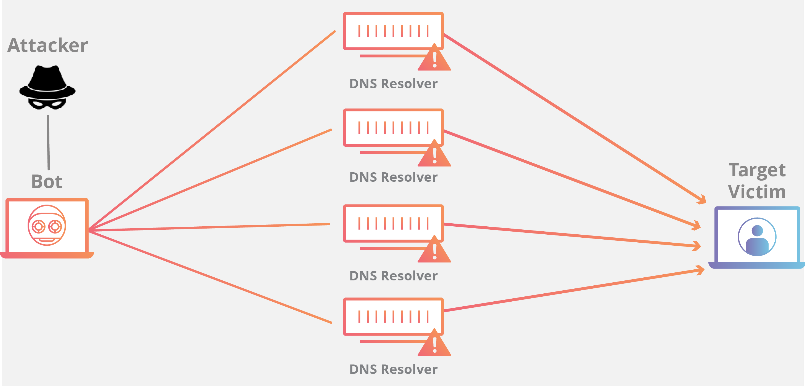
Text

Description automatically generatedA Database Security focused penetration testing tool example is SQLMAP. “*Results showed that each of the content management systems was not susceptible to SQLi attacks but gave warnings about other vulnerabilities that could be exploited”* (Ojagbule, O., Wimmer, H. and Haddad, R.J., 2018, pp1-7) [7]. This shows that content management systems were not susceptible to SQL injection attacks, however they gave warnings about other vulnerabilities which weren’t yet exploited, which means that the attack that was performed via SQLMAP shows that SQL injection was successful and in addition to that it warned the user about exploits that weren’t yet used but could eventually be used if the database won’t be protected well enough. This shows that the database that the SQLMAP was tested on could be improved before a serious attack would be performed on the database and even confidential data could be stolen if the owner of database isn’t careful enough. SQLMAP is command line-oriented database security tool, which means that the tool has very limited usage as some people who may need to use it, might not know how command line-based tools are used and/or what the commands are as seen in the diagram above.

**Figure 4** - **Screenshot of SQLMAP**

## 2.5 – Database Security Threats

One of the main Vulnerability threats for database security is a threat known as platform vulnerability. “*Database server may lead to unauthorized access, data corruption or denial of service. The Blaster Worm, for example, took advantage of a Windows 2000 vulnerability to create denial of service condition*” (Shulman, A. and Co-founder, C.T.O., 2006, p.5) [8]. According to this article database server could be vulnerable to platform attack, which as mentioned could lead to corruption of data and other possible threats, if the database server isn’t protected properly against such attacks.

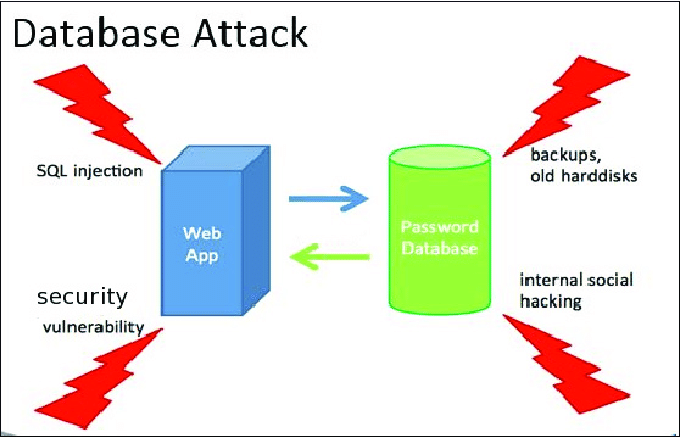
Another possible threat to databases is Denial of Service which is an attack that overloads CPU’s and memory with data so that all the services in the database that are being processed are slowed down, which may lead to the entire servers crashing as they can’t handle all the data at the same time. “*DOS techniques include data corruption, network flooding, and server resource overload (memory, CPU, etc.)*” (Shulman, A. and Co-founder, C.T.O., 2006, p.9) [9]. This shows that the Denial-of-Service attack could be performed in various ways, whether it’s done remotely or on-site, but the idea of the attack is to crash the database servers so that the attacker could gain the upper hand against their victim, this kind of attack is very effective in data breaches that have been happening across the world to big companies such as Twitch or Nvidia.

**Figure 5 - Denial of Service attack Diagram**

## 2.6 – Use of Encryption in Databases

There are three levels of encryption that are defined according to the I. Basharat. “*Three encryption levels are defined. Storage-level encryption, database-level encryption and application-level encryption”* (Basharat, I., Azam, F. and Muzaffar, A.W., 2012, p.4) [10]. Encryption at all levels is important as at each level it protects specific data. At storage level it protects the data that is stored on the drives, database level encryptions protect the entire database from access and application-level encryption protects the actual application from being accessed and used by authorised personnel. There are solutions that could increase the security of a database. “*To overcome the problem of unauthorized access of keys, two solutions were proposed. HSM and Security server approach*”. (Basharat, I., Azam, F. and Muzaffar, A.W., 2012, p.4) [10]. This shows that as new encryption issues are known, an appropriate solution is proposed and the problem is fixed that wasn’t available at the time when the attack made databases vulnerable for some time, meaning that if new issue arises, the solution is always available, however the solution may not be clear as the issue may not be fully understandable at the time when at issue has happened. Security server approach is an approach where the server’s security is being improved so that it is harder to access the server for the unauthorized personnel.

## 2.7 – Attacks on Databases and Prevention

The main type of Attack that is performed on Databases is SQL Injection and the attack can be detected in many web applications. “*SQL Injection (SQLIA) can be detected in many web applications that lack of input variable filtering*.” ( Thiyab, R.M., Ali, M.A. and Basil, F., 2017, pp. 323-331) [11]. This shows that the most used attack on databases can be easily detected in many different web applications, meaning that there is a simple way of detecting this common type of attack on the databases. Also, this prevention technique has some flaws that I will mention. “*Weak input filtration and validation of forms in dynamic web applications and using a single detection and prevention technique against SQL injection attacks.”* ( Thiyab, R.M., Ali, M.A. and Basil, F., 2017, pp. 323-331) [11]. This is a major flaw of this prevention technique as it’s only using one prevention technique which is a weak strategy as if some hackers know how this protection technique works they may figure out other ways to get into the database, meaning that this technique is only useful if there is only one way into the database in terms of SQL injection, but sooner or later the hackers will figure out different technique, which will cause issues to database owners if that’s the only way that their database is protected against SQL Injection attack.

**Figure 6 - Attacks on Database Diagram**

Denial of Service attack is a major attack technique on databases. “It is a attack that prevents the legitimate users of a program/ application/ data to use or access that specific service” (Basharat, I., Azam, F. and Muzaffar, A.W., 2012, p.3)[12]. This shows that attack of this kind could be used to manipulate the databases for the hackers to simplify their access, blocking the access to users that have legitimate access and priority to access the database. In other words this kind of attack is capable of blocking access to database for people who usually use the database and simplify the work for attackers who will have much easier job to get their work done, whether its theft of data or crashing the database servers.

Attackers, that perform cyber - attacks on databases, do this for several reasons, some are doing it to steal data that they will use later on but other attacks are performed so that the servers crash and are unusable until repaired. “*In distributed communication systems, some intruders try to hold the private information of others, try to corrupt the data or modify the data. So some measures are needed in terms of distributed communication security, security in data and data auditing*” Velmurugan, L. and Manoharan, S., 2020, p.4)[13]. This shows that even if the attack that is performed on database is the same, it doesn’t mean that the arrack was performed for the same reason, for example if the Denial of Service attack is performed twice, first time it could be so that the attackers gain access and copy confidential data, but the second attack could be performed to crash that database so that once the database is repaired, finding the details that were stolen is another issue which maybe be hidden and/ or hard to find. The point is that if same attack was performed on database it doesn’t mean that both of them had same purpose, as well as SQL Injection attack, both of these attacks could done for the same reason however the SQL Injection attack is performed only is specific environment such as database server, whereas Denial of Service attack could be performed in any environment. Each attack type has its own uses where they work best.

Another type of an attack is performed on databases could be dictionary attack. “*Dictionary – Attack defined as the way for intruders to attack the SSH servers by guessing the combinations of the numbers in order to get the pin number to access in to the target account*” (Su, Y.N., Chung, G.H. and Wu, B.J., 2011, p.65)[14]. This shows that dictionary attack is possible to performed on the database, however in order to do so, the attacker would first have to do some research into the target user account as the password and/or pin could be related to the users habits or hobbies as usual passwords are. On the other hand more and more people start to use passwords that are less related to them as this could make the account more secure if someone would use dictionary or brute force attack on the database system.

## 2.7 – Database Technology

There are two kinds of Database Technologies, the first kind is the online applications which are accessible via browser, but the other kind of Database technologies are the old fashioned download and install software technologies. “number of database technologies called usually NoSQL, like key-value, column-oriented, and document stores as well as search engines and graph databases. Whereas SQL software vendors offer advanced products with the capability to handle highly complex queries and transactions, NoSQL databases share rather characteristics concerning scaling and performance”. (Pokorný, J., 2015, pp. 1 – 12)[15].This shows that certain technologies have some perks whereas other have completely different perks as for example NoSQL is used to stored more basic data, whereas the software based technologies offer more advanced usage of the tool as the capabilities are better as they can handle far more complex queries which may not be manageable by the web-based technologies such as NoSQL. This is very important as in my project the database that I will connect should test how secure the database is and not knowing which kind of technology it is may hinder the performance of the tool that I’m planning to make.

## 2.8 – Database Vulnerabilities

Graphical user interface, text, application

Description automatically generatedThere are many Database Vulnerabilities but the main ten are mentioned in the diagram below according to the article where I found this list of vulnerabilities. (Shulman, A. and Co-founder, C.T.O., 2006, p. 1)[16]. All of these vulnerabilities and I covered three of the vulnerabilities earlier, which is SQL Injection and Denial of Service so I will just cover in this section the other eight vulnerabilities that are mentioned in this article. One of the vulnerabilities that aren’t mentioned earlier is Excessive Privilege Abuse which is when users are granted database access privileges that exceed requirements of their job function according to the article. (Shulman, A. and Co-founder, C.T.O., 2006, p. 1)[16]. Which is when a user is accessing the database but is using the access for other reasons that he is supposed to, in other words this threat that can only be used by inside man, who gains access to database server. This kind of attack is dangerous as there is no real protection against this attack as if you have access to database you have access to entire database. The Three vulnerabilities that I have mentioned are the three main ones that I will focus on for this project as try to create a tool that tests a database against every major threat will take much longer, therefore I will focus this project to create a tool that tests a database against the vulnerabilities that I have mentioned so far. Weak Authentication vulnerability is vulnerability that the authentication is weak and is quite easy to break into the database, this could be used in variety of different ways, for example easy and/or guessable password or a different way that the user can authenticate themselves that is easy to be performed by hacker, this is important vulnerability as it could be used in almost any situation as most people don’t look into providing a difficult password to crack, which is main problem in security In nowadays, as not many people see this is an important problem they need to solve as with this flaw anyone could potentially get into any account as the easy passwords have something to do to owners life like the password could be “Basketball2011” for basketball player who was born in that year or something important happened to him in that year. This is an example of Weak Authentication Vulnerability. Even though figure 7 shows up to 10 vulnerabilities, as they are the most popular attacks that are performed on databases for multiple different reasons, whether it’s to remove data, steal data or send the data to someone else.

**Figure 7 - List of Database Vulnerabilities[16]**

# 3. Requirements

|  |  |  |
| --- | --- | --- |
| Functional Requirements | | |
| Requirement No. | Requirement | Importance |
| FR1 | Must connect to database. | Must |
| FR2 | Must test the connection to the internet. | Must |
| FR3 | Must react to mouse clicks and keyboard typing. | Must |
| FR4 | Must perform SQL Injection attack when picked from list and button pressed. | Must |
| FR5 | Must have graphical interface. | Must |
| FR6 | Must open new window when running the system. | Must |
| FR7 | Must perform Data removal attack when picked from list and button pressed. | Must |
| FR8 | Must perform Weak Authentication attack when picked from list and button pressed. | Must |
| FR9 | Must terminate connection with database when pressed on Terminate Connection Button. | Must |
| FR10 | Must disable all buttons and lists apart from the ones in the first line. | Must |
| FR11 | Must generate a database when press a button to generate a database | Must |
| FR12 | Must perform the picked attack on unknown/foreign database | Must |

**FR1** – This is important as without this the connection between tool and database couldn’t be made and that is a key concept that must happen for the tool to work as meant to as without the connection the tool is pointless.

**FR2** –This is also a must as this is equally as important as without it the tool could be used but is better if the tool tested the connection to see if the strength is good enough to perform the tests on the databases, when the database may require connection to be connected especially when connecting to database server.

**FR3** – This is quite important as the tool must react to typing and mouse clicks so that it would work as meant and this is a must as without it the tool might be quite flawed as without this requirement the tool would be pointless.

**FR4** – When Clicking on the SQL injection button the system must perform the SQL Injection attack on the connected database, this is important as this is one of the attacks that this tool should test the database against.

**FR5** – The Tool must have a graphical interface; This requirement is linked to the FR6 as this is about having the graphical user interface, which is a must as without this requirement the tool will have very limited usage by users as only people who have experience with command line interface would be able to use it.

**FR6** – When running the tool, the, the window needs to open; is important as when you click onto the tool to run a new window needs to open which will provide the graphical interface so that the tool could be used by many more individuals.

**FR7** – When Clicking on the Denial-of-Service button the system must perform the Denial-of-Service attack on the connected database, this is important as this is one of the attacks that this tool should test the database against.

**FR8** – When Clicking on the Weak Authentication button the system must perform the Weak Authentication attack on the connected database, this is important as this is one of the attacks that this tool should test the database against. This attack is meant to test the strength of Encryption and passwords.

**FR9** – This is important as this is operation will allow user to terminate the connection to database and connect to another without terminating the entire system.

**FR10** – This is a requirement is important as this gives the users choice on whether they want to use the application to use prebuilt database or want to use it on their own database.

**FR11** – This requirement is important as thanks to this requirements user can see how the application works on generated database in application.

**FR12** – This is a must as this will allow users to test their own databases against these three attacks

|  |  |
| --- | --- |
| Non-Functional Requirements | |
| Requirement No. | Requirement |
| NFR1 | Anyone who has access to the program should be able to use it easily. |
| NFR2 | The tool should work as planned with no stutter or lag. |
| NFR3 | This tool should work on major platforms including Windows and Linux |
| NFR4 | The tests length should be dependent on the size of the database, if the database is big then it should take longer to complete the tests compared to small database. |
| NFR5 | The system shouldn’t terminate unexpectedly |
| NFR6 | The system shouldn’t compromise security when in use as this would defeat the purpose of using the system. |
| NFR7 | The tool should work smoothly and with no delays; the first requirement where the requirement isn’t a must but should and this is because requirement and the ones after is either should, could or won’t in the tool. This requirement is a should as because this requirement isn’t a must as it’s about the performance of tool and how reactive it will be to the users typing and clicking. |
| NFR8 | Each button click should take few seconds for the tool to react; is also about the tool reactivity which is about how responsive the tool would be but then it was about the response overall and this requirement is about the responsiveness to clicks of the buttons. |
| NFR9 | Allow wide range of database types that can be connected to via URL or locally; is a Should requirement as this requirement isn’t necessary but it would be good to have, which is compatibility with wide range of database technologies such SQL Lite and NoSQL. |

# 4. Methodology

## 4.1 – Language and IDE

The Programming language that I will use for this system will be most likely Python (Van Rossum, G., 2007) [17], as I used it most recently and the most experience with these languages. In terms of the programming paradigm that I aim to focus on is object-oriented programming (Rentsch, T., 1982) [19], as this application is centred around data analysis and manipulating the given data to user needs and tests that they will be performing on the connected database, that’s why I believe this paradigm of programming is the best option as it’s focused on how data is manipulated and used.

The Integrated Development Environment (IDE) that I will use is the default Python environment which is the built-in shell in exact IDLE Shell 3.10.1 and the graphical interface framework that I will use is ‘PySimpleGui’ (Podrzaj, P., 2019) [18], for the graphics in the system so once finished it would be more openly available to wider range of people.

These are the reasonings behind my choice of programming language and integrated Development Environment. Apart from this I don’t have much to say regarding the reasoning and choices I made/ will make regarding Language and IDE.

## 4.2 – Platform

The platform on which this system will work is easy as there are only two platforms that are required and suited for a penetration tool such as this one, namely: Windows and Linux, some reasons for this is same but there are some unique reasons for both platforms.

Firstly, let’s talk about the Windows this is a required platform as this is extremely popular platform to use by many people in the IT industry as well as this platform is very easy to use thanks to it’s design and Graphical User interface.

Next is Linux and this is a required platform as there is a specific version of Linux that is professionally used when dealing with cyber security issues and penetration tests in IT industry. As well as is quite easy to use in same regards as Windows as it also has graphical interface which makes the usability extremely easy and you don’t need remember specific phrases to use to operate it unlike command line interface operating systems as MS-DOS, however Linux uses quite a bit of Command Line though command prompt which is a lot easier if you know what to type into the command prompt. In this scenario Linux is very beneficial to know how to use it as it could be used using graphical interface such as windows but it is also capable of being used via command line via it’s command prompt.

Both of these platforms are important in cyber security sector of IT industry as mentioned before for different reasons and both of them are valid and will be valid for quite a long time unless something change drastically. Windows is the most popular and common operating system using in the world and Linux Kali is specific operating system that deals best with cybersecurity threats and penetration test in Cyber Security.

# 5. Design

## 5.1 – High-Level Design

### 5.1.1 – System Architecture

Diagram

Description automatically generated

**Figure 8 – System Architecture**

This is the system architecture for the for the system and what needs to be done and in what order these buttons need to be clicked in order to need to the best results possible and when the optional buttons need to clicked to get the best result. Every arrow that isn’t part of the main cycle are described what they are for such as if they are optional or if they are done so that user could perform multiple tests on database or perform tests on multiple databases without closing and reopening the system multiple times. This is how the architecture of the system should work without flaws, at the beginning there will be only 3 attacks to choose from and as time will go on, more attacks would be available to test out on database through this system.

### 5.1.2 – User Interface Design

A picture containing graphical user interface

Description automatically generated

**Figure 9 - System Design(Labelled) Version 1.0**

The user interface design is simple so that as soon as the system is launched anyone should understand what each button/box should mean or should do, however this is the prototype design of the first version of the design above and it may be altered to improve the usability and performance for non-technical users who need to test the database. As you can see next to URL there is the box which is meant for user to type in the URL Address for the database that is desired to test and next to that box there is a button to perform the connection between the tool and the database. The two boxes below on left is the button and the result box to test the latency if the user is questionable internet connection or just to make sure that the connection and the test would be performed safely with no interference. The wide box on version 1 design is where you can pick which test you want to perform and next to it a button to perform the attack on the database. The attack result box will either say ‘FAIL’ or ‘PASS’, if it’s passed then the database wasn’t affected by attack but if fail then it was, and the database need to make some improvement to prevent the attack from happening successfully. The final button which is ‘Terminate Connection’ is a button which you use to change the database you want to perform tests on so that the system won’t make any compromises in databases security. That’s how the User Interface will work with Version 1 of the design. This is also what each of the buttons on the user interface will be responsible for and what each button should do if it isn’t flawed or full of errors/mistakes.

### 5.1.3 – Environment Design

The Environment design in other words the environment where the system will be created is basically the python official environment and their all the stages of the coding will be performed as well as the graphical interface which will make the tool more usable by wider range of individuals. The environment design is Python as this is the programming language that I have used most recently, meaning that I will most likely have the biggest advantage when using this tool.

## 5.2 – Low-Level Design

### 5.2.1 – Design Structure

The structure of the design is simple as every bit of the structure plays a key role to perform and use the tool successfully, however in order to get the best efficiency, the order of actions made is specific to get best results. In order to get the best results. Firstly I need to decide whether you what to use a demo database or your own and after that decision made everything is exactly the same process, the only thing is, that if you pick your own database then the application would have to analyse your database in order to perform the three attacks on database and to do that, firstly the user needs to type in the Name or URL for the database on the network or if it’s a local database file, then click connect button. The next step after that is optional but highly recommended as it tests the latency between the system and the internet as if the database is located on a server, then internet speed will have an impact on how fast the test would be performed. Next is picking which attack the user wishes to perform and after that clicking the button to perform the attack is necessary to obtain a result below, on whether the attack was successful or failure on the given database. The step after that is entirely dependent on user, option number 1 select different attack to be performed on the database. Option 2 is to terminate connection and connection to different database and follow through these steps on different database again. And final option is just to close the system as all things that user wanted to do with the tool is completely done and finished. That’s how this system has been structured to perform and what it was structured for.

### 5.2.2 – Design Interactions

The Design interactions so how the system will interact with the user is explained withing the user interface design as these are linked as in there, each button and or text box are explained with detail and the possible options that the system will give to the user and how the user will interact with it. So basically, the only way that interactions will be allowed from the user in the system is through mouse clicks onto the tool whether it’s buttons or option choices for which attack the user want to perform. Apart from that the other type of interaction is keyboard typing to type in the URL link for the database that is desired to be tested using this tool.

## 5.3 – Design of Tests

The way that the tests will be implemented and designed for the system will be simple as the way that they will be designed is for example Test 1 could be does the FR1 meet the expected result, if so then the test is successful if not then the test is failure. This is performed multiple times dependant on the number of functional requirements that are listed in the requirements analysis.

|  |  |  |
| --- | --- | --- |
| **Test No.: Example** | | |
| Requirement Code | Requirement | PASS/FAIL |

The Table above shows how the Tests are designed so that when looking at the tests it’s easy to see if the test result is a Pass or Fail and for which Requirement the test result is for. This is how each test will look like so that is easy to see result, and this would be done for all of the tests with some comments regarding that particular test below the table for some insight and explanation for that exact test that was performed and what I noticed regarding that test.

# 6. Implementation

## 6.1 – Coding

Graphical user interface, application, Teams

Description automatically generatedAt the beginning phase of the coding phase I had to prepare everything to make sure that everything is ready or to make sure that everything that I believe will be needed is ready to proceed at the beginning. So, I downloaded Python as well as the graphical package that I will need onto the computer to start working on the system. The graphical package that I decided to download, and use is ‘PySimpleGUI’ and so far, the only thing that I managed to do is the graphical interface so that everything graphically is done and the only thing after that is the functions and features of the system to perform the task that the system for designed to perform. As you can see below this is how the first version of the tool looks like from graphical point of view, the only thing that isn’t visble is the places where the results of the tests will be shown and they will be shown next to ‘Latency Test’ button and next to ‘Result:’ textbox and this is how the system will look like, the next part of coding is adding the funtions to buttons as well as other features such as filtering URL textbox so that it will only allow Database URLs and nothing else.

**Figure 10 - Version 1.0 of the System**

Graphical user interface, application

Description automatically generatedText

Description automatically generatedDuring implementing the system I decided to write another python code to create a database, insert data into the database as well as as use select statement to prove that this database works as meant to, and I create this code on separate ‘.py’ file. As seen in the screenshot this is what data the database that I create is holding only five rows of data as I beieve only 5 rows of data is sufficient enough when I will be performing tests on this database in later phase of the project. Also as you can see the final design looks different that before this is because I made some changes to it such as eliminate the result space on the main application and instead just a popup window will show with the result which I think is better as anyway no one will perform two tests at the same time which means that this change didn’t affect the final product in anyway . As you can see above this is how the latest version of the system looks like. In terms of coding the functions the Latency test function is coded as well as the bottom two buttons are finsished as well, the only things left to code is the connect button as well as the attack function. I also decided to combine the database creation into the system so if anyone wants to test the system always will be able to without extensive system that creates the database, meaning that I also had to add two more buttons so that if the user decides to use the demonstrative database then he can create the database but if he wants to use own databas ethen he also can do that but first he nneeds to pick which database he will use. With the updates that I have made over the period when I started imlementation means that the system architecture will look bit different, the only difference is at beginning there will be a section added for user to choose whether he wants to choose the prebuilt database build or their own database and that’s the only difference.

**Figure 12 - Version 2.0 of System**

**Figure 12 - Version 2.0 of the System**

**Figure 11 - Database**

I have made an application so that works with SQLite databases for now and for helpfulness I added as much comments as possible so if I forget what each function is used for, the comment will help me in understanding the code and as well as that I added print function when possible so that what happens on graphical interface will also happen on python shell in command line format so that if someone is confused with graphical interface or forgot what he was doing, he can just check the shell and he will she what just happened on that application. The screenshot below show entire procedure that can be done on demo database from pythons shell, meaning that everything that happens is noticeable and understandable.

Timeline

Description automatically generated with medium confidence So, on the first line it shows a command that Demo database has been chosen and below that the database is printed out on the shell so that the user can see the details of the database and the print command is shown that this database has been created. Then if you like you can test the network connection so that you know if you’re connected to the internet. Next three lines are print commands for every test that this application can do with the database that has been connected to and what the result is, whether it is a pass and attack was unsuccessful or the other way. Next command shows that the connection to the database has been closed and the final command is to show that the appplication has been closed. This is an example of how the application will work. But also if the attacks are performed in specific order the rows will be printed out that are in the database, the reason they aren’t in the shown run is because I begin with denial of service attack which deletes the rows from the table and that’s why there are none shown. As the presented order of attacks shown in the screenshot is delete the rows thorugh denial of service attack, then using weak authenictaion attack steal the data but there is none and the final attack is injectng more rows to the table showing that SQL Injection has happened. The only thing that doesn’t work as planned so far is user input, meaning that user can connect to existing database but if he/she types in a database name that doesn’t exist it should show error message that database isn’t found, but instead it creates a database file with that name.

**Figure 13 - Procedure of the Application**

Text, letter

Description automatically generatedAn update on the process of the implementation. I have managed to fix that error as in user input if the file name is found it will connect to it and if it isn’t found then it will show the error message of file name wasn’t found and no new database file will be created. Also, for the use from unexperienced users I added a button that briefly explain what is the goal of each attack which will be shown with popup messages.

**Figure 14 – Final process with the demonstration route**

I have performed a full run of the application after I got a feedback from project in progress day which will make the message box text and everything else more clear and understandable. As seen in figure 15 it’s much clearer as instead ‘FAIL’ it says ‘Attack Successful’, which is more understandable as when it’s says fail it to much open ended and it could mean anything even that the attack failed which isn’t the case, that’s why I changed the message to be more self-explanatory and understandable for users who doesn’t fully understand the purpose of the application.

## 6.2 – Testing

|  |  |  |
| --- | --- | --- |
| **Test No.: 1** | | |
| FR1 | Must connect to database. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 2** | | |
| FR2 | Must test the internet connection. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 3** | | |
| FR3 | Must react to mouse clicks and keyboard typing. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 4** | | |
| FR4 | Must perform SQL Injection attack when picked from list and button pressed. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 5** | | |
| FR5 | Must have graphical interface. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 6** | | |
| FR6 | Must open new window when running the system. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 7** | | |
| FR7 | Must perform Data removal attack when picked from list and button pressed. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 8** | | |
| FR8 | Must perform Weak Authentication attack when picked from list and button pressed. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 9** | | |
| FR9 | Must terminate connection with database when pressed on Terminate Connection Button. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 10** | | |
| FR10 | Must disable all buttons and lists apart from the ones in the first line. | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 11** | | |
| FR11 | Must generate a database when press a button to generate a database | Result: PASS |

|  |  |  |
| --- | --- | --- |
| **Test No.: 12** | | |
| FR12 | Must perform the picked attack on unknown/foreign database | Result: FAIL |

# 7. Project Evaluation

## 7.1 – Evaluation of Research

In terms of the research or Literature review I think that I ‘ve done well, not amazing but in middle, if rated out of 5 it would be 2.5 as I believe the research was done at okey standard, however the thing that I’m not confident is the relevance of the work towards my project as well as structure of that section of the report. On the other hand, I believe that writing and sections of research was done very well and analysis of the citations I picked in the related reports. Overall, I believe that I done okey in this section of the Digital Systems Project. In terms of the Research some of the references are quite old and the reason for this is that these articles included important information that was crucial to understanding the project comprehensively, one of the articles is quite old because these topics haven’t changed that much since them, one of the papers included the concept of database security which has been basically the same for a long time, even though it has evolved and improved overtime to improve database security to deal with nowadays security threats.

## 7.2 – Evaluation of Requirements

In terms of the requirements, I think that I have done this section very well overall, however over the time I kept changing them because something didn’t work, either I mixed functional requirements with non-functional requirements or didn’t put enough requirements down so that the full project idea would make sense. Excluding these things, I think it was done quite well as in each requirement I clearly stated what each of them is used for and later I went into depth in further explain the importance of each requirement. Overall, I believe that my work on requirement analysis is something around 3.5/5 as it wasn’t perfect, but I tried to do it to best of my ability.

## 7.3 – Evaluation of Methodology

In terms of methodology, I think I did alright, not well but an average, as when I was doing section of the report, I was easily distracted to do different section of the project and even make correction that I noticed during a sentence was written. Meaning that methodology was quite poor in my opinion as, when I noticed an error, whilst doing something else I went to that error to repair the mistake and then resume what I was doing, resulting in forgetting what I was writing then, which isn’t a good methodology in my opinion however I believe that my result was better than I was expecting as I’m not very confident when writing scripts in python. Overall, I would rate this section as 3/5 as I did it to the best of my ability.

## 7.4 – Evaluation of Design

In terms of the design of the project, I believe I did this section on average as I completed the design which is good and once I moved on and doing the later stage I remembered crucial components that I had to fix in the design, meaning that I had to go back to design to fully complete the design as I forgot about a crucial component of the design, which in this case was the decision that allows the user to pick a demo database or foreign database which is important part of the system, but I completely forgot about it till when I started the implementation, showing that in some sections of projects I rushed it as I didn’t think the section through and moved done, resulting in later return to that section to make the corrections which at least I managed to make so that the design of graphical interface isn’t flawed.

## 7.5 – Evaluation of Implementation

In terms of the implementation which includes the coding and testing and believe I did alright for the coding knowledge I obtain as well as I’m not very confident in general in coding, furthermore I believe that what I did in terms of coding I did alright. In terms of coding specifically I did alright, and this is because I’m not confident in coding and what I managed to do with coding is quite good, for what I know in terms of coding overall. In terms of testing there was not much space for error as I was just testing the project and in terms of testing the functional requirements the project passed all the tests apart from the last one which is the test that performs the attack on the third-party database/unknown database that isn’t generated in the system. Overall, I will rate this section of the project as 3/5 as I did quite well for what I know and my experience in coding. Also, during project-in-progress day from the feedback I came to realize that denial of service attack and data removal attack are two separate things and thanks to this feedback I managed to fix these mistakes in the actual tool as well as the report of the project.

# 8. Further Work and Conclusions

In Conclusion the project was done in average quality, as some stages of the project were done very well, whereas other stages of the project not so well, for example I believe that my design stage was done very well but the stage that I don’t have done well is how relevant the literature review is to my project as I’m not too confident on the literature reviews relevancy to the project. Overall, I have managed to create a system that test databases’ most common attacks, however only on the generated database but the thing that I didn’t manage to fully complete is the systems tests on foreign databases so the ones that are not generated within the system. This project was average quality overall in my opinion as some stages I believe I wasted time on some stages of the project as sometimes I made the work harder for myself by overcomplicating a design for the project, which wasn’t necessary for the project to succeed in what it was meant to do, therefore it slowed down my progress for the project. However, thanks to my research for this project it helped me to deepen my understanding of database security and the encryption of databases. Summing it all up, I believe that I was successful in completing my project, even though the system isn’t full working, the system can show the potential that it can achieve when it will be fully completed. Also the system could be updated and upgraded overtime so that in future it could test databases against other attacks, potentially against attacks that don’t exist yet, this tool is just the beginning that could be developed and improved as time goes on, meaning that it’s potential can be limitless as more attacks could be invented meaning that when the new attack is known, then this tool could be implemented in a way to add this attack into the list of attacks so that user could use this new attack and test this new attack against his database so that he could know if his/hers database is protected against the new attack that was recently revealed.

# Glossary

**Authentication** - Providing some evidence to verify yourself to be the user you claim to be.

**Authorization** - In this case it is a security mechanism to determine access levels or user/client privileges related to the system resources which includes files, services, computer programs, data and applications

**Availability** - In this case this term means that the data should be available to the authorized users who should be able to use the data appropriately

**Confidentiality** - In this case it’s preserving authorized restrictions on information access and disclosure, meaning protecting personal privacy and proprietary information

**Cyber Security** - Protection of computer systems and networks from information disclosure, theft of or damage to their hardware, software or electronic data.

**Database** - An organized collection of structured information or data

**Denial of Service** - Cyber-attack in which attacker seeks to make a machine or network resources unavailable

**Integrity** - In this case it means guarding data and information against improper information modification or destruction

**Python** - One of the most popular programming and software development languages on the internet

**SQL Injection** - Cyber-attack where code is injected into data-driven applications

# References / Bibliography

[1] Zaw, T.M., Thant, M. and Bezzateev, S.V., 2019, June. Database security with AES encryption, elliptic curve encryption and signature. In *2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)* (pp. 1-6). IEEE. [Accessed 22nd October 2021)

[2] Denning, D.E., Akl, S.G., Heckman, M., Lunt, T.F., Morgenstern, M., Neumann, P.G. and Schell, R.R., 1987. Views for multilevel database security. *IEEE Transactions on Software Engineering*, (2), pp.129-140. [Accessed 23rd October 2021]

[3] Khan, M.I., Foley, S.N. and O'Sullivan, B., 2020. Database Intrusion Detection Systems (DIDs): Insider Threat Detection via Behavioural-based Anomaly Detection Systems--A Brief Survey of Concepts and Approaches. *arXiv preprint arXiv:2011.02308*. [Accessed 23rd October 2021]

[4] Guimaraes, M., 2006, September. New challenges in teaching database security. In *Proceedings of the 3rd annual conference on Information security curriculum development* (pp. 64-67). [Accessed 24th October 2021]

[5]Bertino, E. and Sandhu, R., 2005. Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and secure computing*, *2*(1), pp.2-19. [Accessed 19th October 2021]

[6] Antunes, N. and Vieira, M., 2009, November. Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services. In *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing* (pp. 301-306). IEEE. [Accessed 26th October 2021]

[7] Ojagbule, O., Wimmer, H. and Haddad, R.J., 2018, April. Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP. In *SoutheastCon 2018* (pp. 1-7). IEEE. [Accessed 26th October 2021]

[8] Shulman, A. and Co-founder, C.T.O., 2006. Top ten database security threats. *How to Mitigate the Most Significant Database Vulnerabilities*. (p.5). IEEE. [Accessed 27th October 2021]

[9] Shulman, A. and Co-founder, C.T.O., 2006. Top ten database security threats. *How to Mitigate the Most Significant Database Vulnerabilities*. (p.9). IEEE. [Accessed 27th October 2021]

[10] Basharat, I., Azam, F. and Muzaffar, A.W., 2012. Database security and encryption: A survey study. *International Journal of Computer Applications*, *47*(12). (p.4). IEEE. [Accessed 29th October 2021].

[11] Thiyab, R.M., Ali, M.A. and Basil, F., 2017, April. The impact of SQL injection attacks on the security of databases. In *Proceedings of the 6th International Conference of Computing & Informatics* (pp. 323-331). [Accessed 1st November 2021]

[12] Basharat, I., Azam, F. and Muzaffar, A.W., 2012. Database security and encryption: A survey study. *International Journal of Computer Applications*, *47*(12). [Accessed 2nd November 2021].

[13] Velmurugan, L. and Manoharan, S., 2020. Designing Factors of Distributed Database System: A Review. *Data Mining and Knowledge Engineering*, *12*(1), pp.7-10. [Accessed 2nd November 2021]

[14] Su, Y.N., Chung, G.H. and Wu, B.J., 2011. Developing the upgrade detection and defense system of SSH dictionary-attack for multi-platform environment. *Online at (http://www. SciRP. org/journal/ib)*, *55*. [Accessed 4th November 2021]

[15] Pokorný, J., 2015, June. Database technologies in the world of big data. In *Proceedings of the 16th International Conference on Computer Systems and Technologies* (pp. 1-12). [Accessed 11th November 2021]

[16] Shulman, A. and Co-founder, C.T.O., 2006. Top ten database security threats. *How to Mitigate the Most Significant Database Vulnerabilities*. [Accessed 12th November 2021]

[17] Van Rossum, G., 2007, June. Python Programming Language. In *USENIX annual technical conference* (Vol. 41, p. 36). [Accessed 16th January 2022]

[18] Podrzaj, P., 2019, August. A brief demonstration of some Python GUI libraries. In *Proceedings of the 8th International Conference on Informatics and Applications ICIA2019* (pp. 1-6). [Accessed 16th January 2022]

[19] Rentsch, T., 1982. Object oriented programming. *ACM Sigplan Notices*, *17*(9), pp.51-57. [Accessed 16th January 2022]

# Appendix A: Project Proposal

**Project Title:** Database Vulnerability Tester

**Tutor:** Dr Phil Legg

**Project Proposal:** My project idea is a Database Vulnerability Tester which is basically a program that analyzes databases and scans them for vulnerabilities. The problem that this will solve is that there are many data breach in nowadays and this project will help companies to improve the security of their databases as the idea behind the project is for people who have no to little knowledge about database security and running this program against a database system to test how secure the database is and then the program will list all the test it performed and whether the test was successful or not. I have looked on google scholar to research about the idea and I found a similar tool called SqlMap which is the principal idea of my project, however that’s a command line centric tool and I aim to add graphical interface, which will make the tool more accessible to people who have no clue about database security. The program will test a database against most common attack techniques on databases, which includes SQL Injection, Weak Authentication attack etc.

# Appendix B: Meeting with Supervisor

|  |  |  |
| --- | --- | --- |
| **Date** | **Reason** | **Description** |
| 19/10/2021 | Introductory Meeting | This meeting was to meet my supervisor and to discuss the project for the first time so that I could start my research for the Database Security research. |
| 02/11/2021 | 1st Meeting | Update on project research phase and conversation regarding the project and how I will implement the project idea talk |
| 11/11/2021 | 2nd Meeting | Update on the research as well as discussion on the next move. Talking about certain topics I should research which includes Database Technologies as well as some of the main vulnerabilities that I should do the research on. |
| 16/11/2021 | 3rd Meeting | Update on my project as well as discussion on what is the difference between functional and non-functional requirements for the requirement analysis section of the report. |
| 23/11/2021 | 4th Meeting | In depth talk about requirement analysis and so that I should prepare requirement analysis for next week. |
| 30/11/2021 | 5th Meeting | Revising the requirement analysis with supervisor and feedback so that I could make the necessary improvements for requirement analysis. |
| 14/12/2021 | 6th Meeting | Talking about what I have been doing with project in past 2 weeks and my progress, with some questions regarding the design section of the report |
| 18/01/2022 | 7th Meeting | Updating on the progress with the supervisor so that the supervisor knows at what stage I’m at. Also, a discussion on what is the next stage in the project is to succeed. As well as talk about the project-in-progress day. |
| 25/01/2022 | 8th Meeting | An update on the project-in-progress day poster and implementation as well as discussion on the next steps in the process. Also, we talked about the video for project-in-progress day. |
| 03/02/2022 | 9th Meeting | An update on poster and feedback on the poster from supervisor so that I know what I need to do to make the required improvements. So that on the next meeting I can provide much better result so that hopefully I can focus on the video for the project in progress day. |
| 17/02/2022 | 10th Meeting | On this meeting I gave an update on my progress which was developing user input section in the tool as well as working on the video for project in progress day, as the project in progress day is on 2nd March 2022. |
| 24/02/2022 | 11th Meeting | This meeting was about the video and project in progress day, where I gave my update on project in progress day and my overall progress of the project. We as well discussed the actual project in progress day and the session that I will be in as well as who will be my 2nd marker for the project. |
| 10/03/2022 | 12th Meeting |  |
| 17/03/2022 | 13th Meeting |  |
| 24/03/2022 | 14th Meeting |  |
| 31/03/2022 | 15th Meeting |  |
| 07/04/2022 | 16th Meeting |  |

# Appendix C: Source Code

###########################################################################Imports

#import for graphical interface

import PySimpleGUI as sg

#import for SQL use

import sqlite3 as sql

#import to check network connection

import requests

#import to show database on shell

import pandas as pd

#import to look for data/files on the system

import os

#########################################################################Foundation

sg.theme('DarkBlue4')

#Variables for list items

DOS = 'Delete Data'

WA = 'Access Data'

SQLI = 'Insert Data'

#Variables for Demo

con = sql.connect('Demo.db')

cur = con.cursor()

#Layout

layout = [ [sg.Text('Demo Database or your Own:'), sg.Button('Demo Database'), sg.Button('Own Database')],

[sg.Button(('Create Database'), disabled = True), sg.Text('Database:'), sg.InputText((), key= 'Database', disabled = True), sg.Button(('Connect'), disabled = True)],

[sg.Button(('Latency Test'), disabled = True)],

[sg.Listbox(values=(DOS, WA, SQLI), size=(30,3), key='Attacks', disabled = True),sg.Button(('i'),disabled = True), sg.Button(('Attack'), disabled = True)],

[sg.Button(('Terminate Connection'), disabled = True),sg.Button('Close Application')]]

#Variables for Own Database

conn = sql.connect('Database')

cursor = conn.cursor()

window = sg.Window('Database Tester', layout)

event, values = window.read()

##########################################################################Functions

def Create():

con = sql.connect('Demo.db')

cur = con.cursor()

#Creating the database table

cur.execute('''

CREATE TABLE IF NOT EXISTS Demo

([ID] INTEGER PRIMARY KEY, [Name] TEXT, [Password] CHAR, [Year] INT)''')

#Inserting data into the Demo table of database

cur.execute('''

INSERT or REPLACE INTO Demo (ID, Name, Password, Year)

VALUES

(1, 'James Johannson', 'Shhsbwdjajhz736', 1974),

(2, 'Sarah Gordon', 'SjhddbiIDADUYU208!', 1987),

(3, 'Christopher Johnson', 'Sjsakjjbjkad058%', 1998),

(4, 'Marianna Pips','KSKNLSLkkncjai298\*', 2000),

(5, 'Jacob Jordan', 'Starhcj209Hsyat$', 2003)

''')

#Commiting the SQL Queries

con.commit()

#Selecting everything that there is in the Demo Table of the Database

cur.execute('''

SELECT \* FROM Demo

''')

#Print on command line shell what has been selected in using Select query

df = pd.DataFrame(cur.fetchall())

print(df)

#GUI popup message box

sg.popup("Database 'Demo' has been created")

#Command Line Record

print("Database 'Demo' has been created\n")

#Connecting to the database

def Connection():

#set the input text as a variable

conn = values['Database']

#if this file in the variable is found connect to it

if os.path.isfile(conn):

sql.connect(values['Database'])

#Command Line Record

print("Connected to Database")

#GUI popup message box

sg.popup('Connected to Database')

#if the file isn't found do this action

else:

#Command Line Record

print("No Database has been found with this name\n")

#GUI popup message box

sg.popup('No Database has been found with this name')

#User decides whether using database created in the system or their own

def Decision():

window['Create Database'].update(disabled=True)

window['Connect'].update(disabled=True)

window['Database'].update(disabled=True)

window['i'].update(disabled=True)

window['Latency Test'].update(disabled=True)

window['Attacks'].update(disabled=True)

window['Attack'].update(disabled=True)

window['Terminate Connection'].update(disabled=True)

#If clicked on demo database then create button will be enabled

if event == 'Demo Database':

window['Demo Database'].update(disabled=True)

window['Own Database'].update(disabled=True)

window['Create Database'].update(disabled=False)

window['i'].update(disabled=False)

window['Latency Test'].update(disabled=False)

window['Attacks'].update(disabled=False)

window['Attack'].update(disabled=False)

window['Terminate Connection'].update(disabled=False)

#GUI popup message box

sg.popup('You have chosen Demo Database')

#Command Line Record

print("You have chosen Demo Database\n")

#If clicked on Own Database then connect button will be enabled

elif event == 'Own Database':

window['Demo Database'].update(disabled=True)

window['Own Database'].update(disabled=True)

window['Connect'].update(disabled=False)

window['Database'].update(disabled=False)

window['i'].update(disabled=False)

window['Latency Test'].update(disabled=False)

window['Attacks'].update(disabled=False)

window['Attack'].update(disabled=False)

window['Terminate Connection'].update(disabled=False)

#GUI popup message box

sg.popup('You have chosen your own Database')

#Command Line Record

print("You have chosen your own Database\n")

#Testing connection to the network

def LatencyTest():

try:

#check the connection to internet by tryiing to access google.com using requests package

requests.get('https://www.google.com/').status\_code

LatencyTest = 'PASS'

#if the connection is successful show this popup

#GUI popup message box

sg.popup('PASS - Connection to the network is strong enough')

#Command Line Record

print('PASS - Connection to the network is strong enough\n')

except:

LatencyTest = 'FAIL'

#if the connection isn't successful show this popup

#GUI popup message box

sg.popup('FAIL - Connection to the network is too weak')

#Command Line Record

print('FAIL - Connection to the network is too weak\n')

#exit()

#performing database attack dependant on what was chosen from the list and the result of the attack, this function is performed when the database is generated inside the app

def AutoAttack():

#ResultProcess variable affects the outcome of the function

ResultProcess = False

#If denial of service attack is picked do this action

if values['Attacks'] == [DOS]:

#try to complete these commands

try:

#Deleting records from database

#looping the amount of times equal to number of rows

cur.execute("DELETE FROM Demo")

#if something isn't completed then do this

except:

ResultProcess = True

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("Attack Failed - The Data was protected")

#Command Line Record

print("Attack Failed - The Data was protected\n")

else:

#GUI popup message box

sg.popup("Attack Successful - The Data has been deleted")

#Command Line Record

print("Attack Successful - The Data has been deleted\n")

#if weak authentication attack is picked do this action

elif values['Attacks'] == [WA]:

#try to complete these commands

try:

#check if database can be accessed

#looping the amount of times equal to number of rows

for row in cur.execute("SELECT \* FROM Demo"):

#if there or no rows print this out and in popup message

#GUI popup message box

sg.popup(row)

#Command Line Record

print(row)

#if something isn't completed then do this

except:

ResultProcess = True

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("Attack Failed - The Weak Authentication attack wasn't Successful")

#Command Line Record

print("Attack Failed - The Weak Authentication attack wasn't Successful\n")

else:

#GUI popup message box

sg.popup("Attack Successful - The Weak Authentication attack was Successful")

#Command Line Record

print("Attack Successful - The Weak Authentication attack was Successful\n")

#if SQL Injection attack is picked do this action

elif values['Attacks'] == [SQLI]:

#try to complete these commands

try:

#Inserting data into the Demo table of database via SQL Injection

cur.execute('''

INSERT or REPLACE INTO Demo (ID, Name, Password, Year)

VALUES

(6, 'Andrew Clark', 'Shhsdfgsagajhz736', 1974),

(7, 'John Hiller', 'SjhddytrtdADUYU208!', 1987),

(8, 'Phil Johnsen', 'Sjs grtbjkad058%', 1999)

''')

#Selecting all files in the database file with added files

cur.execute('''

SELECT \* FROM Demo

''')

#if something isn't completed then do this

except:

ResultProcess = True

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("Attack Failed - The SQL Injection attack wasn't Successful")

#Command Line Record

print("Attack Failed - The SQL Injection attack wasn't Successful\n")

else:

#GUI popup message box

sg.popup("Attack Successful - The SQL Injection attack was Successful")

#Command Line Record

print("Attack Successful - The SQL Injection attack was Successful\n")

#If no attack types was selected print/popup message this

else:

#GUI popup message box

sg.popup("Error: Attack Type not Selected")

#Command Line Record

print("Error: Attack Type not Selected\n")

#performing database attack dependant on what was chosen from the list and the result of the attack, this function is performed when the database isn't generated from app

def attack():

#ResultProcess variable affects the outcome of the function

ResultProcess = False

#If denial of service attack is picked do this action

if values['Attacks'] == [DOS]:

#try to complete these commands

try:

pass

#if something isn't completed then do this

except:

pass

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("PASS - The SQL Injection attack wasn't Successful")

#Command Line Record

print("PASS - The SQL Injection attack wasn't Successful\n")

else:

#GUI popup message box

sg.popup("FAIL - The SQL Injection attack was Successful")

#Command Line Record

print("FAIL - The SQL Injection attack was Successful\n")

#if weak authentication attack is picked do this action

elif values['Attacks'] == [WA]:

#try to complete these commands

try:

pass

#if something isn't completed then do this

except:

pass

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("PASS - The SQL Injection attack wasn't Successful")

#Command Line Record

print("PASS - The SQL Injection attack wasn't Successful\n")

else:

#GUI popup message box

sg.popup("FAIL - The SQL Injection attack was Successful")

#Command Line Record

print("FAIL - The SQL Injection attack was Successful\n")

#if SQL Injection attack is picked do this action

elif values['Attacks'] == [SQLI]:

#try to complete these commands

try:

pass

#if something isn't completed then do this

except:

pass

#If test was successfully done print/popup 'Fail' otherwise print 'Pass'

if ResultProcess == True:

#GUI popup message box

sg.popup("PASS - The SQL Injection attack wasn't Successful")

#Command Line Record

print("PASS - The SQL Injection attack wasn't Successful\n")

else:

#GUI popup message box

sg.popup("FAIL - The SQL Injection attack was Successful")

#Command Line Record

print("FAIL - The SQL Injection attack was Successful\n")

#If no attack types was selected print/popup message this

else:

#GUI popup message box

sg.popup("Error: Attack Type not Selected")

#Command Line Record

print("Error: Attack Type not Selected\n")

#Refreshes all the fields to prepare for the new attack on new database

def ConTermination():

#remove the text from the user input field

window['Database'].update("")

#unselect values from list if any are selected

window['Attacks'].set\_value([])

window.refresh()

#close the connection with database

con.close()

#show this popup

#GUI popup message box

sg.popup('Connection with database was terminated')

#Command Line Record

print('Connection with database was terminated\n')

#Reset all buttons and fields to how they were when booting the system up

window['Demo Database'].update(disabled=False)

window['Own Database'].update(disabled=False)

window['Create Database'].update(disabled=True)

window['Connect'].update(disabled=True)

window['Database'].update(disabled=True)

window['Latency Test'].update(disabled=True)

window['Attacks'].update(disabled=True)

window['Attack'].update(disabled=True)

window['Terminate Connection'].update(disabled=True)

#Close the application

def CloseApp():

#close the application

window.close()

#show this popup

#GUI popup message box

sg.popup('Application has been closed')

#Command Line Record

print('Application has been closed\n')

#Show information regarding each attack type

def information():

#Show information about inserting attack

sg.popup('Insert Data - this attack is about testing the database against unathorized insertion of data into the connected database')

print("Insert Data - this attack is about testing the database against unathorized insertion of data into the connected database\n")

#Show information about accessing database attack

sg.popup('Access Data - this attack is about testing the database against unauthorized access to the database')

print("Access Data - this attack is about testing the database against unauthorized access to the database\n")

#Show information about deletion of data in database attack

sg.popup('Delete Data - this attack is about testing the database against unauthorized deletion of data within database')

print("Delete Data - this attack is about testing the database against unauthorized deletion of data within database\n")

#Functions in the correct order

#Decision function is so that the user can decide whether he want a test on demo database or his own

#Create function is to create the demo database

#Connection function to allow user to connect to already existing database

#LatencyTest function to see if internet connnection is strong enough - this is optional

#AutoAttack function to perform the attack and see what the result from the attack is with generated database

#AutoAttack function to perform the attack and see what the result from the attack is with foreign database

#ConTermination function to close the connection between the application and the database

#CloseApp function to close the entire app down

#########################################################################Main Event Loop

#Main event loop starts here

Decision()

#while this loop is true repeat all the events within this loop

while True:

event, values = window.read()

#if create database is clicked perform Create function

if event == 'Create Database':

Create()

#if connect button is clicked perform Connection function

elif event == 'Connect':

Connection()

#if latency test button is clicked perform Latency test

if event == 'Latency Test':

LatencyTest()

#if attack button is clicked perform Attack function

if event == 'Attack':

#if create button was clicked at the beginning run autoattack function

#if event == 'Create Database':

AutoAttack()

#if connect button was clicked at beginning run attack function

#elif event == 'Connect':

# Attack()

#else do nothing

#else:

#pass

#if terminate connection button is clicked perform ConTermination function

if event == 'Terminate Connection':

ConTermination()

window.refresh()

Decision()

#if i button is clicked information about attacks will be shown and what the attack will do to the database.

if event == 'i':

information()

#if close application button is clicked perform CloseApp function

if event == 'Close Application':

CloseApp()

#if the application is closed break from the loop

#this statement is important as without this the application will crash when closing the application

if event == sg.WIN\_CLOSED:

break