

**Faculty of Science and Engineering**

**Dissertation Title:** SQL Injection: - The popular methods, solutions and is it still an issue

**Student Name:** James Braznell

**Student Id:** 1007022

**Supervisor:** Jun Li

**Examination Board:** Postgraduate Computing

**Award Title: Msc Computer Science**

Presented in partial fulfilment of the assessment requirements for the above award.

This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

It is acknowledged that the author of any dissertation work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free licence to do all or any of those things referred to in section 16(i) of the Copyright Designs and Patents Act 1988 (viz: to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make an adaptation of the work).

Signature:…James Braznell…………………………….

Date:……30/06/15……………………………….

**Dissertation Declaration**

**This document must accompany all Dissertation submissions**

**PLEASE READ THIS VERY CAREFULLY.**

The University considers seriously all acts of Academic Misconduct, which by definition are dishonest and in direct opposition to the values of a learning community. Misconduct may result in penalties ranging from the failure of the assessment to exclusion from the University.

Further help and guidance can be obtained from your academic tutor or from our guide on How to avoid Academic Misconduct – available at <http://www.wlv.ac.uk/skills>

**By submitting this document for assessment you are confirming the following statements**

I declare that this submission is my own work and has not been copied from someone else or commissioned to another to complete.

Any materials used in this work (whether from published sources, the internet or elsewhere) have been fully acknowledged and referenced and are without fabrication or falsification of data.

I have adhered to relevant ethical guidelines and procedures in the completion of this assignment.

I have not allowed another student to have access to or copy from this work.

This work has not been submitted previously.

**By this declaration I confirm my understanding and acceptance that –**

1. The University may use this work for submission to the national plagiarism detection facility. This searches the internet and an extensive database of reference material, including other students’ work and available essay sites, to identify any duplication with the work you have submitted. Once your work has been submitted to the detection service it will be stored electronically in a database and compared against work submitted from this and other Universities. The material will be stored in this manner indefinitely.

2. In the case of project module submissions, not subject to third party confidentiality agreements, exemplars may be published by the University Learning Centre.

I have read the above, and declare that this is my work only, and it adheres to the standards above.

Signature\_\_James Braznell\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date\_\_\_30/06/15\_\_\_\_\_\_\_\_\_\_

Print Name\_\_\_James T Braznell\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Student ID\_1007022\_\_\_\_\_\_\_

🞏 I have submitted this digitally and will provide a signed copy prior to marking.

Abstract

This paper looks at the issue of SQLIA and has performed extensive research on the popular SQLI attacking methods in use by malicious individuals aiming to damage a database, steal confidential data etc. The paper also looks at the solutions developed to detect and prevent these attacks which come in various forms such as defensive coding, intrusion detection systems, firewalls etc. This research was used to select candidates for both attacking methods and solutions for testing on several popular DBMSs to see their effectiveness and from this produce a report that aims to answer the question of whether SQLIA is still an issue.

Table of Contents

Abstract iii

Table of Contents iv

1 Introduction 1

1.1 Background and Context 1

1.2 Scope and Objectives 3

1.3 Overview of Dissertation 4

2 Literature Review 5

3 Technical Chapters (change this to something appropriate) Research Methods, Findings, Critical Self-Evaluation 22

3.1 Research Method 22

3.2 Artefact 22

3.2.1 Artefact - DBMS SQLIA testing 23

3.2.2 Artefact - Burp Suite testing 25

3.2.3 Artefact - Metasploit testing 25

3.2.4 Artefact - Report 26

4 Conclusion 28

4.1 Summary 28

4.2 Evaluation 28

5 References 29

6 Appendix 1 - Proposal 34

7 Appendix 2 – User guide 55

8 Appendix 3 – Installation guide 56

9 Appendix 4 – Artefact - DBMS SQLIA testing 57

9.1 Tautologies 57

9.1.1 MySQL tautology attack 57

9.1.2 MySQL tautology attack results 57

9.1.3 Oracle tautology attack 57

9.1.4 Oracle tautology attack results 57

9.1.5 MSQLS tautology attack 58

9.1.6 MSQLS tautology attack results 58

9.2 Illegal/Logically Incorrect Queries 58

9.2.1 MySQL Illegal/Logically Incorrect Queries attack 58

9.2.2 MySQL Illegal/Logically Incorrect Queries attack results 58

9.2.3 Oracle Illegal/Logically Incorrect Queries attack 59

9.2.4 Oracle Illegal/Logically Incorrect Queries attack results 59

9.2.5 MSQLS Illegal/Logically Incorrect Queries attack 59

9.2.6 MSQLS Illegal/Logically Incorrect Queries attack results 59

9.3 Union Query 60

9.3.1 MySQL Union Query attack 60

9.3.2 MySQL Union Query attack results 60

9.3.3 Oracle Union Query attack 60

9.3.4 Oracle Union Query attack results 60

9.3.5 MSQLS Union Query attack 61

9.3.6 MSQLS Union Query attack results 61

9.4 Piggy-backed queries 61

9.4.1 MySQL Piggy-backed Queries attack 61

9.4.2 MySQL Piggy-backed Queries attack results 61

9.4.3 Oracle Piggy-backed Queries attack 62

9.4.4 Oracle Piggy-backed Queries attack results 62

9.4.5 MSQLS Piggy-backed Queries attack 62

9.4.6 MSQLS Piggy-backed Queries attack results 62

9.5 Blind Injection 63

9.5.1 MySQL Blind Injection attack 63

9.5.2 MySQL Blind Injection attack results 63

9.5.3 Oracle Blind Injection attack 63

9.5.4 Oracle Blind Injection attack results 64

9.5.5 MSQLS Blind Injection attack 64

9.5.6 MSQLS Blind Injection attack results 65

9.6 Timing Attack 65

9.6.1 MySQL Timing Attack 65

9.6.2 MySQL Timing Attack results 65

9.6.3 Oracle Timing Attack 65

9.6.4 Oracle Timing Attack results 66

9.6.5 MSQLS Timing Attack 66

9.6.6 MSQLS Timing Attack results 66

9.7 Alternate Encodings 66

9.7.1 MySQL Alternate Encodings Attack 66

9.7.2 MySQL Alternate Encodings Attack results 67

9.7.3 Oracle Alternate Encodings Attack 67

9.7.4 Oracle Alternate Encodings Attack results 67

9.7.5 MSQLS Alternate Encodings Attack 67

9.7.6 MSQLS Alternate Encodings Attack results 67

9.8 SQLIA results 68

10 Appendix 5 – Artefact - Burp Suite testing 69

10.1 Login request sent 69

10.2 Request detection 69

10.3 Input shown 69

10.4 SQLIA intercepted 70

10.5 MySQL SQLIA blocked 70

10.6 Oracle SQLIA blocked 71

10.7 MSQLS SQLIA blocked 72

11 Appendix 6 – Artefact - Metasploit testing 73

11.1 Metasploit testing options 73

11.2 Web app test configuration 73

11.3 Metasploit testing results 73

11.4 Metasploit SQLI vulnerabilities detected 74

12 Appendix 7 – Artefact - Microsoft SQL server application 75

12.1 msqls\_home.html 75

12.2 msqls\_check.php 76

12.3 msqls\_yes.php 77

12.4 msqls\_app.php 78

13 Appendix 8 – Artefact - MySQL application 80

13.1 mysql\_home.html 80

13.2 mysql\_check.php 81

13.3 mysql\_yes.php 82

13.4 mysql\_app.php 82

14 Appendix 8 – Artefact - Oracle application 84

14.1 oracle\_home.html 84

14.2 oracle\_check.php 85

14.3 oracle\_yes.php 86

14.4 oracle\_app.php 86

# Introduction

## Background and Context

In recent years information security has become more and more of a popular topic of research due to the growing nature of the internet as more and more individuals gain access to it and as such the number of cyber-attacks on businesses and the confidential customer data that they hold has increased as a result. Therefore there has been a fair amount of research into stopping these attacks specifically at the front end of applications that businesses are using to interact with customers however this does leave the back end of the application vulnerable if attackers are able to reach it which is a part of much of the current research in information security, how to deal with the application vulnerabilities that attackers are taking advantage of. One of the bigger vulnerabilities is SQL injection attack (SQLIA) which is a technique where the attacker attempts to alter a SQL query by injecting SQL code into front end applications for example with an empty text box that allows the malicious individual access to the database in order to retrieve sensitive information, alter the database or even wipe the database entirely.

SQLIAs come in several different forms, one of which centres around the SQL command UNION that combines queries, if used for malicious purposes this command will alter the original intended query by merging it with a query devised by the hacker to gain access to more of the database, steal data or damage the database if the correct data types for the columns involved are selected. Another type of SQLIA attempts to bypass any efforts by developers to stop SQLIAs within their code or with third party tools, this is done by altering the malicious query to instead use alternate encoding such as hexadecimal or Unicode which is not a part of developers considerations when it comes to securing their application and database allowing the attacker to perform malicious acts. Another type of SQLIA takes advantage of any leftover sets of SQL commands packaged together as a procedure that came with the particular database management system (DBMS) that a developer may have chosen to support their application, the attacker will attach their malicious SQL statement to a procedure so that when run their statement will allow them to perform malicious acts, a SQLIA similar to this is where the attacker attaches their malicious SQL statement to another non malicious SQL statement by adding a semicolon at the end so that it ends it, this will cause the first statement to be run but also allow the attackers malicious statement to be run. Another SQLIA is where the attacker does not get any prompt from the application that their efforts are being successful in the form of error messages, the attacker instead injects SQL statements that are correct or incorrect and the application will determine this and based on the answer the attacker will learn more about the databases stored data than was intended. Another SQLIA makes use of time centric operators such as BENCHMARK() in the malicious SQL statement as if this is set to repeat a high enough number of times and returns correctly the statement will run and hamper server performance severely as the malicious statement is being run. Another SQLIA makes use of malicious SQL statements that are made purposefully incorrect by the attacker which when injected and run will cause error messages to appear allowing the attacker to learn the structure of the applications database as well as individual records. Another SQLIA makes use of SQL statements that are purposefully set up to be considered correct by the application they are injected into allowing the attacker access to more of the database than they should leading to stolen data and damages.

Based on the research performed on this topic several solutions have been devised to help secure applications against this kind of attack, there are tools that will scan an application in its entirety to look for known vulnerabilities in the code and will depending on the tool, make the user aware of the issue, provide assistance in rectifying the problem code or change the code in question to no longer be vulnerable to SQL injection attack on an automatic basis, while this is very useful for securing applications against known threats it is however ineffective against new SQLIAs that it does not have a solution for making it not very adaptable in the long term.

Another solution for dealing with the threat of SQLIAs is SQLIA detection tools; this is where the tool will run in the background of the normal application runtime and based on the tools method will scrutinize the SQL code that is passed from the front end application to the back end database against known SQLIAs and will block any deemed to be a threat automatically, the tools available do this in various ways and some are even quite adaptable in that they will attempt learn the traits of SQLIAs and be able to block new SQLIAs based on this, however due to the fact that the tools will run in the background of normal operations depending on the tool it may cause a noticeable slowdown in performance.

Another solution for securing against SQLIAs would be to introduce a third party encryption tool that will take the records in the database and encrypt them in such a way so that even if a SQLIA is successful the attacker will only receive data that is essentially useless to them without the password to decrypt them an effective method that has seen much use as a part of standard DBMSs to deal with other types of database threats such as malicious employees for example. Depending on the tool they can offer added benefits such as being easier to implement as well as fine tuning the encryption to what is required whether it be a high level of encryption for added security or light encryption for security with less of a performance cost added to it, as well as other added benefits that such tools bring with them, however depending on the amount of encryption and how widespread its use on the database there will be a performance cost as well as the time needed to decrypt the data whenever it is needed.

Another solution for dealing with the issue of SQLIA would be to use a web framework for a web application, this is a pre-built skeleton that will make up an applications structure, the advantage of using this is that those frameworks made for this issue will have built in methods for SQLIA detection and prevention before the application is constructed, as such it is an efficient defence against this attack however due to their nature it must be considered at the design stage of an application rather than after.

There are two types of analysis when it comes to SQLIA detection, static and dynamic, static analysis looks at the SQL queries that are passed at runtime in order to stop an attack, it does this by only allowing SQL queries to be passed if they do not match its fixed set of SQL values that are known to be used in SQLIAs, this effectively detects and prevents the attack however due to the methods use of fixed values it cannot stop attacks that do not conform to those said values without manually changing the application and its stored values. Dynamic analysis takes this further by learning from the responses web applications give by directing input to the application in every way possible in order to detect the vulnerabilities that lie in an application based on what the application sends back, therefore no changes are necessary as SQLIA protection will be automatic with no manual input required. Using these two analysis methods together means that both the SQL queries and the web application are analysed, thus taking both advantages that they have with at runtime SQLIA detection as well as dynamic vulnerability testing this method can be seen as a 'best of both worlds' approach.

Another method for detecting SQLIAs is by adding randomized characters to a SQL query that is being passed in a web application, it is possible to tell if it is malicious or not, this is known as instruction set randomization. Another type of dynamic analysis, compares a web applications SQL queries to those that are created when the application is running in order to detect SQLIAs, this is known as SQL query profiling. Another method which is similar to dynamic analysis, is machine learning which is where the SQL queries an application generated are learned from in order to create how the method detects SQLIAs, this does rely on good data to learn from but it has potential to be very useful as a defence against SQLIAs.

## [Scope](http://www.cs.stir.ac.uk/~kjt/research/conformed.html) and Objectives

Based on this it is clear to see that much has been said and done about the issue of SQLIA, including research on the various methods attackers use to gain access to what should be secure data as well as the solutions that have been devised in an attempt to secure against this and ultimately end the threat that it poses, however despite this there is no all-inclusive method for dealing with SQLIA as well as the disadvantages some of the current methods have when implemented.

Therefore this paper will look further at the current SQLIA methods in use by attackers, the solutions for dealing with them and from this test against several DBMSs whether SQLIA is still an issue. After performing this extensive research into SQLIAs and the solutions to that threat there will be several tests on whether SQLIA is still an issue. This will involve the development of test applications that perform specific jobs possibly using several different programming languages such as PHP, HTML, etc in order to test their resilience against this threat.

They will be connected to several databases using different Database Management systems(DBMS) such as Oracle, MySQL, Microsoft SQL Server etc with a substantial amount of test data to test their built in methods for dealing with various SQLIA methods and then move on to testing with the addition of third party tools for SQLIA defence that were found based on the research performed beforehand. With the information gained from this artefact there will be a report produced on the research and the findings combined in order to provide an answer on whether SQL Injection is still an issue.

## Overview of Dissertation

In this paper is included an abstract of the entire paper summarising the problem of SQLIA causing applications to no longer be as secure as they once were, the objectives of this paper and how this paper attempts to accomplish those objectives and help solve the problem. Next this paper acknowledges those who have helped in the creation of this paper, a table of contents that includes every section of this paper, a list of the figures used to demonstrate the points of this paper in picture form, the introduction including background and context, scope and objectives and an overview of the dissertation. Next is the research methods used, the findings of this paper, a critical self-evaluation, conclusions in the form of a summary, evaluation and any future work based on this paper, a reference list and appendixes.

# Literature Review

SQLIA is a problem as it is a type of attack that goes through a front end to gain access to the database directly, making most front end security ineffective in order to perform malicious actions, this is outlined by the work of Shar, L (2013) who takes a broad approach to educating its reader on this issue concerning the weaknesses that exaggerate the issue such as insecure coding practices as well as the defences employed in response to SQLIA such as SQLIA prevention at runtime, third party tools and defensive coding. This is similar to the work of Dorai, R (2011) as both deal with the issue of database security for example the first source provides details on SQLIA detection software/tools as well as the different features that may be included such as whether vulnerability locating is automatic or manual, tips to modifying code for the better and more which is very useful as the tools it shows may be used in the practical section of this paper, meanwhile the second source discusses how the database is vulnerable to this attack such as with escape characters, types inappropriately dealt with or the database server itself which may become part of the testing of the practical section of this paper, the source also shows several tools for the detection and prevention of SQL Injection such as GreenSQL an open source software for this purpose, due to it being open source it has the advantage in terms of cost over paid software however due to this it will undoubtedly be less effective than the paid software such as perhaps some of the tools outlined in the first source that will impose less of an overhead to the systems overall performance at runtime with query and transaction times, therefore this is an important point to be considered when designing the artefact and so will help answer the research question.

SQLIA is a large issue due to its effectiveness and its popularity as an attacking method by malicious individuals as shown by the work of Dorai, R (2011) who discusses SQL Injection but also shows how this relates to database security as a whole by detailing the large amount of attacks on websites that were reported in a certain time frame in order to outline the gravity of the issue. The source discusses several SQLIA methods such as those that take advantage of types and escape characters being handled incorrectly in order to gain information about the system or the records it holds, the database server itself as well as intentionally causing error messages to appear with the injection of SQL code in order to take advantage of this and again access to information they normally would be restricted to this is useful as this may be used as part of a testing suite on the finished artefact for this paper. The source also outlines methods for the prevention of SQLIAs which is similar to the work of Kim, M (2014) both discuss SQL Injection and the complexities the topic brings with it such as how the application is coded and if there are any vulnerabilities that attackers can take advantage of such as error, type mishandling and more. The first source specialises in prevention of SQLIAs with its tool GreenSQL that is essentially a firewall for the database that attempts to block malicious SQL statements which is an improvement on simply detecting attacks however this may cause runtime issues such as queries that are not malicious in nature being blocked impeding a company's productivity as well as the overhead such a tool will impose on the queries, transactions or any action on the database. Meanwhile the second source outlines its design for detecting SQLIAs based on data mining that is broken up into several different phases such as the data collection, pre-processing, training with the use of support vector machines(SVMs) which is a machine learning technique where the tool will learn from the data it is given in order to better perform its task which is the last phase of actual detection of SQLIAs, which is very useful considering the numerous malicious statements that must be learnt an automatic detection system will drastically increase the amount of attacks that are known of and in so doing improve security however this will still be after the attack has taken place instead of stopping it which is why prevention is more useful however both will no doubt impact the design and testing of this papers practical section and in so doing help answer the research question.

SQLIA is a complex issue due to the amount of permutations in the technique for attackers to take advantage of as well as the differences in design for the solutions devised to defend against it as shown by the work of Kim, M (2014) who discusses SQL Injection and the details of its design for detecting SQLIAs which it shows to be sectioned into several phases that add up to a tool that provides the user with the ability of sensing attacks of this kind in order to learn from them and take steps to secure their system against it. The phases are the collection of data, its pre-processing, and then the training of the tool via support vector machines(SVMs) which is a part of machine learning where the tool is given key data that acts as an anchor for the tool to draw on so that further data given to it is analysed correctly in order to perform its task so in a sense the tool will learn and improve over time without major alterations to its structure, coding etc and from that detect SQLIAs. This is similar to the work of Shar, L (2013) as both sources deal with the issue of SQL Injection by detailing tools to help secure the database against this, the first source focuses on detection which while this is important as it is necessary to learn the vulnerabilities in a system it is not perfect as damage can still be done in order to learn of the problems with a system meanwhile the second source also shows tools for detection but also tools that show how to improve coding in the system to make it less vulnerable as well as tools that take this further and attempt to prevent SQLIAs entirely which if implemented properly would greatly improve database security and as such will be a basis for further research in this paper as well as the possible implementation of one or more tools to test against the artefact however the issue with these is that they are to be used at runtime and may affect performance and come after all the design considerations have been taken and implemented and therefore a better solution may be a hybrid of detection and prevention which will also be a focus of this paper and should help answer the research question.

Many solutions to SQLIA have been developed most notably in the form of third party tools that attempt to defend against this threat at runtime as shown by the work of Pinzón, C (2013) discusses SQL Injection but also provides its solution as an example of how to safeguard against this kind of attack by first detecting it and then prevent similar occurrences from happening with its proposed hybrid tool that comprises a classifier agent using as case based reasoning engine similar in nature to machine learning where the tool learns from past problems which in the case of SQLIAs would be a formidable ally to have due to the myriad methods available for attackers having a tool to automatically learn from attacks will drastically increase security. The tool also includes a visual agent in order to physically see suspicious queries and determine if they are malicious in nature or not which is useful as it allows for the human eye to be a factor in the defence of the database as software can always make mistakes that the human eye will be able to see however this does mean that a company using this tool will need to employ personnel to perform this task which decreases possible performance and its cost effectiveness of an automatic system. This is similar to the work of Kim, M (2014) as both provide solutions to SQLIAs but their methods and the techniques involved are different as while the first source has its hybrid tool of detection and prevention the second source focuses solely on detection and also uses SVMs instead of cased based reasoning which due to its nature if implemented properly with the right data for the task may outperform cased based reasoning in terms of precision due to the amount of data that can be provided for it giving the second sources tool a strong platform to learn and deal with the issue of SQLIAs and therefore both sources tools will be considered when designing the artefact and may play a part in the practical stage of this paper in order to answer the research question.

Some authors make use of several techniques for their solution to SQLIA such as in the work of Lee, I (2012) who discusses SQL Injection but in particular it shows its method for SQLIA detection, which attempts to simplify the detection process of such attacks by taking the SQL query out of whatever is submitted to a particular application via dynamic analysis and then the tool then makes a decision via Static analysis against pre-determined SQL queries to see if it is malicious or not and take necessary methods to ensure it does not cause damage to the database. However there are still flaws in this method as although due to the combined approach its detection rate is notably higher than related methods as shown in the paper, it is not capable of handling all SQLIAs meaning a system implementing this method would still be vulnerable to attack. This is because the method relies upon automatically taking user input and analysing it against known values however these known values can quickly become outdated especially with the diverse nature of SQLIA, new methods of attack would drastically reduce this solutions defensive capability and attempting to manually update those known values each time a new type of SQLIA is discovered is simply impractical therefore this method works well against SQLIA in the short term but will most likely be phased out in the long term. This is similar to the work of Pinzón, C (2013) as both sources cover SQLIA and their proposed solutions make use of several technologies or are hybrid in nature but take different approaches to dealing with SQLIA as Pinzón's approach seems to be a more permanent solution with its cased based reasoning engine and machine learning meaning the tool will learn as time goes on about new and different types of attack in order to provide up to date protection and due to it being a third party tool it should be relatively unproblematic to implement however based on these advanced techniques it may be a costly solution compared to Lee's solution which is simpler in scope providing base level protection by detecting SQLIAs at runtime by taking user input where the attacks originate and comparing this against a set amount of known SQLIA values, this method in comparison is the simpler option and as such will undoubtedly be cheaper to implement however this is a method that must be included in the systems design or alter the current design to allow for it, which makes this method harder to implement due to its intrinsic nature which will be taken into account as part of the artefact in order to answer the research question.

Some Authors make use of the analysis techniques such as the work of Cho, Y (2015) who discusses the impact of internet technology on people's lives specifically in relation to their privacy and security being under threat from attackers using techniques such as but not limited to SQLIA as well as the methods employed to thwart them. The paper debuts its method for demonstrating the industries lack of readiness to protect their customers data by testing a select number of academic websites for vulnerabilities. The papers method first uses dynamic analysis to find potentially vulnerable websites via keyword search and then looks specifically at the vulnerabilities the website has via static analysis, as part of the testing the paper used several methods attackers use such as SQL injection, web crawling etc to better be able to test the websites defences. Because of this extensive testing the paper is able to demonstrate in a practical way the industries lack of preparedness against these kinds of attacks at least in small scale, however due to the use of these analysis methods there are flaws that prevent this vulnerability detection tool from being all-inclusive which is similar to the work of Lee, I (2012) who also makes use of these techniques, whereby SQLIAs are detected using dynamic and static analysis which allows for automatic detection of the object in question and then making a decision whether said object is malicious based on pre-determined values. This means that in the short term both tools will be able to perform their duties in the role of vulnerability detection and as such will help in the strengthening of industry wide web security however, in the long term the known values used to make a decision on whether something is malicious will become outdated where more and more attacks will be created that are not a part of the systems capabilities to track, therefore this requires constant research on the attack methods used and manually changing the list of known values the system relies upon which will take time and effort by staff that will be needed to be employed for this task, therefore this is inefficient and costly in the long term.

In order to deal with the threat of SQLIA, some authors prefer a more tried and tested approach, while not specifically targeted at SQLIA, the work of Yu, *et al* 2013 shows the use of encryption in a medical based web application and database. The papers method is divided into three parts a secure login system, server access via secure sockets layer (SSL) which uses encryption to securely allow access to the server. As well as the encryption of important data using an algorithm known as the advanced encryption standard (AES), this is useful as even if a SQLIA is successful in bypassing the first two barriers and retrieves data, it will still be encrypted and therefore useless to the attacker without the decryption key needed to decipher the data which keeps it protected, however using this method does not completely protect the system itself from harm as instead of stealing data an attacker may choose to damage the database by deleting all of the data as well as if the decryption key is stored in the database itself then that itself may be stolen making this defence meaningless if the designers choose to set up the system in such a poor manner. This sort of defence is best considered early on in the design stage as although it helps protect the database it will undoubtedly cost an amount of performance that the system would have otherwise had which is similar to the work of Dorai, R (2011) who deals more with the issue of SQLIA and how to defend against it as well as showcasing several tools that can be implemented for this such as GreenSQL which due to its open source nature will also incur the same performance costing effect as encryption on the database, however as it is a third party tool it is much easier to implement and being open source software it is a very cost effective method of SQLIA defence therefore both of these methods will be considered in the artefact as they could be used in tandem to provide even greater protection and help answer the research question with the results.

Some authors deal with the issue of querying data that was encrypted in order to protect it such as the work of HWEEHWA, P 2014 who discusses the problem of insecure IT outsourcing and provides its solution is a design for an encrypted database with the express purpose of performing join queries on third party held data as well as algorithms that speed up the query process. This is useful as not only is the data protected but also there are performance gains that can be made using this method that otherwise like many other methods that use encryption simply would not have, the paper goes further in its explanation of the algorithms to show how it improves performance by creating equivalence classes which contains knowledge of all the records in a table so that when it comes to join those tables the system only has to check if one part of both tables are the same instead of their entirety which severely reduces query processing time as a result. However this method is best included in the initial design stage of a system as it is not a third party tool that is not easily implemented as an add on to the main system, implementing after the system is live would be difficult due to it affecting the core mechanics of said system which is similar to the work of Yu, et al 2013 with both sources showcasing methods that include the use of encryption techniques that apply to the core systems mechanics however the second source deals with securing the entire system using encryption while the first source focuses on encryption on the database side as well as providing a means to dampen the performance costs these methods inter. Therefore both sources will be considered in the artefact for their encryption techniques especially the performance increasing algorithms of the first source as comparison against an unencrypted database without these algorithms, in order to help answer the research question.

With the advances in technology more services become available to us, however at the same time the threats to our privacy and security also increases such as with cloud computing as shown by the work of Wei, et al 2013 who discusses the issue of protecting data that is stored in the cloud with the threats that come with using this technology such as shared access to data leaving the system vulnerable to malicious users, malicious employees, users lacking control over their data, data unavailable if the service becomes unavailable. As well as several possible solutions such as auditing the data stored so that any changes can be tracked and who made them, encrypting the data so that even if malicious users gain access to it, they will be unable to make use of it without the decryption key. As such the paper lists encryption algorithms created with cloud computing in mind as well as its advice on the strengths such algorithms should strive for, which is similar to the work of HWEEHWA, P 2014 who also deals with the issue of data security and provides its own encryption method as a solution which goes further by including algorithms that improve query performance meanwhile the first source only shows the necessary aspects of an encryption algorithm as well as data auditing techniques that are useful but do not actively prevent SQLIA, therefore the second sources algorithms will be of more use especially in the artefact design stage, however the information from the first source will still be useful enough to help answer the research question.

Some authors focus on the protection of the most sensitive of data held in databases with the use of encryption algorithms such as the work of Boicea, *et al* 2010 who discusses how to stop attackers from making sense of sensitive data by encrypting it instead of the entire database as this significantly reduces the performance costs that such an action would cause as well as the use of machines that’s sole purpose is the encryption of this sensitive data instead of traditional built in DBMS encryption in order to further reduce the performance costs. The paper goes further to show the strengths and weaknesses of encryption within and without the DBMS so as to show when best to use either solution which is similar to the work of Wei, et al 2013 who also discusses the use of encryption of data and various techniques but specific to data stored in the cloud as well as showcasing data auditing techniques for the cloud which is useful as the data that may be stolen via SQLIA will still be protected and there will be records of the attack meanwhile the first source focuses on encryption and aims to protect the most sensitive data only for its defence as well as a way of saving performance costs, it also includes algorithms for this very purpose. Therefore both sources will provide useful data when implemented in the artefact stage as it will be important to compare how this helps protect the data as well as the performance aspects in order to help answer the research question.

Some authors deal with the issue of how to encrypt data stored in the cloud in such a way that the owner of the data has the decryption key rather than the service provider so that owners have more power over their data and are protected against malicious employees. The paper goes further to show how CryptDB attempts to solve this issue by having the data be encrypted at cell level so that when queried the database does not need to be entirely decrypted only the data involved as well as testing to ensure the results are the same as the cipher created when the algorithm encrypts the data which allows querying on the encrypted data without the need of an decryption key, this is useful as it increases security and performance for cloud services as well as empowering the users of said services. The paper goes further to show how the method chooses the correct encryption for the different attributes in the database which is similar to the work of Boicea, *et al* 2010 who also discusses encryption in the database but only on the most sensitive of data in order to ensure it is protected as well as to save on the performance costs of encrypting the entirety of the database which is an advantage over the first source as although the entirety of the data is more secure than it was without encryption it will undoubtedly affect query run-times, however the advantage this first source has over the second is that the users have the decryption key which empowers them, gives them confidence in their data’s security and stops malicious insiders using it against them. Therefore both sources have positives and negatives which will be taken into account for the artefact design stage in order to test out these factors and see how they help answer the research question with the results.

In order to deal with the issue of SQLIA it is important to be able to detect that they have or are occurring, such as with the work of Roy, 2011 who outlines its method for discovering the weaknesses in web applications that attackers using SQLI are taking advantage of. This is achieved by its SQLI vulnerability scanner which as an external tool makes integration into the main system much easier as there the core functionality of the system does not need to be changed, the tool looks for vulnerable parts in the hypertext transfer protocol (HTTP), attempts to use this vulnerability to attack the website which if successful it makes the user aware of it, this is useful as it is a safe way to check a web applications security against this kind of attack and make sure it is not easy for a successful SQLIA. The paper also shows its framework for SQLIA prevention which is tested against its vulnerability scanner which is similar to the work of Lee, I (2012) who also discusses how to detect SQLIAs but takes a different combined approach using dynamic and static analysis which is useful in that it provides the advantages of both techniques to the method however it still runs into the problem of using set values to detect SQLIAs which will undoubtedly become obsolete over time given the varying nature of SQLI and as such the security it provides will degrade unless it is maintained which adds further cost to what should be an automatic system. Meanwhile the first source with its vulnerability scanner and web framework while similar in that they both test websites for SQLI vulnerabilities and affect the core mechanics of how any web application works they are different in that they focus on the web page itself rather than set values by analysing the URL of the website as well as web crawling the entire website to check for vulnerabilities which means coupled with a framework designed for SQLIA defence they are better suited for dealing with this issue at least in the long term, nonetheless the techniques involved will be tested against the artefact and possibly be included in the design stage the results of which will help answer the research question.

There are numerous ways of detecting SQLIAs all with their own advantages and disadvantages such as in the work of Kerner, S 2013 who discusses the need for a detection tool with previous methods becoming ineffective. The tool proposed is designed to learn at is goes along in order to improve over time, it is external to the web application being scanned and does not slow it down due to it simply learning from the data flowing around the web application as well as SQL statements to see how they could have been changed to be malicious. This is useful as it allows for users of the tool to see the malicious SQL statements as they occur and be able to take steps in order to stop them impacting on the privacy and security of data store which is similar to the work of Roy, 2011 who also attempts to deal with SQLIA via its own detection tool which is different in that instead of looking at all the information that is created at run time and learns from it over time, it analyses the http URL as well as web crawls the application for SQLIAs but is similar in that it does not affect application performance due to its external nature however due to it being coupled with a web framework it becomes the harder to implement option as this must be considered in the design stage rather than modifying a fully designed and built application due to it affecting core mechanics, nonetheless this is still a powerful solution to the threat of SQLIA although it may not be as long term as the first source’s solution due to its learning nature which will play a part in the artefact design and construction stage in order to help answer the research question.

Some authors choose to provide a detailed description and analysis of the current situation when it comes to database security and SQLIA such as the work of Kindy, D 2013 who discusses the issue of SQLIA, the weaknesses of web applications that allow this attack to be so threatening, the various SQLIA methods as well as the defences that can be employed to deal with this threat, which is useful in that it provides the reader with a broad knowledge base on the topic up to that point in time so as to keep the industry informed on the problem and enhance its overall security as a result. The SQLIA attack methods discussed are quite diverse such as a type of attack that takes advantage of any commenting in the SQL code, another that uses triggers to provide the attacker with unauthorised information on the database, also the defences make use of both detection and prevention techniques such as a framework known as SQL DOM that aims to solve any blockages etc in the calls made to the database via creating the SQL statements with object manipulation, another takes advantage of database procedures checking the SQL input. Therefore the paper provides a wealth of knowledge similar to the work of Dorai, R (2011) who also deals with the issue of SQLIA but focuses on the defences that can be employed with the features involved and as such it will provide useful insight into the defences to implement for testing in the artefact stage, as well as changes to developers coding that will help leave applications in a less vulnerable state while the first source provides this and information on the types of SQLIAs which is useful in that they show how attackers are taking advantage of weaknesses in applications and as such pave the way to eliminating them, all of which will impact the design stage of the artefact and be useful for comparison with its results in order to help answer the research question.

Some authors attempt to not only provide solutions in the form of external tools but also to help change developers coding practices to be less vulnerable to SQLIA in the form of frameworks such as the work of Salama, *et al* 2012 who discusses their framework in how it uses detection techniques that look for malicious or different from the normal input expected. The method is able to do this by building up a representation of the normal database queries that cause the data flow between it and the web application; this is then used as comparison against any SQL query in question to determine if it is malicious in nature or not, this captures key commands in SQL queries that can be used to attack the database and as such are treated as suspicious characteristic’s to be scrutinized. This is useful as it provides a means of detecting SQLIA similar to the work of Kindy, D 2013 who also deals with SQLIA solutions but rather than provide its own solution it showcases several solutions in order to see how they compare as well as the different approaches to the issue available while the first source does provide its own solution which as a framework is quite powerful in that the entire application has been designed with SQLIA in mind making it less vulnerable, however unlike some of the external tools discussed by the second source a framework must be considered early on when creating a web application in order to take full advantage of it, making it the harder to implement option as a result, the second source also discusses the types of SQLIAs providing the reader with a greater understanding of why they are successful which is quite useful when trying to defend against it, therefore both sources provide different but useful options for dealing with SQLIA and can be tested against in the artefact stage in order to scrutinize their effectiveness and help answer the research question.

Some authors focus on the attackers point of view such as the work of Štampar, M 2013 who discusses a different type of SQLIA that makes use of the domain name system (DNS) coupled with its malicious SQLIA query in order to gain access to a database. The paper demonstrates this with the use of the tool sqlmap which is designed to perform this kind of attack by setting itself up as a fake server which intercepts any DNS requests by responding to them in order to perform an SQLI exfiltration attack carried out through the HTTP request with the intention of grabbing data from the target application. This detailed and practical example of a type of SQLIA is similar to the work of Kindy, D 2013 who also discusses the SQLI methods used by attackers but in a more general fashion as a part of its showcase of the current attack methods, defences and research in the area which allows the reader to see the topic as a whole in contrast to the first source which focuses on one part of the topic but as a result is able to provide much more detailed analysis and enables greater understanding of the threat so as to limit its impact on industry coupled with the further research outlined in the paper, therefore both sources have a part to play in solving the issue of SQLIA but in different ways and as such will help answer the research question with the knowledge gained.

Some authors educate their readers on the types of SQLIAs before moving on to show their solution for dealing with them such as the work of Asha, 2012 who discusses several types of SQLIA such as those that add malicious queries onto normal queries and those that attempt to inject code with a where clause whereby the database sees it as correct and outputs data as a result. The paper goes further by showing its solution that attempts to prevent SQLIA via strengthening the login system of web applications so that only authorised users can access the database, this is done in two parts, firstly the SQL code in the HTTP request is checked independent of the database so even if it is malicious it will not impact the database, then the request is checked via XML to ensure the correct password was used so that even if the first part is seen to be correct the second part will still fail and ensures a secure login system that protects the database. Due to this independency of the database the security is increased with full prevention of SQLIA which is similar to the work of Salama, *et al* 2012 but instead of prevention this source opts for detection via its web framework that learns the typical nature of the database and application which allows it to discern any malicious queries, as a framework it is harder to implement than the first sources method which affects the login security of an application however with its dual authentication it is powerful in stopping SQLIAs known to attack in ways that are caught with this method but as a result it will cause noticeable slowdowns of a systems performance especially with the database being made independent of the login process until after a request is deemed safe, therefore both methods will impact the artefact design stage in order to provide tests, the results of this will help answer the research question.

Some authors focus on both aspects of SQLIA defence such as the work of Dogbe, 2013 who discusses SQLIA detection and prevention methods with their good and bad points as well as their own solution which aims to take in the best aspects of said methods without any of the drawbacks. The paper goes into detail on its method which it shows to have two defences built in, the first uses static and dynamic analysis whereby when a query is received it is analysed by first learning from it to add to the queries the system has knowledge of which it then uses as comparison to determine if a query is malicious or not through static analysis which is important as this means that it solves the disadvantage of static analysis becoming unusable long term when new SQLIA methods are developed. The second defence, fine grained Role Based Access Control (RBAC) which is where users of an application are given roles which have certain privileges to perform only the actions necessary to complete the tasks for that role, therefore if an attacker breaches the first defence they will have very limited access to the database and any actions that go beyond what the user is allowed to do will be blocked. This dual protection method is similar to the work of Asha, 2012 who deals with the security of the login system of applications and uses both SQL and XML authentication to ensure attackers cannot gain unauthorized access to the database, however due to the added security and the database being made independent of this process there will be a drop in performance which is less of an issue with the first sources method as it performs its authentication with the database still a part of the process and its second part does not hamper performance, therefore both solutions are useful with their advantages and disadvantages but will be tested to determine their effectiveness in the artefact stage to help answer the research question.

Educating the IT industry of the dangers of SQLIA is another way of preventing this kind of attacks successfulness as the more the industry is informed and starts taking action against it the better equipped they will be to deal with it such as in the work of Patel, 2011 who discusses the types of SQLIAs as well as the methods for solving this issue. One popular attack uses where clauses to create a statement that returns correct causing the database to release data to the attacker while another modifies a normal SQL statement to be malicious in nature, one of the defences discussed is instead of getting input from an open text box which an attacker can inject SQL into, a check box, drop down list etc where choices of input were predetermined thereby eliminating that vulnerability, another validates any input especially string inputs to ensure they are not malicious. This education of SQLIA is similar to the work of Dogbe, 2013 who goes further by showcasing its own solution that uses dynamic and static analysis to analyse SQL statements for malicious intent and solves the issue of static analysis degrading over time by having dynamic analysis strengthen it so that the method learns as it goes on, the method also implements RBAC that ensures that should an attacker gain access they will have very little power to perform malicious actions, therefore both sources provide a knowledge base for their readers with the first source opting for a description of both sides of the topic for a fuller picture of the situation while the second source takes a more practical approach with its own method, the techniques involved and the results in order to show in a more developed fashion how to deal with SQLIA which will help answer the research question by impacting the artefacts design.

Some authors when educating their readers on the topic of SQLIA take it upon themselves to perform a deeper analysis of the situation such as the work of Tajpour, 2012 who shows several SQLIA vulnerabilities such as weak variables, unlimited size of variables, weak database privileges, input that is not checked for malicious SQL statements etc as some of the common failings of web applications it is important to ensure developers are aware of these failings in order to rectify them to increase data security. Some of the attack methods discussed use UNION or WHERE clauses etc to gain access to the database and others make use of the applications own queries to either modify them into malicious statements or add the attackers own query to the normal statement, this is useful as it shows not only how attackers are successful but also provides an inkling into how to defend against them. The defences are both detection and prevention with some analysing the vulnerabilities in a web application and others that detect SQLIA by comparing queries against known safe statements etc. Similarly Patel, 2011also showcases SQLI attack and defences although considerably less but goes into more detail in their function while the first source lists more as well as the vulnerabilities in web applications and then compares them for their performance in successfully attacking and defending web applications which is very useful for calculating a methods successfulness without implementing it, therefore the first source while it does not have quite the range of attack and defensive methods as the second source it does go into more detail on them and will help in the design of the artefact for the better by influencing the techniques chosen for the better so as to better answer the research question.

Some authors focus on SQLIA prevention rather than detection such as the work of Ahmad, 2011 who showcases its solution which filters malicious SQL statements to prevent an attack from being successful. This is done by ensuring if the login input is SQL code then the login will fail and the encryption of username and password adds another layer of security, the login interface uses open text boxes, but the paper does highlight further techniques that if implemented would enhance security such as the use of white list input validation and strict database privileges for users which is similar to the work of Tajpour, 2012 who also shows how database privileges should be strict so that if a malicious employee or an outside attacker successfully logs in they will not be able to damage the database irreparably as well as other SQLIA vulnerabilities, methods and defences to provide information on the current situation on the topic which in comparison to the first source is quite general while the first source goes into detail on its own solution which is useful for testing purposes however it uses open text boxes which allows attackers to attempt SQLIAs on the application through this, the first source is also quite limited in scope as it focuses only on the techniques it has chosen to implement and it is not immediately apparent whether this is truly as effective as the paper claims unlike in the second source which tests numerous SQLIA methods and defences for their effectiveness and compares them, therefore both sources have their ups and downsides and it will be useful to take the best of both these sources and use them to improve the artefacts design that will help answer the research question.

Some authors focus on securing specific to applications built in a particular language such as the work of Jang, Y 2014 who discusses the problem of SQLIA and demonstrates how its method aims to solve that. The paper goes further to show its method which substitutes all variables in an application to ensure that a malicious SQL statement cannot be successful as the statement will be attempting to take advantage of variables that have since been substituted out, this useful as it means attackers cannot take advantage of their knowledge of how typical web applications are built for a successful SQLIA and this is targeted at Java based applications. This reworking of how an application is built for the sake of SQLI defence is similar to the work of Salama, *et al* 2012 as both changes the core mechanics of an application especially in terms of how it interacts with the database with the second source providing SQLIA detection via comparing SQL statements against the perceived nature of the data flow between application and database that has been built up over the life span of the system, the advantage of both these methods are that they are quite powerful as the systems have been built with SQLIA in mind rather than after they have been built however the disadvantage of this is that it is a harder to implement option especially for those who wish to protect their current applications and cannot implement this without lengthy downtimes which they may not be able to afford. The issue with the first sources method is that unlike the second sources method it is target at Java applications only which limits its usefulness to those running applications using other languages, therefore both sources methods are similar in their setup but different in their function which will be interesting to compare ion the artefact stage in order to help answer the research question.

Some authors aim to improve current SQLI vulnerability scanners such as the work of Shar, L, Tan, H 2013 who shows how dangerous this issue is and proposes its solution that it describes as an alternative to existing vulnerability scanners. The paper goes further to show how the method is a framework comprising several techniques such as static code attributes that are based on methods designed to ensure no malicious input harms the database by learning about the vulnerabilities present over time and predicting them, next is data pre-processing and reduction that ensures the attributes used are unbiased and well suited for that particular application, next classifiers go through the data gained to more accurately create the models that predict vulnerabilities. This is useful as it allows for the prediction of web application vulnerabilities specifically SQLI focused over time as an application is running which is similar to the work of Tajpour, 2012 who showcases several of the SQLI vulnerabilities that the first source aims to eliminate as well as the attacking methods and SQLI defences but unlike the first source it does not provide its own solution but does allow the gauging of their effectiveness with its comparison of SQLIAs and defences before implementation unlike the first source which does go into detail on the techniques used, however it is difficult to estimate its effectiveness compared to other methods without implementation, therefore it will be important to include this in the artefact design considerations to test first hand its effectiveness and in so doing will help answer the research question.

Some authors aim to formalize how we detect SQLI vulnerabilities such as the work of Lei, 2013 with its test case generation model which comprises three parts, first its Global Test Rule (GTR) which creates the test cases used when analysing an application for SQLI vulnerabilities which is useful as it gives structure to the testing process. The second part is its SQL injection vulnerability test matrix (SQLTM) model which creates the GTRs, lastly is its Multiple Phases Detection Approach (MDPA) aims to improve on the last two parts by dynamically creating test cases and detecting vulnerabilities by instead of running all tests at once they are run one at a time so that the results of which are learnt from and are able to improve further test cases. This improvement of results by the program learning as time goes on is similar to the work of Dogbe, 2013 who uses dynamic and static analysis to analyse a query for malicious intent by comparing it against static values, however the method learns from each query it analyses with dynamic analysis by updating those static values thereby eliminating the problem of static analysis becoming outdated much in the same way that the first source learns from each test case that is run, this learning over time will be very useful if implemented in the artefacts design and its testing will provide interesting results to help answer the research question.

Some authors look at the design concepts of certain applications and attempt to alter them to help prevent SQLIA such as the work of Noiumkar, P 2012 who discusses how they altered the application to use certain techniques and how this improves SQLIA prevention. The paper goes further to show how the application 'IPTABLES' works like a firewall to block malicious input, to modify it to block SQLIA a rule is added whereby if a particular pattern is detected that is known to be used in malicious SQL statements such as 'WHERE 1 = 1' it is blocked and to stop false positives the method uses techniques such as cartesian product, rule repositioning and relation which it then shows to be effective by comparing with and without this. This improvement of applications for the sake of SQLIA prevention is similar to the work of Shar, L, Tan, H 2013 who improves SQLI vulnerability scanners by having them learn over time about the kinds of SQL statements that are malicious in nature rather than being stagnant and eventually becoming ineffective, this is useful as time and effort is not needed to be spent on updating the application over time to be up to date with the latest SQLIA methods, the advantage of the first source is that unlike the second it provides assurances in the form of results on its effectiveness which is important as without implementation it would be difficult to see how successful this method would be in a particular situation and so both sources will need to have their methods tested in order to see if they can help answer the research question.

Some authors focus their efforts on protecting particular types of systems against SQLIA such as the work of Abawajy, J 2013 who shows how as radio-frequency identification (RFID) tags have become more commonplace with more data stored on them, the threat attackers pose by using SQLIAs coupled with this technology to harm systems has also increased. Therefore the source debuts its method that aims to detect and prevent this, first it sets constraints on database attributes so that the data stored is specific to its use and cannot be easily changed without knowing the constraints and in so doing should an attacker attempt to inject SQL code to retrieve, modify or damage data the likeliness of them getting the correct input to be successful is significantly decreased which is an effective prevention method although it is an added burden on the system that must be cleared before implementation to ensure it does not conflict with normal operations. Next the method shows its SQLIA detection algorithm which sets the conditions that SQL statements must conform to such as the size so that should any query not meet these conditions it will be treated as malicious and blocked as attackers will not be aware of said conditions so any query that does not meet them is likely to be malicious in nature which is an effective detection and prevention method although once again this does constrict operational queries as well. This constriction of the allowed SQL statements is similar to the work of Noiumkar, P 2012 as both sources attempt to structure the systems they are protecting in such a way that only specific queries are allowed to prevent SQLIA, the second source does this with the use of a tool that acts as a firewall preventing suspect queries from harming the data which is much easier to implement than the first sources method that is more like a framework that should be considered when in the design phase rather than after as it would require a major reworking of said system, both provide results of their testing of their respective methods which is useful to show their effectiveness at a glance and so both have advantages and disadvantages which should be considered in the artefact design stage to see if the techniques used can be implemented to help answer the research question.

Some authors give detailed reviews of the SQLI defences such as the work of Johari, R 2012 who provides this as well as its recommendations for where further research should be focused. The paper goes in further detail to show why SQLIA is a threat and the methods employed to protect against it such as one that strengthens the login system of an application through encryption, another that uses static and dynamic analysis to detect SQLIA vulnerabilities as well as others all of which are reviewed with advantages and disadvantages and then finally concludes with failings seen in most of these methods. This review of numerous SQLIA detection and prevention methods similar to the work of Tajpour, 2012 who discusses SQLIA defences but also the attacking methods and compares them for their effectiveness as well as showcasing the vulnerabilities that attackers take advantage of while the first source reviews numerous defences albeit in greater detail, therefore both are useful for providing the reader with information on the topic with their ups and downs comparatively and so will help answer the research question with the knowledge gained.

Some authors aim to improve the detection of SQLIAs such as the work of Alserhani, 2011 who showcases its model for the prediction of an attack using a combination of both statistical and knowledge based model techniques. The paper goes further to show its method to be an improved version of another model that looks at SQLIAs as a series of stages that an attacker takes so as to spot these in the early stages of an attack in order to block it, this is improved with real time processing in order to show in real time the progress of attacks in graph form as well as graph algorithms that work online and offline in order to create a graph that can be dealt with. This improvement of current techniques is similar to the work of Shar, L, Tan, H 2013 who improves SQLIA vulnerability scanners by having the tool learn over time by analysing the normal and malicious SQL statements as the system is run which means the tool will not become quickly ineffective, both methods are much like a framework in that they affect the core mechanics of an application and so must be considered early on in an applications development life cycle to avoid difficulties, however the first source unlike the second source only shows its model and only mentions the tool(s) created as a result and so the results it shows which lacks graph form is harder to analyse to determine effectiveness, therefore extensive testing in the artefact stage will be needed for this so as to help answer the research question.

Some authors aim to find new ways to improve new technologies such as the work of Mittal, P 2013 who showcases their solution, a combination of two encryption techniques. The paper goes further to show the first encryption technique as Bitslice AES which allows for great efficiency due to the encryption process being handled in parallel by changing the encryption technique into logical bit operations and its second encryption technique that makes use of GPU processing through CUDA which is effective due to encryption being run in parallel with CUDA excelling at parallel processing allowing for substantial performance boosts. This is used to prevent SQLIA by encrypting the parts of the application that allow for open input such as in a text box, from the user such as the username and password which is a typical SQLIA vulnerability, as well as encrypting other inputs or actions from the user. This improvement of old technology with new techniques is similar to the work of Alserhani, 2011 who improves SQLIA detection with real time processing in order to see attack progress as they occur, while this is useful to highlight the attacks on a system it does not prevent them unlike the first source which improves encryption by removing its disadvantage of being performance costly by handling the calculations in a new way so that it is able to not only be an effective SQLIA prevention method but also performance inexpensive, therefore the first source will be useful for implementation in the artefact stage while the second will help by keeping the techniques in mind so as to answer the research question.

Some authors modify tried and trusted techniques to take on new threats such as the work of Avireddy*, et al* 2012 who aims to prevent SQLIA by using encryption that is randomized. The paper goes further to show how all input from users are encrypted to ensure no malicious statements are successful, the randomization element supposedly increases performance and security from the encryption performed, however the paper is light on details and considering the performance costs imposed by encryption techniques it is likely that this solution should it be effective will come at a cost. This use of encryption but in a new way is similar to the work of Mittal, P 2013 who also uses encryption to prevent SQLIA but instead of modifying encryption with a software solution they opt for hardware with the use of GPU CUDA processing for greater performance when using encryption and goes into detail on this process unlike the first source which keeps its showcase of their solution quite general which in this case is a downside as it makes it unclear the effect their solution will have, therefore both sources solutions will need investigating in order to provide interesting results that will help answer the research question.

# Technical Chapters (change this to something appropriate) Research Methods, Findings, Critical Self-Evaluation

## Research Method

The research method this paper will employ is quantitative as the research performed will be of a secondary basis, copious amounts of journal articles, thesis's, books, conference papers etc from the current research by other researchers will be collected and analysed in order to understand the scope of the issue and be able to see where the research has yet to go or take what has been done further. Research priority will be on case studies where SQLIA occurred in industry, defences were actively used, as well as statistical data on SQLIAs in order to see how much of an issue and how widespread throughout the IT industry it was, where it is now. Statistical data on the various SQLIA methods would be beneficial so as to see which would be best to test against in the artefact as well as the solutions for dealing with them to see which are the most popular for this issue so they can also be used in the artefact and in order to help answer the research question.

The specific research methodology to be used will be the waterfall method, this will involve an analysis of exactly what is required to develop the artefact as well as answering the research question, next a design will be planned out for the proposed artefact which will take advantage of what was learned in the analysis stage, from this the artefact will be developed according to the design and then tested as shown in the deliverable, this will be analysed in order to formulate the answer to the research question.

## Artefact

The artefact produced from the literature consists of three parts, first the testing of the chosen DBMS's built in SQLI defences against the different types of SQLIAs found in the literature and the results of that testing. Second is the testing of the chosen DBMS's against SQLIAs with the use of third party tools found in the literature and the results of that testing. Thirdly is the collation of the results of all testing into a report of how much of a threat SQLIA poses with and without third party tools and how this relates to the research question.

### Artefact - DBMS SQLIA testing

In this part of the artefact the MySQL, Oracle and Microsoft SQL server databases are attacked by SQLIAs without any third party support to check their resilience to such attacks; a tautology attack is launched on the test databases, starting with MySQL, the input with a username of ‘USER’ and password of ‘PASS OR WHERE 1=1’ as shown in figure [[9.1.1](#_MySQL_tautology_attack)] this statement attempts to gain access to the database by appending a query that always returns true which causes the database to accept the input, however as shown in figure [[9.1.2](#_MySQL_tautology_attack_1)] this was not accepted and an error message is shown, meaning the database did not accept the input and the SQLIA failed. A similar input was used to attack the Oracle database as shown in figure [[9.1.3](#_Oracle_tautology_attack)] and was successful as shown in figure [[9.1.4](#_Oracle_tautology_attack_1)] as well as the attack on the Microsoft SQL server database with a similar input as shown in figure [[9.1.5](#_MSQLS_tautology_attack)] which was successful as shown in figure [[9.1.6](#_MSQLS_tautology_attack_1)].

Illegal or logically incorrect queries take advantage of error messages produced by the web application when the database detects a query that is incorrect and throws back information that attackers can use to learn about the structure of the database from those error messages, this was attempted on the MySQL database with the input of username ‘USER’ and password of ‘pass1\’ as shown in figure [[9.2.1](#_MySQL_Illegal/Logically_Incorrect)], with the backslash causing the query to become incorrect and is shown to be successful in figure [[9.2.2](#_MySQL_Illegal/Logically_Incorrect_1)] as it produces an error that shows the desired type of input needed to gain access to the database. This was attempted on the Oracle database with a similar input in figure [[9.2.3](#_Oracle_Illegal/Logically_Incorrect)] but instead of producing an error it was successful in granting access to the database as shown in figure [[9.2.4](#_Oracle_Illegal/Logically_Incorrect_1)], with similar input an attack on the Microsoft SQL server database as shown in figure [[9.2.5](#_MSQLS_Illegal/Logically_Incorrect)] and is shown to be successful in producing a revealing error message in figure [[9.2.6](#_MSQLS_Illegal/Logically_Incorrect_1)].

Union queries make use of the ‘UNION’ command in SQL statements in order to append malicious statements onto regular statements, this was attempted on the MySQL database with the input of username ‘USER’ and password ‘1 UNION ALL SELECT PASSWORD FROM SQLI’ as shown in figure [[9.3.1](#_MySQL_Union_Query)] which resulted in an unsuccessful SQLI attempt as shown in figure [[9.3.2](#_MySQL_Union_Query_1)]. A similar input was used to attack the Oracle database as shown in figure [[9.3.3](#_Oracle_Union_Query)] and is shown to be successful in figure [[9.3.4](#_Oracle_Union_Query_1)], as well as with a similar input for the attack on the Microsoft SQL server database in figure [[9.3.5](#_MSQLS_Union_Query)] and is shown to be successful in figure [[9.3.6](#_MSQLS_Union_Query_1)].

Piggy-backed queries are similar to the Union query attack by appending malicious statements onto regular queries but it does this differently by using a semicolon instead of the 'UNION' command, this is shown in figure [[9.4.1](#_MySQL_Piggy-backed_Queries)] with the MySQL database as a target and the input of username 'USER' and password '1; DROP TABLE SQLI' which was unsuccessful as shown in figure [[9.4.2](#_MySQL_Piggy-backed_Queries_1)]. A similar input was used in the attack on the Oracle database as shown in figure [[9.4.3](#_Oracle_Piggy-backed_Queries)] and was unsuccessful in causing damage to the database but did gain access to it as shown in figure [[9.4.4](#_Oracle_Piggy-backed_Queries_1)], likewise the attack on the Microsoft SQL server used similar input as shown in figure [[9.4.5](#_MSQLS_Piggy-backed_Queries)] and also resulted in gaining access to the database as shown in figure [[9.4.6](#_MSQLS_Piggy-backed_Queries_1)].

Blind injection attack is similar to the incorrect queries attack in that the attacker uses input to cause error messages in order to learn about the target database and determine the malicious statement needed to gain access to the database this is shown in figure [[9.5.1](#_MySQL_Blind_Injection)] with the MySQL database as a target and the input of username 'test and 1=1' for the first attempt and 'test and 1=0' for the second as shown in figure [[9.5.1](#_MySQL_Blind_Injection)] which was unsuccessful as shown in figure [[9.5.2](#_MySQL_Blind_Injection_1)]. A similar input was used in the attack on the Oracle database as shown in figure [[9.5.3](#_Oracle_Blind_Injection)] but it did not generate an error message but was successful in gaining access to the database as shown in figure [[9.5.4](#_Oracle_Blind_Injection_1)], likewise the attack on the Microsoft SQL server used similar input as shown in figure [[9.5.5](#_MSQLS_Blind_Injection)] and also resulted in gaining access to the database as shown in figure [[9.5.6](#_MSQLS_Blind_Injection_1)].

Timing attacks take advantage of the 'WAITFOR' command in order to force the database to delay its response and in so doing gain access to it as shown in figure [[9.6.1](#_MySQL_Timing_Attack)] with the MySQL database as the target and input of username 'test = user() if 1=1 waitfor delay '0:1:5'' which was unsuccessful and generated an error message as shown in figure [[9.6.2](#_MySQL_Timing_Attack_1)]. A similar input was used in the attack on the Oracle database as shown in figure [[9.6.3](#_Oracle_Timing_Attack)] which did not cause a delay but was successful in gaining access to the database as shown in figure [[9.6.4](#_Oracle_Timing_Attack_1)], likewise the attack on the Microsoft SQL server used similar input as shown in figure [[9.6.5](#_MSQLS_Timing_Attack)] and also resulted in gaining access to the database as shown in figure albeit with an error message as shown in figure [[9.6.6](#_MSQLS_Timing_Attack_1)].

Alternate encodings append a particular bad character to a regular query, in this case 'char(44)' , in order to gain access to the database as shown in figure [[9.7.1](#_MySQL_Alternate_Encodings)] with an input of username 'test' and password '0; exec (0x73587574 64 5f77 6e)' and was unsuccessful as shown in figure [[9.7.2](#_MySQL_Alternate_Encodings_1)]. A similar input was used in the attack on the Oracle database as shown in figure [[9.7.3](#_Oracle_Alternate_Encodings)] that was successful in gaining access to the database as shown in figure [[9.7.4](#_Oracle_Alternate_Encodings_1)], likewise the attack on the Microsoft SQL server used similar input as shown in figure [[9.7.5](#_MSQLS_Alternate_Encodings)] and also resulted in gaining access to the database as shown in figure [[9.7.6](#_/_MSQLS_Alternate)].

In total as shown by figure [[9.8](#_SQLIA_results)] the MySQL database was the most resilient to these attacks while Oracle and Microsoft SQL server were much more so in most cases although there were examples where SQLIA attempts resulted in access to the database where it was not expected however none of the test databases detected SQLIAs were taking place.

### Artefact - Burp Suite testing

This tool allows for the testing of web applications by intercepting requests made to an application such as a login request by stopping the request from being sent to the database and provides the user input from the login request for analysis so as to see if it is harmful, based on this the request can be forwarded to the database or be blocked. First a request to the application is sent as shown in figure [[10.1](#_Login_request_sent)] with the input of username 'user1' and password 'pass1' which is then intercepted and its details outlined as shown in figure [[10.2](#_Request_detection)] and the exact input in figure [[10.3](#_Input_shown)].

Next SQLI attempts are made on the test databases starting with MySQL with a tautology SQLIA as shown in figure [[10.4](#_SQLIA_intercepted)] with the input of username 'test' and password 'PASS OR WHERE 1=1' which is intercepted by the program and the input involved correctly identified, this is then successfully blocked as shown by figure [[10.5](#_MySQL_SQLIA_blocked)]. A Union query attack is made on the Oracle database with the input of username 'USER' and password '1 UNION ALL SELECT PASSWORD FROM SQLI' which is intercepted and input correctly identified as shown by figure [[10.6](#_Oracle_SQLIA_blocked)]. A alternate encodings attack is launched on the Microsoft SQL server database with the input of username 'user1'' and password '0; exec (0x73587574 64 5f77 6e)' which is intercepted and correctly outlined in figure [[10.7](#_MSQLS_SQLIA_blocked)].

This is useful as it has shown itself to be an effective defence against SQLIAs however due to its manual nature it would be impractical for this to be implemented in a large companies web service unless it could be programmed with white and block lists that allow automatic forwarding or blocking for requests at runtime.

### Artefact - Metasploit testing

This tool tests web applications by attacking them with various methods in order to determine any vulnerabilities they may have, some of the tools included are penetration testing, the sending of dubious emails or spam for phishing purposes, a web application scan as well as vulnerability scanning and more as shown in figure [[11.1](#_Metasploit_testing_options)]. The web application scan is selected for its broad vulnerability detection and exploitation functionality as shown in figure [[11.2](#_Web_app_test)] where the test is configured with the links to the web applications, the maximum pages to request per URL, the time limit for each of the links and the number of simultaneous requests to be made for each of the links. The program then finds and exploits any vulnerabilities present in the web applications and generates a report on its findings where as shown in figure [[10.3](#_Metasploit_testing_results)] two vulnerabilities were found and in figure [[10.4](#_Metasploit_SQLI_vulnerabilities)] they are shown to be SQLI vulnerabilities with the exact details of the vulnerabilities in question.

The program has successfully scanned and found SQLI vulnerabilities, therefore the program is useful in defending against SQLIAs by strengthening applications against them with the finding of vulnerable parts that attackers take advantage of in order to perform malicious attacks as well as giving advice on how to deal with this so that these vulnerabilities are no longer an issue with attackers unable to take advantage of them which effectively improves security.

### Artefact - Report

The DBMS's tested mostly showed some resilience to attack but not in all cases and in all cases the attacks were not detected as SQLIA as shown in figure [9.8], the MySQL testing in particular showed that it was well secured against the forms of SQLIAs that were pitted against it with all attempts resulting in either a failure to gain access to the database or errors that were not explicitly revealing of the databases structure for a an attacker to take advantage of which is telling of the effort put into the research and development of SQLI defences for this DBMS, however it is important to note that although several types of SQLIA were used to test this DBMSs defensive capability there are still more that it could be tested against that may take advantage of security loopholes that have as yet been undetected or dealt with.

The Oracle database was also tested for its SQLI defensive capability against several types of SQLIAs and as shown in figure [[9.8](#_SQLIA_results)] it is clear that although not every SQLI attempt was successful in its objective such as gaining access to the database, learning the databases structure through error messages or causing a delay in its response as the database did not always react in a way that was expected such as granting access at times where it was not the objective or producing error messages etc there were enough successful SQLIAs as well as being undetected to show how much more vulnerable the DBMS is in this particular test than its competitors, this is amplified by the fact that there are many more types of SQLIA that could be used to gain access to the database and outlines the need for third party SQLI defensive tools.

The Microsoft SQL server database underwent SQLI testing in order to determine its base defensive ability against the chosen types of SQLIAs as shown in figure [[9.8](#_SQLIA_results)] where it illustrates how even though this DBMS in particular was more resilient to SQLIA than the Oracle database it is clear that it is still vulnerable to these types of attacks and did not always react as expected with error messages and access granted where it shouldn’t have been, when compared to the MySQL database it reflects poorly as it performed much better with no successful attacks, this combined with the amount of SQLIA types that could have been used to test its defences show how useful including SQLI detection and prevention methods through third party support would be.

With third party support however the security of the test databases was vastly increased with attacks intercepted and blocked using Burp Suite as shown in figure [[10](#_Appendix_5_–)] as every request can be monitored to determine if it is a threat although this can be improved to be automated in nature so that permanent staff placements aren't as needed in order to implement such a method in a large system architecture. As well as correctly diagnosing vulnerabilities as SQLI in nature through the use of Metasploit in figure [[11](#_Appendix_6_–)] although this does take time which detracts from system performance where it may be needed to be taken down for maintenance should a scan be performed. Nevertheless this could be improved as each circumstance where a defence for SQLIA in a web service is needed is different in nature and so some methods will be more effective than others at stopping malicious attacks on databases.

The testing performed on the test databases, MySQL, Oracle and Microsoft SQL server with and without third party tools it is clear that SQLI is still popular as an attacking method and based on testing is shown to be still successful as such but in a reduced state to what it once was before the vast amount of research was performed by academics seeking to find a solution to this issue.

# Conclusion

## Summary

In conclusion this paper has looked at the issue of SQLIA and researched the popular attacking methods in use by malicious individuals aiming to gain prohibited access to databases, the solutions developed to detect and prevent these attacks. This research was used to test both attacking methods and solutions on popular DBMSs to see their effectiveness and from this produce a report that aims to answer the question of whether SQLIA is still an issue.

## Evaluation

This paper has looked at various SQLIA methods in use by attackers as well as the solutions for dealing with them and the advantages and disadvantages that come with them, however compared to the amount of research that has been performed in this area it is apparent that this is not a comprehensive introduction to the topic by any means, therefore if there was time or in the future this could be improved to find and document even more SQLIA methods and solutions as well as providing examples of statistical data on their use for and against the issue at hand, also including examples of these in use with specific case studies and from this be able to make a more well informed analysis of both sides of SQLIA.

In the artefact stage several types of SQLIAs were tested against popular DBMSs which included a significant amount of test data, with and without third party support in the form of SQLIA solutions found in the research, although this amply shows their effectiveness for the attacking methods, solutions and the DBMSs this could be improved by if there was more time or in the future, including different types of attacking methods as this will further test the defences of the DBMSs with and without third party support. This could also be improved by further testing the functionality of the chosen SQLIA solutions as well as introducing more solutions in order to test their effectiveness in detecting and preventing SQLIA to better answer the research question.

The report produced as a result of the research and testing that was performed although goes into detail on the work undertaken during this project in order to provide an acceptable answer to the research question, this could be improved by if there was more time or in the future by being able to use the results of more research and testing, comparing this to other works in academia with their results based on their research and testing so as to better see if the results of these project are similar or not in order to tell their legitimacy and in so doing be able to produce a more well rounded and thorough answer the research question of whether SQLIA is still an issue.

# References

Shar, L, & Tan, H 2013, 'Defeating SQL Injection', Computer, 46, 3, pp. 69-77, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=d15044ea-dcee-4f89-92c5-2b8ba737f5cb%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=86642466>

Dorai, R, & Kannan, V 2011, 'SQL Injection-Database Attack Revolution and Prevention', Journal Of International Commercial Law & Technology, 6, 4, pp. 224-231, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=efa309f6-4288-4ded-a88b-e67f95acfb88%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=67217369>

Kim, M, & Lee, D 2014, 'Data-mining based SQL injection attack detection using internal query trees', Expert Systems With Applications, 41, 11, pp. 5416-5430, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=00dc6188-3a98-4251-bcf5-3ef2674d7f68%40sessionmgr198&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=95713988>

Pinzón, C, De Paz, J, Herrero, Á, Corchado, E, Bajo, J, & Corchado, J 2013, 'idMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL injection through data mining', Information Sciences, 231, pp. 15-31, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=e2420720-2d0e-4d8f-a4ca-87267c368cc0%40sessionmgr110&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=85615586>

Lee, I, Jeong, S, Yeo, S, & Moon, J 2012, 'A novel method for SQL injection attack detection based on removing SQL query attribute values', Mathematical & Computer Modelling, 55, 1/2, pp. 58-68, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=ee7215cf-7be2-4a8d-aae0-90ff03d332c0%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=67248440>

Cho, Y, & Pan, J 2015, 'Design and Implementation of Website Information Disclosure Assessment System', Plos ONE, 10, 3, pp. 1-29, Academic Search Complete, EBSCOhost, viewed 13 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=43b56978-87a6-46e8-a556-e9892551ccbd%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=101838118>

Yu, H, Lai, H, Chen, K, Chou, H, Wu, J, Dorjgochoo, S, Mendjargal, A, Altangerel, E, Tien, Y, Hsueh, C, & Lai, F 2013, 'A sharable cloud-based pancreaticoduodenectomy collaborative database for physicians: Emphasis on security and clinical rule supporting', Computer Methods & Programs In Biomedicine, 111, 2, pp. 488-497, Academic Search Complete, EBSCOhost, viewed 16 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=d12e5a0a-0328-4fb4-87a5-6fb4c96f6448%40sessionmgr112&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=89136411>

HWEEHWA, P, & XUHUA, D 2014, 'Privacy-Preserving Ad-Hoc Equi-Join on Outsourced Data', ACM Transactions On Database Systems, 39, 3, pp. 23-23:40, Academic Search Complete, EBSCOhost, viewed 17 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=c6fe2ef4-fd11-4840-ab08-a83ea08e79a6%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=98910097>

Wei, D, Murugesan, S, Kuo, S, Naik, K, & Krizanc, D 2013, 'Enhancing Data Integrity and Privacy in the Cloud: An Agenda', Computer, 46, 11, pp. 87-90, Academic Search Complete, EBSCOhost, viewed 18 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=45b2a410-a7d1-4465-affb-0899b04e9f84%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=92663519>

Boicea, A, Ghita, V, Radulescu, F, & Sarbu, A 2010, 'ENCRYPTION STRATEGIES IN DATABASES', Annals Of DAAAM & Proceedings, pp. 165-166, Academic Search Complete, EBSCOhost, viewed 19 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=1c50ad14-dfa2-4a35-ac90-2165e1abfc4c%40sessionmgr110&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=55674541>

Suciu, D 2012, 'SQL on an Encrypted Database', Communications Of The ACM, 55, 9, p. 102, Academic Search Complete, EBSCOhost, viewed 21 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=2c0daa8e-1793-4ff9-88f0-5856905aa3fc%40sessionmgr111&vid=2&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=79648438>

Roy, S, Singh, A, Sairam, A 2011, ‘Detecting and Defeating SQL Injection Attacks’, International Journal of Information and Electronics Engineering, 1, 1, pp.38-46, Viewed 15 June 2015. Available at:

<http://www.ijiee.org/papers/6-E080.pdf>

Kerner, SM 2013, 'DB Networks Looks to Detect SQL Injection With New Appliance', *Eweek*, p. 1, Academic Search Complete, EBSCO*host*, viewed 24 June 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?vid=3&sid=9916cce8-fb31-4ff2-94df-2d60ca2f897d%40sessionmgr114&hid=123&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=91701564>

Kindy, D, Pathan, A 2013, ‘A Detailed Survey on various aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks and Remedies’, International Journal of Communication Networks and Information Security (IJCNIS), 5, 2, pp.80-92, Viewed 24 June 2015. Available at:

<http://www.ijcnis.org/index.php/ijcnis/article/view/364/115>

Salama, S, Marie, M, El-Fangary, L, Helmy, Y 2012, ‘Web Anomaly Misuse Intrusion Detection Framework for SQL Injection Detection’, International Journal of Advanced Computer Science and Applications(IJACSA), 3, 3, pp.133-157, Viewed 24 June 2015. Available at:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.4544&rep=rep1&type=pdf#page=133>

Štampar, M 2013, ‘Data Retrieval over DNS in SQL Injection Attacks’, arXiv13033047, pp.1-7, Viewed 24 June 2015. Available at:

<http://arxiv.org/ftp/arxiv/papers/1303/1303.3047.pdf>

Asha, N, Kumar, M, Vaidhyanathan, G 2012, ‘Preventing SQL Injection Attacks’, International Journal of Computer Applications, 52,13, pp.28-32, Viewed 24 June 2015. Available at:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8182&rep=rep1&type=pdf>

Dogbe, E, Millham, R, Singh, P 2013, ‘A combined approach to prevent SQL Injection Attacks’, Science and Information Conference 2013 October 7-9 London UK, pp.406-410, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6661770>

Patel, N, Mohammed, F, Soni, S 2011, ‘SQL Injection Attacks: Techniques and Protection Mechanisms.’, International journal on computer science and engineering, 3, 27, pp.199-203, Viewed 24 June 2015. Available at:

<http://www.enggjournals.com/ijcse/doc/IJCSE11-03-01-139.pdf>

Tajpour, A, Ibrahim, S, Sharifi, M 2012, ‘Web Application Security by SQL Injection Detection Tools’, International journal of computer science issues, 9, 2, pp.332-339, Viewed 24 June 2015. Available at:

< http://www.ijcsi.org/papers/IJCSI-9-2-3-332-339.pdf>

Ahmad, K, Shekhar, J, Yadav, K 2011, 'Coalesce Techniques to Secure Web Applications and Databases against SQL Injection Attacks', Electronic Journal of Computer Science & Information Technology, 3, 1, pp.26-30, Viewed 24 June 2015. Available at:

<http://ejcsit.uniten.edu.my/index.php/ejcsit/article/view/14/23>

Jang, Y, Choi J 2014, 'Detecting SQL injection attacks using query result size', Computers & Security (2014), 44, pp.104-118, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com/S0167404814000595/1-s2.0-S0167404814000595-main.pdf?\_tid=a63e4910-1b5b-11e5-9d41-00000aacb361&acdnat=1435251795\_b38933d613ea110bcdc5f7c3070129ee>

Shar, L, Tan, H 2013, 'Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns', Information and Software Technology (2013), 55, 10, pp.1767-1780, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com.ezproxy.wlv.ac.uk/S0950584913000852/1-s2.0-S0950584913000852-main.pdf?\_tid=7fd998a0-1b5c-11e5-a50c-00000aab0f02&acdnat=1435252160\_88dad02114910435c3a5b8d682d01316>

Lei, L, Jing, X, Minglei, L, Jufeng, Y 2013, 'A Dynamic SQL Injection Vulnerability Test Case Generation Model Based on the Multiple Phases Detection Approach', 2013 IEEE 37th Annual Computer Software and Applications Conference (2013), pp.256-261, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6649829>

Noiumkar, P 2012, ' The SQL Injection Attacking Prevention Applying the Design Techniques of IPTABLES', Journal of Convergence Information Technology (2012), 7, 22, pp.66-75, Viewed 24 June 2015. Available at:

<http://www.globalcis.org/jcit/ppl/JCIT2130PPL.pdf>

Abawajy, J 2013, ' SQLIA detection and prevention approach for RFID systems', Journal of Systems and Software (2013), 86, 3, pp.751-758, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com/S0164121212003159/1-s2.0-S0164121212003159-main.pdf?\_tid=71891994-1b5f-11e5-a1e8-00000aacb362&acdnat=1435253424\_e943fd8ce08f8aa32e017139e486b240>

Johari, R, Sharma, P 2012, 'A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection', Proceedings - International Conference on Communication Systems and Network Technologies, CSNT 2012 (2012), pp.452-458, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6200667>

Alserhani, F, Akhlaq, M, Awan, I, Cullen, A 2011, 'Event-based alert correlation system to detect SQLI activities', Proceedings - International Conference on Advanced Information Networking and Applications, AINA (2011), pp.175-182, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5763386>

Mittal, P, Jena, S 2013, 'A fast and secure way to prevent SQL injection attacks', 2013 IEEE Conference on Information and Communication Technologies, ICT 2013 (2013), pp.730-734, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6558190>

Avireddy, S, Perumal, V, Zowraj, N, Kannan, R, Thinakaran, P, Ganapthi, S, Gunasekaran, J, Prabhu, S 2012, 'Random4: An application specific randomized encryption algorithm to prevent SQL injection', Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012 (2012), pp.1327-1333, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6296134>

# Appendix 1 - Proposal

**Name -** James Braznell

**Student Number -** 1007022

**Award -** MSC Computer Science

**Project Title -**

SQL Injection: - The popular methods, solutions and is it still an issue

**Aim -**

To create a report on SQL Injection database intrusion techniques and its current state as a security threat to be considered

**Objectives -**

* Perform extensive re research into SQL Injection
* Build databases, develop related applications and test numerous database management systems against SQL Injection attack
* Create a report on findings based on research and practical tests

**Deliverable -**

After performing extensive research into SQLIAs and the solutions to that threat there will be several tests on whether SQLIA is still an issue. This will involve the development of test applications that perform specific jobs possibly using several different programming languages such as PHP, Java, HTML, etc in order to test their resilience against this threat and they will be connected to several databases using different Database Management systems(DBMS) such as Oracle, MySQL, Microsoft SQL Server etc with a substantial amount of test data to test their built in methods for dealing with various SQLIA methods and then move on to testing with the addition of third party tools for SQLIA defence that were found based on the research performed beforehand. With the information gained from this artefact there will be a report produced on the research and the findings combined in order to provide an answer on whether SQL Injection is still an issue.

**Research question(s) -**

What are the popular SQL Injection database intrusion methods and has the computing industry made the practice ineffectual?

**Background –**In recent years information security has become more and more of a popular topic of research due to the growing nature of the internet as more and more individuals gain access to it and as such the number of cyber-attacks on businesses and the confidential customer data that they hold has increased as a result. Therefore there has been a fair amount of research into stopping these attacks specifically at the front end of applications that businesses are using to interact with customers however this does leave the back end of the application vulnerable if attackers are able to reach it which is a part of much of the current research in information security, how to deal with the application vulnerabilities that attackers are taking advantage of. One of the bigger vulnerabilities is SQL injection attack (SQLIA) which is a technique where the attacker attempts to alter a SQL query by injecting SQL code into front end applications for example with an empty text box that allows the malicious individual access to the database in order to retrieve sensitive information, alter the database or even wipe the database entirely.

SQLIAs come in several different forms, one of which centres around the SQL command UNION that combines queries, if used for malicious purposes this command will alter the original intended query by merging it with a query devised by the hacker to gain access to more of the database, steal data or damage the database if the correct data types for the columns involved are selected. Another type of SQLIA attempts to bypass any efforts by developers to stop SQLIAs within their code or with third party tools, this is done by altering the malicious query to instead use alternate encoding such as hexadecimal or Unicode which is not a part of developers considerations when it comes to securing their application and database allowing the attacker to perform malicious acts. Another type of SQLIA takes advantage of any leftover sets of SQL commands packaged together as a procedure that came with the particular database management system (DBMS) that a developer may have chosen to support their application, the attacker will attach their malicious SQL statement to a procedure so that when run their statement will allow them to perform malicious acts, a SQLIA similar to this is where the attacker attaches their malicious SQL statement to another non malicious SQL statement by adding a semicolon at the end so that it ends it, this will cause the first statement to be run but also allow the attackers malicious statement to be run. Another SQLIA is where the attacker does not get any prompt from the application that their efforts are being successful in the form of error messages, the attacker instead injects SQL statements that are correct or incorrect and the application will determine this and based on the answer the attacker will learn more about the databases stored data than was intended. Another SQLIA makes use of time centric operators such as BENCHMARK() in the malicious SQL statement as if this is set to repeat a high enough number of times and returns correctly the statement will run and hamper server performance severely as the malicious statement is being run. Another SQLIA makes use of malicious SQL statements that are made purposefully incorrect by the attacker which when injected and run will cause error messages to appear allowing the attacker to learn the structure of the applications database as well as individual records. Another SQLIA makes use of SQL statements that are purposefully set up to be considered correct by the application they are injected into allowing the attacker access to more of the database than they should leading to stolen data and damages.

Based on the research performed on this topic several solutions have been devised to help secure applications against this kind of attack, there are tools that will scan an application in its entirety to look for known vulnerabilities in the code and will depending on the tool, make the user aware of the issue, provide assistance in rectifying the problem code or change the code in question to no longer be vulnerable to SQL injection attack on an automatic basis, while this is very useful for securing applications against known threats it is however ineffective against new SQLIAs that it does not have a solution for making it not very adaptable in the long term.

Another solution for dealing with the threat of SQLIAs is SQLIA detection tools; this is where the tool will run in the background of the normal application runtime and based on the tools method will scrutinize the SQL code that is passed from the front end application to the back end database against known SQLIAs and will block any deemed to be a threat automatically, the tools available do this in various ways and some are even quite adaptable in that they will attempt learn the traits of SQLIAs and be able to block new SQLIAs based on this, however due to the fact that the tools will run in the background of normal operations depending on the tool it may cause a noticeable slowdown in performance.

Another solution for securing against SQLIAs would be to introduce a third party encryption tool that will take the records in the database and encrypt them in such a way so that even if a SQLIA is successful the attacker will only receive data that is essentially useless to them without the password to decrypt them an effective method that has seen much use as a part of standard DBMSs to deal with other types of database threats such as malicious employees for example. Depending on the tool they can offer added benefits such as being easier to implement as well as fine tuning the encryption to what is required whether it be a high level of encryption for added security or light encryption for security with less of a performance cost added to it, as well as other added benefits that such tools bring with them, however depending on the amount of encryption and how widespread its use on the database there will be a performance cost as well as the time needed to decrypt the data whenever it is needed.

Based on this it is clear to see that much has been said and done about the issue of SQLIA, including research on the various methods attackers use to gain access to what should be secure data as well as the solutions that have been devised in an attempt to secure against this and ultimately end the threat that it poses, however despite this there is no all inclusive method for dealing with SQLIA as well as the disadvantages some of the current methods have when implemented. Therefore this paper will look further at the current SQLIA methods in use by attackers, the solutions for dealing with them and from this test against several DBMSs whether SQLIA is still an issue.

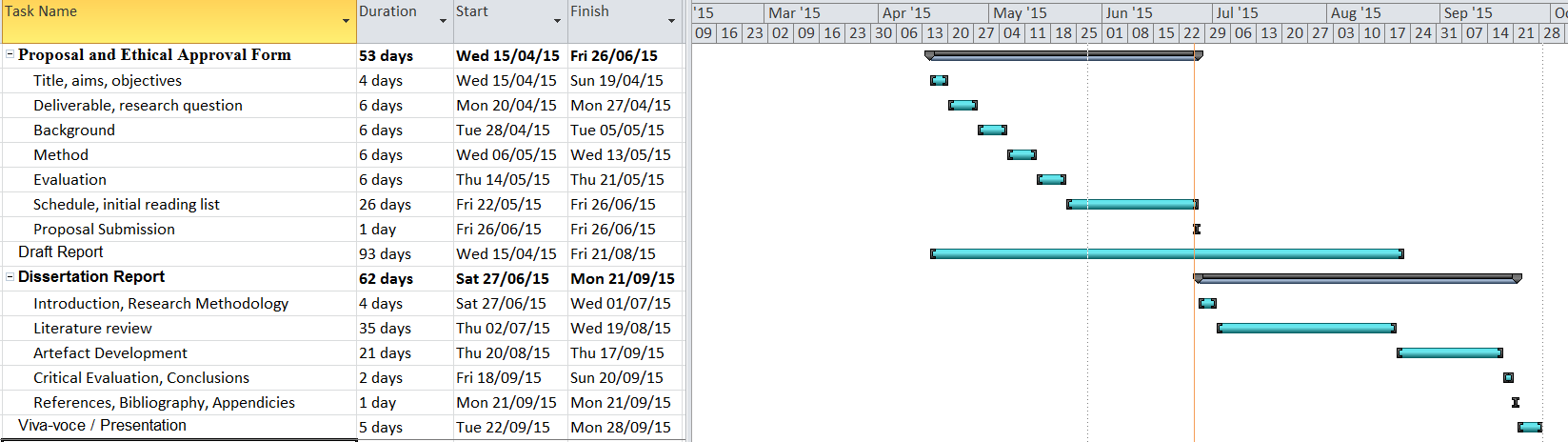
**Method -**

The research method this paper will employ is quantitative as the research performed will be of a secondary basis, copious amounts of journal articles, thesis's, books, conference papers etc from the current research by other researchers will be collected and analysed in order to understand the scope of the issue and be able to see where the research has yet to go or take what has been done further. Research priority will be on case studies where SQLIA occurred in industry, defences were actively used, as well as statistical data on SQLIAs in order to see how much of an issue and how widespread throughout the IT industry it was, where it is now. Statistical data on the various SQLIA methods would be beneficial so as to see which would be best to test against in the artefact as well as the solutions for dealing with them to see which are the most popular for this issue so they can also be used in the artefact and in order to help answer the research question.

The specific research methodology to be used will be the waterfall method, this will involve an analysis of exactly what is required to develop the artefact as well as answering the research question, next a design will be planned out for the proposed artefact which will take advantage of what was learned in the analysis stage, from this the artefact will be developed according to the design and then tested as shown in the deliverable, this will be analysed in order to formulate the answer to the research question.

**Evaluation -**

This paper has looked at various SQLIA methods in use by attackers as well as the solutions for dealing with them and the advantages and disadvantages that come with them, however compared to the amount of research that has been performed in this area it is apparent that this is not a comprehensive introduction to the topic by any means, therefore if there was time this could be improved to find and document even more SQLIA methods and solutions as well as providing examples of statistical data on their use for and against the issue at hand, also including examples of these in use with specific case studies and from this be able to make a more well informed analysis of both sides of SQLIA.

**Provisional schedule -**

**Initial reading list -**

Shar, L, & Tan, H 2013, 'Defeating SQL Injection', Computer, 46, 3, pp. 69-77, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=d15044ea-dcee-4f89-92c5-2b8ba737f5cb%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=86642466>

Dorai, R, & Kannan, V 2011, 'SQL Injection-Database Attack Revolution and Prevention', Journal Of International Commercial Law & Technology, 6, 4, pp. 224-231, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=efa309f6-4288-4ded-a88b-e67f95acfb88%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=67217369>

Kim, M, & Lee, D 2014, 'Data-mining based SQL injection attack detection using internal query trees', Expert Systems With Applications, 41, 11, pp. 5416-5430, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=00dc6188-3a98-4251-bcf5-3ef2674d7f68%40sessionmgr198&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=95713988>

Pinzón, C, De Paz, J, Herrero, Á, Corchado, E, Bajo, J, & Corchado, J 2013, 'idMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL injection through data mining', Information Sciences, 231, pp. 15-31, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=e2420720-2d0e-4d8f-a4ca-87267c368cc0%40sessionmgr110&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=85615586>

Lee, I, Jeong, S, Yeo, S, & Moon, J 2012, 'A novel method for SQL injection attack detection based on removing SQL query attribute values', Mathematical & Computer Modelling, 55, 1/2, pp. 58-68, Academic Search Complete, EBSCOhost, viewed 9 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=ee7215cf-7be2-4a8d-aae0-90ff03d332c0%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=67248440>

ALLEN, J 2014, 'It's Two A.M.: Do You Know Where Your Data Is and Who Can Access It?', American Journal Of Family Law, 28, 1, pp. 20-24, Academic Search Complete, EBSCOhost, viewed 10 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=352ca9a7-c9ce-4492-b269-d3c382864f4f%40sessionmgr113&vid=2&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=94589596>

Popa, R, Redfield, C, Zeldovich, N, & Balakrishnan, H 2012, 'CryptDB: Processing Queries on an Encrypted Database', Communications Of The ACM, 55, 9, pp. 103-111, Academic Search Complete, EBSCOhost, viewed 11 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=1d1000ae-c4a7-4284-9cfd-34e41ec6b65c%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=79648445>

de Morais, W, Lundström, J, & Wickström, N 2014, 'Active In-Database Processing to Support Ambient Assisted Living Systems', Sensors (14248220), 14, 8, pp. 14765-14785, Academic Search Complete, EBSCOhost, viewed 12 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=ecab6623-2d08-49b9-9155-ce5c4126a523%40sessionmgr198&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=97665156>

Cho, Y, & Pan, J 2015, 'Design and Implementation of Website Information Disclosure Assessment System', Plos ONE, 10, 3, pp. 1-29, Academic Search Complete, EBSCOhost, viewed 13 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=43b56978-87a6-46e8-a556-e9892551ccbd%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=101838118>

Yaseen, Q, & Panda, B 2012, 'Insider threat mitigation: preventing unauthorized knowledge acquisition', International Journal Of Information Security, 11, 4, pp. 269-280, Academic Search Complete, EBSCOhost, viewed 14 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=f9b1c3b2-1f6a-42be-91f4-fb4740d469c4%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=77959064>

Chakraborty, A, Majumdar, A, & Sural, S 2010, 'A column dependency-based approach for static and dynamic recovery of databases from malicious transactions', International Journal Of Information Security, 9, 1, pp. 51-67, Academic Search Complete, EBSCOhost, viewed 15 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=fee2b5f0-3760-4f84-bdce-a844c2c97265%40sessionmgr114&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=47728631>

Yu, H, Lai, H, Chen, K, Chou, H, Wu, J, Dorjgochoo, S, Mendjargal, A, Altangerel, E, Tien, Y, Hsueh, C, & Lai, F 2013, 'A sharable cloud-based pancreaticoduodenectomy collaborative database for physicians: Emphasis on security and clinical rule supporting', Computer Methods & Programs In Biomedicine, 111, 2, pp. 488-497, Academic Search Complete, EBSCOhost, viewed 16 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=d12e5a0a-0328-4fb4-87a5-6fb4c96f6448%40sessionmgr112&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=89136411>

HWEEHWA, P, & XUHUA, D 2014, 'Privacy-Preserving Ad-Hoc Equi-Join on Outsourced Data', ACM Transactions On Database Systems, 39, 3, pp. 23-23:40, Academic Search Complete, EBSCOhost, viewed 17 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=c6fe2ef4-fd11-4840-ab08-a83ea08e79a6%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=98910097>

Wei, D, Murugesan, S, Kuo, S, Naik, K, & Krizanc, D 2013, 'Enhancing Data Integrity and Privacy in the Cloud: An Agenda', Computer, 46, 11, pp. 87-90, Academic Search Complete, EBSCOhost, viewed 18 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=45b2a410-a7d1-4465-affb-0899b04e9f84%40sessionmgr111&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=92663519>

Boicea, A, Ghita, V, Radulescu, F, & Sarbu, A 2010, 'ENCRYPTION STRATEGIES IN DATABASES', Annals Of DAAAM & Proceedings, pp. 165-166, Academic Search Complete, EBSCOhost, viewed 19 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=1c50ad14-dfa2-4a35-ac90-2165e1abfc4c%40sessionmgr110&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=55674541>

NAGY, N, & AKL, S 2010, 'A QUANTUM CRYPTOGRAPHIC SOLUTION TO THE PROBLEM OF ACCESS CONTROL IN A HIERARCHY', Parallel Processing Letters, 20, 3, pp. 251-261, Academic Search Complete, EBSCOhost, viewed 20 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=1f5543fe-6571-4d34-9e08-208750e7b82b%40sessionmgr115&vid=0&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=53843832>

Suciu, D 2012, 'SQL on an Encrypted Database', Communications Of The ACM, 55, 9, p. 102, Academic Search Complete, EBSCOhost, viewed 21 May 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?sid=2c0daa8e-1793-4ff9-88f0-5856905aa3fc%40sessionmgr111&vid=2&hid=124&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=79648438>

Roy, S, Singh, A, Sairam, A 2011, ‘Detecting and Defeating SQL Injection Attacks’, International Journal of Information and Electronics Engineering, 1, 1, pp.38-46, Viewed 15 June 2015. Available at:

<http://www.ijiee.org/papers/6-E080.pdf>

Kerner, SM 2013, 'DB Networks Looks to Detect SQL Injection With New Appliance', *Eweek*, p. 1, Academic Search Complete, EBSCO*host*, viewed 24 June 2015. Available at:

<http://web.b.ebscohost.com/ehost/detail/detail?vid=3&sid=9916cce8-fb31-4ff2-94df-2d60ca2f897d%40sessionmgr114&hid=123&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=a9h&AN=91701564>

Kindy, D, Pathan, A 2013, ‘A Detailed Survey on various aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks and Remedies’, International Journal of Communication Networks and Information Security (IJCNIS), 5, 2, pp.80-92, Viewed 24 June 2015. Available at:

<http://www.ijcnis.org/index.php/ijcnis/article/view/364/115>

Salama, S, Marie, M, El-Fangary, L, Helmy, Y 2012, ‘Web Anomaly Misuse Intrusion Detection Framework for SQL Injection Detection’, International Journal of Advanced Computer Science and Applications(IJACSA), 3, 3, pp.133-157, Viewed 24 June 2015. Available at:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.4544&rep=rep1&type=pdf#page=133>

Štampar, M 2013, ‘Data Retrieval over DNS in SQL Injection Attacks’, arXiv13033047, pp.1-7, Viewed 24 June 2015. Available at:

<http://arxiv.org/ftp/arxiv/papers/1303/1303.3047.pdf>

Asha, N, Kumar, M, Vaidhyanathan, G 2012, ‘Preventing SQL Injection Attacks’, International Journal of Computer Applications, 52,13, pp.28-32, Viewed 24 June 2015. Available at:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.258.8182&rep=rep1&type=pdf>

Dogbe, E, Millham, R, Singh, P 2013, ‘A combined approach to prevent SQL Injection Attacks’, Science and Information Conference 2013 October 7-9 London UK, pp.406-410, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6661770>

Patel, N, Mohammed, F, Soni, S 2011, ‘SQL Injection Attacks: Techniques and Protection Mechanisms.’, International journal on computer science and engineering, 3, 27, pp.199-203, Viewed 24 June 2015. Available at:

<http://www.enggjournals.com/ijcse/doc/IJCSE11-03-01-139.pdf>

Tajpour, A, Ibrahim, S, Sharifi, M 2012, ‘Web Application Security by SQL Injection Detection Tools’, International journal of computer science issues, 9, 2, pp.332-339, Viewed 24 June 2015. Available at:

< http://www.ijcsi.org/papers/IJCSI-9-2-3-332-339.pdf>

Ahmad, K, Shekhar, J, Yadav, K 2011, 'Coalesce Techniques to Secure Web Applications and Databases against SQL Injection Attacks', Electronic Journal of Computer Science & Information Technology, 3, 1, pp.26-30, Viewed 24 June 2015. Available at:

<http://ejcsit.uniten.edu.my/index.php/ejcsit/article/view/14/23>

Jang, Y, Choi J 2014, 'Detecting SQL injection attacks using query result size', Computers & Security (2014), 44, pp.104-118, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com/S0167404814000595/1-s2.0-S0167404814000595-main.pdf?\_tid=a63e4910-1b5b-11e5-9d41-00000aacb361&acdnat=1435251795\_b38933d613ea110bcdc5f7c3070129ee>

Shar, L, Tan, H 2013, 'Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns', Information and Software Technology (2013), 55, 10, pp.1767-1780, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com.ezproxy.wlv.ac.uk/S0950584913000852/1-s2.0-S0950584913000852-main.pdf?\_tid=7fd998a0-1b5c-11e5-a50c-00000aab0f02&acdnat=1435252160\_88dad02114910435c3a5b8d682d01316>

Lei, L, Jing, X, Minglei, L, Jufeng, Y 2013, 'A Dynamic SQL Injection Vulnerability Test Case Generation Model Based on the Multiple Phases Detection Approach', 2013 IEEE 37th Annual Computer Software and Applications Conference (2013), pp.256-261, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6649829>

Noiumkar, P 2012, ' The SQL Injection Attacking Prevention Applying the Design Techniques of IPTABLES', Journal of Convergence Information Technology (2012), 7, 22, pp.66-75, Viewed 24 June 2015. Available at:

<http://www.globalcis.org/jcit/ppl/JCIT2130PPL.pdf>

Abawajy, J 2013, ' SQLIA detection and prevention approach for RFID systems', Journal of Systems and Software (2013), 86, 3, pp.751-758, Viewed 24 June 2015. Available at:

<http://ac.els-cdn.com/S0164121212003159/1-s2.0-S0164121212003159-main.pdf?\_tid=71891994-1b5f-11e5-a1e8-00000aacb362&acdnat=1435253424\_e943fd8ce08f8aa32e017139e486b240>

Johari, R, Sharma, P 2012, 'A survey on web application vulnerabilities (SQLIA, XSS) exploitation and security engine for SQL injection', Proceedings - International Conference on Communication Systems and Network Technologies, CSNT 2012 (2012), pp.452-458, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6200667>

Alserhani, F, Akhlaq, M, Awan, I, Cullen, A 2011, 'Event-based alert correlation system to detect SQLI activities', Proceedings - International Conference on Advanced Information Networking and Applications, AINA (2011), pp.175-182, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5763386>

Mittal, P, Jena, S 2013, 'A fast and secure way to prevent SQL injection attacks', 2013 IEEE Conference on Information and Communication Technologies, ICT 2013 (2013), pp.730-734, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6558190>

Prabakar M, KarthiKeyan, M, Marimuthu, K 2013, 'An efficient technique for preventing SQL injection attack using pattern matching algorithm', 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN) (2013), pp.503-506, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6528551>

Avireddy, S, Perumal, V, Zowraj, N, Kannan, R, Thinakaran, P, Ganapthi, S, Gunasekaran, J, Prabhu, S 2012, 'Random4: An application specific randomized encryption algorithm to prevent SQL injection', Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012 (2012), pp.1327-1333, Viewed 24 June 2015. Available at:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6296134>

**Hardware/software needed -**

Modern PC, Oracle and MySQL DBMS

**Supervisor (if known) -**

Jun Li

**UNIVERSITY OF WOLVERHAMPTON**

**SCHOOL OF TECHNOLOGY**

**ETHICAL CONSIDERATION FOR STUDENTS STUDYING TAUGHT PROGRAMMES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Section 1: Your details** | | | |
| **First Name & Surname:** | James Braznell | **Student No:** | 1007022 |
| **Course:** | Computer Science | | |
| **Project Title** | SQL Injection: - The popular methods, solutions and is it still an issue | | |
| **Supervisor:** | Jun Li | | |

|  |  |
| --- | --- |
| **Section 2: Your Project Topic** | |
| **2.1** What problem is this project addressing? (100 words or less) | The issue of SQL Injection attacks, whether it is still an issue, the popular methods, solutions and how it can be dealt with better. |
| **2.2** Will information or artefact resulting from your project be available externally to the University? | Yes/**No**? |
| **2.2.1**  **If you answered ‘yes’ to 2.2,**  Will any such information place anyone at risk or possibly result in any action that might be detrimental to their wellbeing? (See guidelines) | Yes/No? |
| **2.2.2**  In what format will the information or artefact be made available? | In a word document submitted on WOLF |

|  |  |
| --- | --- |
| **Section 3: Method of Data Collection** | |
| **Please attach samples with this form if you intend to do interviews, surveys, or questionnaires.** | |
| **3.1** Does any part of your proposed project involve human participants?  If No go to Section 4. | Yes/**No**? |
| **3.1.1**  **If you answered ‘yes’ to 3.1,**  Is the sole involvement of human participants in order to provide opinions to support the specification or testing of an artefact to be produced as an outcome of the project? | Yes/No? |
| **3.1.2**  **If you answered ‘yes’ to 3.1.1,**  Does this artefact/information have any characteristics which might be detrimental to the wellbeing of any human participants in your project? If so, explain. | Yes/No? |
| **3.2**  **If you answered ‘yes’ to 3.1,**  Are there other ways you might meet your project aims without involving human participants? If not, why?  If yes discuss with your Supervisor how you will achieve this and go to section 4. |  |
| **3.2.1**  How will you select your participants? |  |
| **3.2.2**  How many participants will you contact? |  |
| **3.2.3**  How will you approach potential participants? E.g. email, letter, face to face? |  |
| **3.2.4**  Are your participants adults? (over 18 and competent to give consent) If no, answer 3.2.5 | Yes/No? |
| **3.2.5**  Are your participants children or adults over 18 and not competent to give consent? If yes, why is it necessary to involve these participants? (See guidelines)  Explain how you will ensure parental/guardian consent. |  |
| **3.2.6**  Are you offering any incentives to any of your participants, financial or otherwise? (See guidelines) | Yes/No? |
| **3.2.7**  How much time do you estimate will be needed from any participants? (See guidelines) |  |
| **3.2.8**  Please list the method of data collection and analysis intended to be used |  |
| **3.2.9**  Will all of the data collected contribute towards your results? |  |

|  |  |
| --- | --- |
| **Section 4: Confidentiality and data handling** | |
| **Please read methods of ensuring confidentiality in the guidelines.** | |
| **4.1** Will you ensure the anonymity of data collected from/and about participants? | **Yes**/No |
| **4.2** Will you store/protect data collected from individuals e.g. password protected files? | **Yes**/No |
| **4.3** Once your project is complete and information is no longer needed, will you destroy your data? | **Yes**/No |
| **4.4** Will anyone else have access to the data collected? | Yes/**No** |
| **If so,**  (i) please name the individuals and/or groups that will have access;  (ii) why is access being given to those listed in (i)? |  |

|  |  |
| --- | --- |
| **Section 5: Working with other parties and companies** | |
| **5.1** Will you be using data on subjects held by another party or organisation? | Yes/**No** |
| **If Yes,**  (i) Please give details.  (ii) How will you gain access to this information? |  |
| **5.2** Do you require written permission from a company, organisation or location, e.g. an employer or local authority? | Yes/**No** |
| **If Yes,**  (i) Please complete an [external agreement form](file://C:\Documents%20and%20Settings\in5541\Local%20Settings\Temporary%20Internet%20Files\in7475\Local%20Settings\Documents%20and%20Settings\in7475\Local%20Settings\Documents%20and%20Settings\in5505\Local%20Settings\Temporary%20Internet%20Files\Content.Outlook\0JYTHJ5A\External%20Computing%20Project%20Agreement%20Form.doc) and include this with your submission. |  |
| **NB: If working with another organisation or company please familiarise yourself with their Health & Safety procedures.** | |

**Things you must be aware of:**

**Data Protection Act**: <http://www.ico.gov.uk/what_we_cover/data_protection.aspx>

**Freedom of Information Act**: <http://www.opsi.gov.uk/Acts/acts2000/ukpga_20000036_en_1>

[University of Wolverhampton Ethical Approval Procedural Guidelines](http://www.wlv.ac.uk/PDF/aca-pols-ethics-scrutiny.pdf)

**Checklist:**

1. If you are using a questionnaire or interview sheet please include a list of sample questions with your submission.

2. In addition, please include an introductory cover letter stating some information about you, your project proposal and how your data will be used.

3. If you are undertaking a project involving a company or organisation you will need to show that you have approval from that organisation. Please include a completed copy of the [External Agreement Form](file://C:\Documents%20and%20Settings\in5541\Local%20Settings\Temporary%20Internet%20Files\in7475\Local%20Settings\Documents%20and%20Settings\in7475\Local%20Settings\Documents%20and%20Settings\in5505\Local%20Settings\Temporary%20Internet%20Files\Content.Outlook\0JYTHJ5A\External%20Computing%20Project%20Agreement%20Form.doc).

|  |  |
| --- | --- |
| **Student’s Declaration**  Sign and date against **one** declaration **only** | |
| **Category 0.** I have answered  ‘No’ to questions **2.2** and **3.1**.  My project involves no human participation except for myself and I agree to ensure that any information or artefact produced will not be available outside the University. | James Braznell  26/06/15 |
| **Category A1.** I have answered (delete one from each block)   |  | | --- | | ‘Yes’ to questions **2.2**, **4.1**, **4.2** and **4.3** and ‘No’ to questions **2.2.1** and **4.4**. | | ‘No’ to question **2.2** |   **and**   |  | | --- | | ‘Yes’ to questions **3.1**, **3.1.1** and **3.2.4** and ‘No’ to question **3.1.2** | | ‘No’ to question **3.1** |   My project involves limited human participation and I agree to ensure that   1. any such participation is not detrimental in any way to the interests of the participants; 2. all information collected as a part of the project will be handled in accordance with the answers that I gave to question **4**; 3. No information or artefacts which may place anyone at risk or be detrimental to their wellbeing will be made available outside the University. |  |
| **Category A2.** I have answered  ‘Yes’ to question **3.1**  **or** I have answered  ‘Yes’ to question **2.2**  and my answers to subsequent questions prevent the project being classified as A1.  My project involves human participation and may present some risk to participants. I have considered alternative means of pursuing the project which do not entail this risk but believe that there is no practicable alternative. I agree to ensure that I take all necessary steps to minimise risks to participants and third parties. I agree not to proceed with any activities involving human participation until I have received approval from the Department Ethics Panel. |  |
| **Category B-E.** My project does not conform to Category 0, A1 or A2. I have considered alternative means of pursuing the project which do not entail risk to human participants but believe that there is no practicable alternative to the proposal made. I agree to ensure that I take all necessary steps to minimise risks to participants. I agree not to proceed with any activities involving human participation until I have received approval from the School or University Ethics Committee, as appropriate. |  |

|  |  |
| --- | --- |
| **Supervisor’s Declaration**  Sign and date against **one** declaration **only** | |
| **Category 0 or A1.** I concur with the classification of this project as **0** or **A1** and authorise continuation of the project. I have forwarded a copy of this form to the Department Ethics Panel for monitoring purposes. |  |
| **Other.** I believe that this project should be classified other than **0** or **A1.** I do **not** authorise continuation of the project until approval has been received from the appropriate Ethics Panel or Committee. I have forwarded a copy of this form to the Department Ethics Panel for consideration. |  |

**FOR SUPERVISOR/PANEL/COMMITTEE USE ONLY:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLASSIFICATION ALLOCATED BY SUPERVISOR** | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0, A1** |  | **Supervisor Action:** Authorise and forward to DEP | | | | | | | | | | | | | | | | | | Date | | | |  | | | | |
| **DEP Action:** File for possible monitoring | | | | | | | | | | | | | | | | | | | Date | | | |  | | | | |
|  | | Selected for monitoring | | | |  | (tick) | | | | | | | | | | | | Date | | | |  | | | | |
| Classification agreed? | | | | Yes |  | No |  | | | If ‘No’, give: | | | | | | | | | | | | | | | |
| Reason | | | |  | | | | | | | | | | | | | | | | | | | | | |
| Action | | | |  | | | | | | | | | | | | | | | | | | | | | |
| **Other** |  | **Supervisor Action:** Refer to DEP for decision | | | | | | | | | | | | | | | | | | Date | | | |  | | | | |
| **CLASSIFICATION ALLOCATED BY DEPARTMENT ETHICS PANEL** | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0, A1** | |  | **DEP Action:** Project authorised to continue | | | | | | | | | | | | | | | | | Date | | | |  | | | |
| **A2** | |  | **Considered by DEP below** | | | | | | | | | | | | | | | | | Date | | | |  | | | |
| **2.2** Is any risk associated with access to project acceptable in context? If no, give reasons below: | | | | | | | | | | | | | | | | | Yes | | |  | | | No | |  |
|  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **3.1** Is involvement of human participants justified? If no, give reasons below: | | | | | | | | | | | | | | | | | Yes | | |  | | | No | |  |
|  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **3.3** Is experimental method acceptable with regard to risk and inconvenience to participants? If no, give reasons below: | | | | | | | | | | | | | | | | | Yes | | |  | | | No | |  |
|  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **4** Are arrangements for confidentiality and data protection appropriate? If no, give reasons below | | | | | | | | | | | | | | | | | Yes | | |  | | | No | |  |
|  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **5** Do arrangements for working with external bodies protect interests of participants and the external bodies? If no, give reasons below | | | | | | | | | | | | | | | | | Yes | | |  | | | No | |  |
|  | | | | | | | | | | | | | | | | | | | | | | | | | |
| **DEP Action:** Continuation of project approved: | | | | | | | | Yes | | |  | | No | |  | | Date | | | |  | | | | |
| Conditions: | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Other** | |  | **DEP Action:** Refer to School Ethics Committee | | | | | | | | | | | | | | | | | Date | | | |  | | | |
| **CLASSIFICATION ALLOCATED BY SCHOOL ETHICS COMMITTEE** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0, A1** | |  | **SEC Action:** Continuation of project approved | | | | | | | | | | | | | | Date | | | |  | | | | |
| **A2, B** | |  | **Considered by SEC below** | | | | | | | | | | | | | | Date | | | |  | | | | |
| **2.2** Is any risk associated with access to project acceptable in context? If no, give reasons below: | | | | | | | | | | | | | | | Yes | | |  | | | | No |  |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| **3.1** Is involvement of human participants justified? If no, give reasons below: | | | | | | | | | | | | | | | Yes | | |  | | | | No |  |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| **3.3** Is experimental method acceptable with regard to risk and inconvenience to participants? If no, give reasons below: | | | | | | | | | | | | | | | Yes | | |  | | | | No |  |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| **4** Are arrangements for confidentiality and data protection appropriate? If no, give reasons below | | | | | | | | | | | | | | | Yes | | |  | | | | No |  |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| **5** Do arrangements for working with external bodies protect interests of participants and the external bodies? If no, give reasons below | | | | | | | | | | | | | | | Yes | | |  | | | | No |  |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| **SEC Action:** Continuation of project approved: | | | | | | | Yes | | |  | | No | |  | | | Date | |  | | | | |
| Conditions: | | | | | | | | | | | | | | | | | | | | | | | |
| **Other** | |  | **SEC Action:** Refer to University Ethics Committee | | | | | | | | | | | | | | | | Date | |  | | | | |

**Guidelines**

**Section 1: Categorisation for ethical approval**

**Category 0:** There are no third parties directly involved in the project and any artefacts produced by the project will not be accessible to a general audience.

**Category A1**

Projects involving human volunteers are involved solely for the purposes of:

- providing data to inform the specification of an artefact

- testing the usability or fitness for purpose of an artefact

where the nature of that artefact or its use will present no risk to the volunteers

and, if any artefact is accessible to a general audience, access to that artefact will present no risk.

**Category A2**

Projects involving human volunteers other than those defined in category A1 but not in activities defined in other categories or if any artefact is accessible to a general audience, access to that artefact may present some risk.

**Category B**

Projects involving human volunteers including potential risk, for instance,

studies using new research methodologies, studies involving certain vulnerable

populations or therapeutic interventions or other significant risk to anyone involved in

the research (but not including trials of artefacts intended for therapeutic purposes).

**Category C**

Research being conducted by staff or postgraduate research students involving

Patients, clients staff, records etc. within the sphere of the NHS, Social Services, etc (but not including clinical trials of medicinal or related products).

**Category D**

Research being conducted by undergraduate or taught postgraduate students involving

Patients, clients staff, records etc. within the sphere of the NHS, Social Services, etc (but not including clinical trials of medicinal or related products).

**Category E**

Clinical trials of medicinal or related products involving patients or healthy volunteers as direct users of the product.

**Question 2.2:** You should answer yes if your artefact, product or information might be of direct risk or might lead or encourage people to alter their behaviour in a way which would be detrimental to them. Examples of direct potential risk might be a machine that could injure someone if it malfunctioned or a web resource which contained information which if it was misused would lead to risk (for instance, children’s identities or addresses). Examples of artefacts which might encourage detrimental behaviour could be a web resource offering alternatives to expert (such as GP or lawyer) advice or products which purport to have a therapeutic effect.

**Question 3.2.5:** As a general principle, all participants should be informed of their role in the experiment and freely consent (in writing) to it, which implies competence to give consent. Very occasionally it may be necessary to undertake an experiment without consent, or with participants who are not competent but then any decision about the acceptability of the proposal would be taken on the basis of the absolute benefit of the experiment in a wider context, and it would have to be established that there was no alternative.

**Question 3.2.6:** With regard to freedom of consent, it likely that this principle would be breached of the participants were subject to some kind of inducement or coercion, however minor. For instance, it is likely that participants who were under the management of the person undertaking the experiment would be considered to be under a degree of coercion.

**Question 3.2.7:** It may be considered that expecting a participant to spend undue time or effort participating in an experiment would be detrimental to the interests of that person, particularly where the results of the work offered no clear benefits. It may be appropriate to compensate participants for their time, but it is not acceptable to offer inducements to participate.

**Section 4 Anonymity:**

It is to be expected that due care and attention be paid to protecting information about individuals. Depending on the nature of the experiment, the following may be considered.

* Type 1: Complete anonymity of participants (i.e., You will not meet, or know the identity of participants, as they are part of a random sample and are required to return responses with no form of personal identification)?
* Type 2: Anonymised samples or data (i.e., an irreversible process whereby identifiers are removed from data and replaced by a code, with no record retained of how the code relates to the identifiers. It is then impossible to identify the individual to whom the sample of information relates)?
* Type 3: De-identified samples or data (i.e., a *reversible* process whereby identifiers are replaced by a code, to which you retain the key, in a secure location)?
* Type 4: Subjects being referred to by pseudonym in any publication arising from the project?
* Type 5: Any other method of protecting the privacy of participants? (eg. use of direct quotes with specific, written permission only; use of real name with specific, written permission only)

# Appendix 2 – User guide

If you produced software that is intended for others to use, or that others may wish to extend/improve, then a user guide and an installation guide appendices are ***essential***.

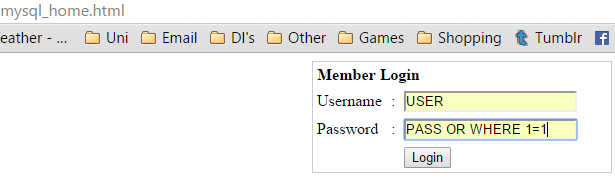
# Appendix 3 – Installation guide

If you produced software that is intended for others to use, or that others may wish to extend/improve, then a user guide and an installation guide appendices are ***essential***.

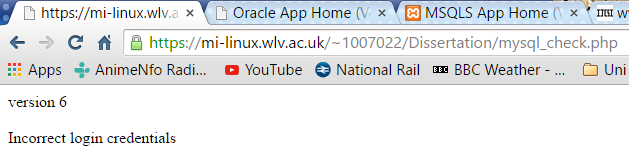
# Appendix 4 – Artefact - DBMS SQLIA testing

## Tautologies

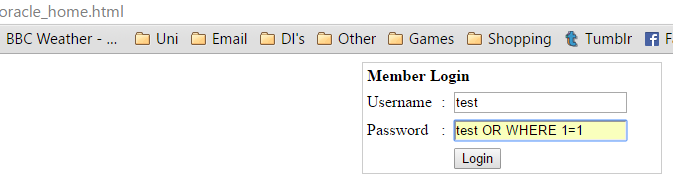
### MySQL tautology attack



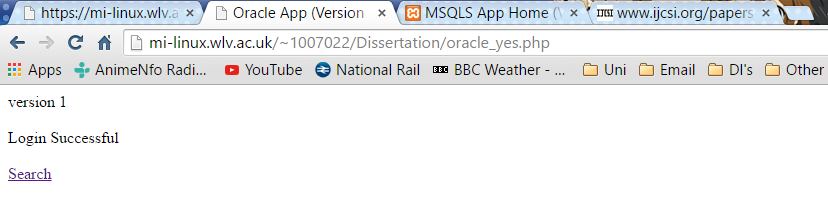
### MySQL tautology attack results



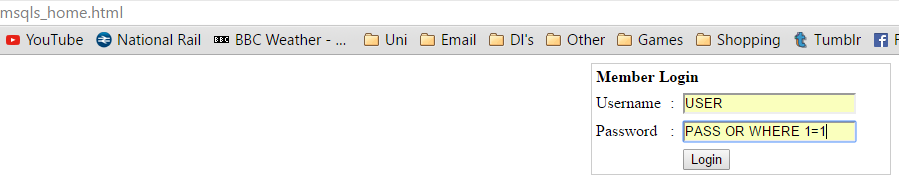
### Oracle tautology attack



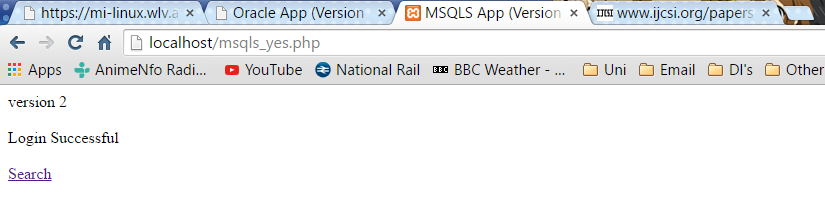
### Oracle tautology attack results



### MSQLS tautology attack

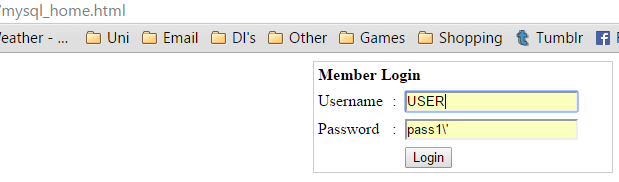


### MSQLS tautology attack results



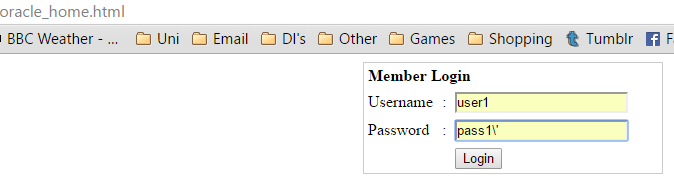
## Illegal/Logically Incorrect Queries

### MySQL Illegal/Logically Incorrect Queries attack



### MySQL Illegal/Logically Incorrect Queries attack results

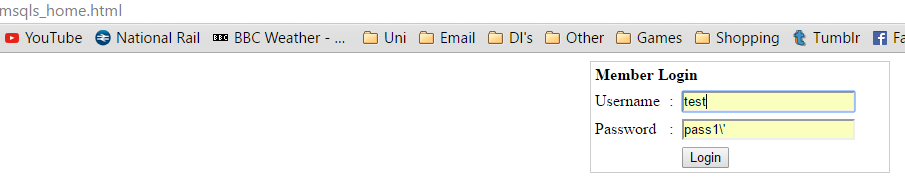
### Oracle Illegal/Logically Incorrect Queries attack



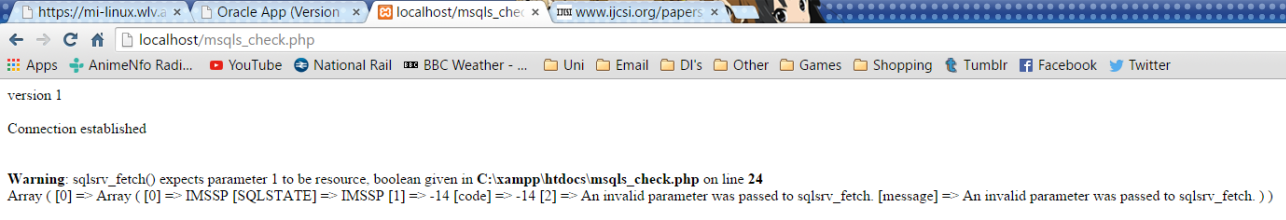
### Oracle Illegal/Logically Incorrect Queries attack results



### MSQLS Illegal/Logically Incorrect Queries attack

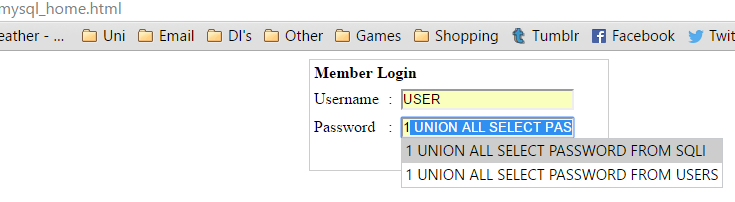


### MSQLS Illegal/Logically Incorrect Queries attack results

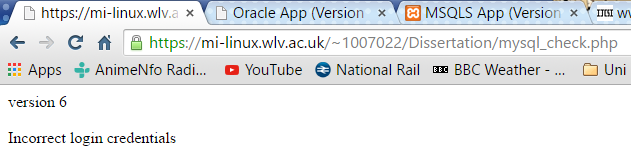


## Union Query

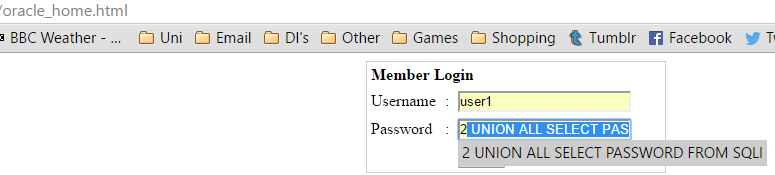
### MySQL Union Query attack



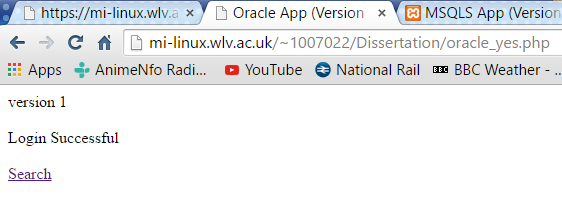
### MySQL Union Query attack results



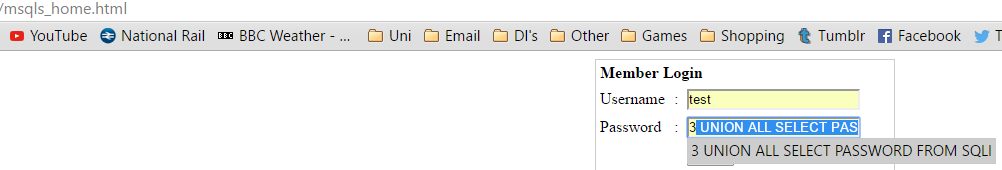
### Oracle Union Query attack



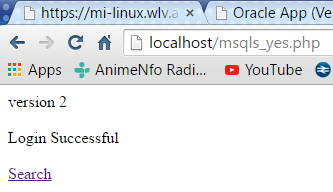
### Oracle Union Query attack results



### MSQLS Union Query attack

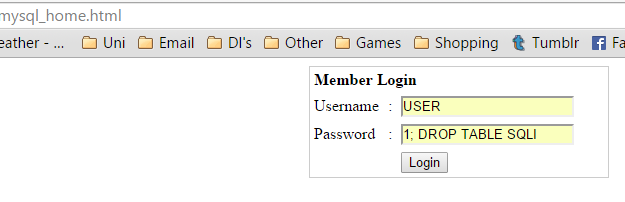


### MSQLS Union Query attack results

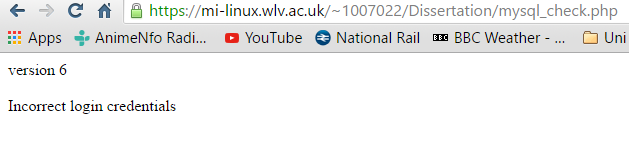


## Piggy-backed queries

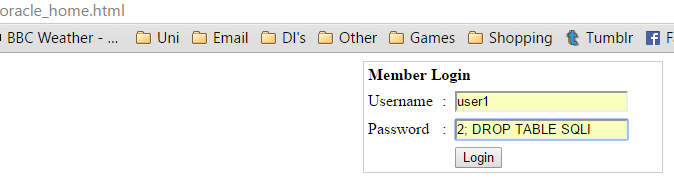
### MySQL Piggy-backed Queries attack



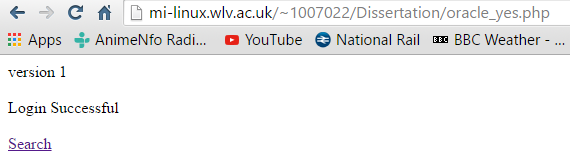
### MySQL Piggy-backed Queries attack results



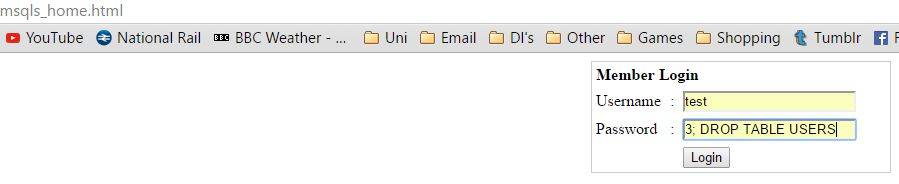
### Oracle Piggy-backed Queries attack



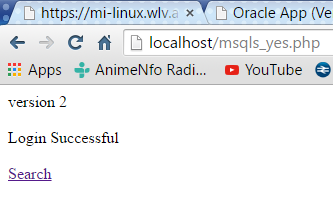
### Oracle Piggy-backed Queries attack results



### MSQLS Piggy-backed Queries attack

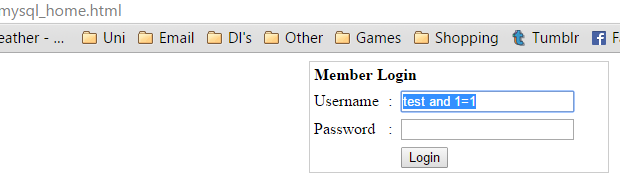


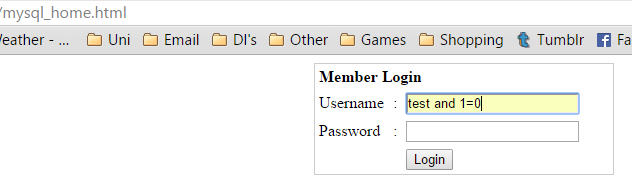
### MSQLS Piggy-backed Queries attack results



## Blind Injection

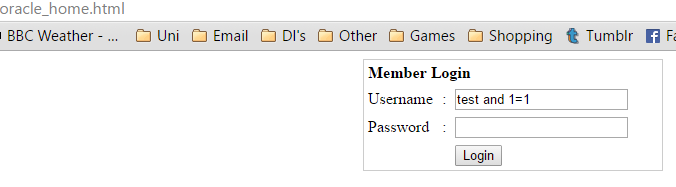
### MySQL Blind Injection attack

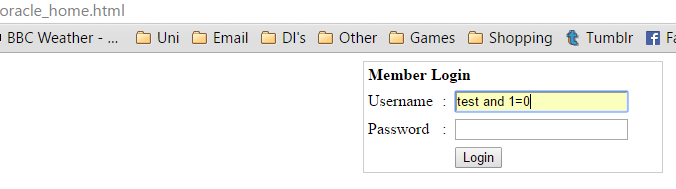




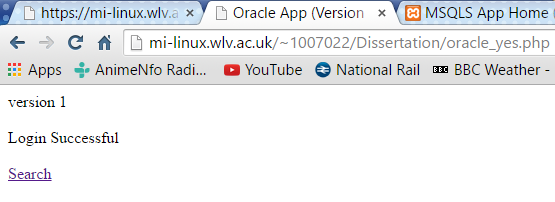
### MySQL Blind Injection attack results

### Oracle Blind Injection attack

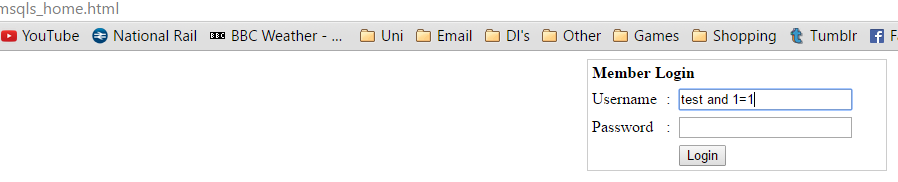


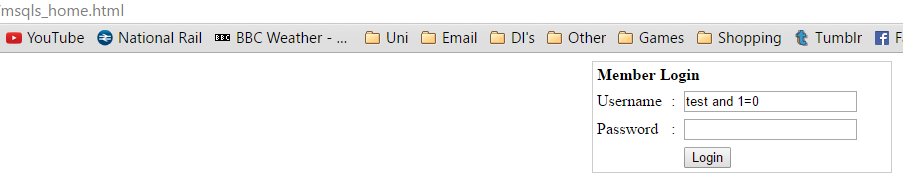


### Oracle Blind Injection attack results

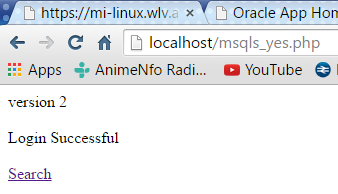


### MSQLS Blind Injection attack



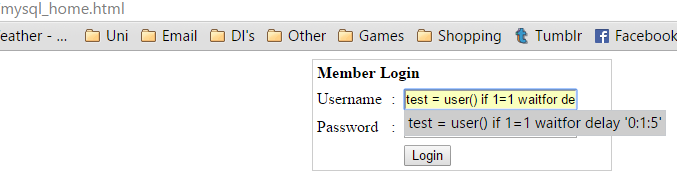


### MSQLS Blind Injection attack results

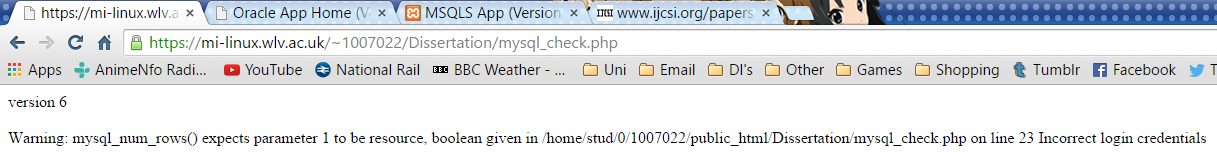


## Timing Attack

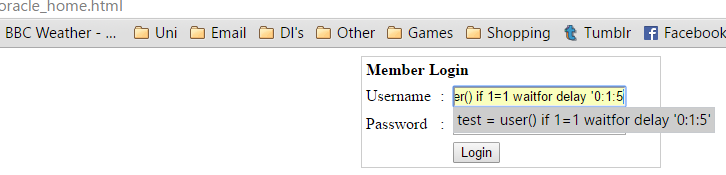
### MySQL Timing Attack



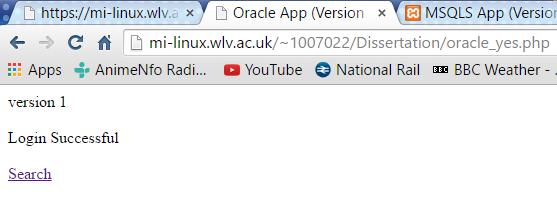
### MySQL Timing Attack results



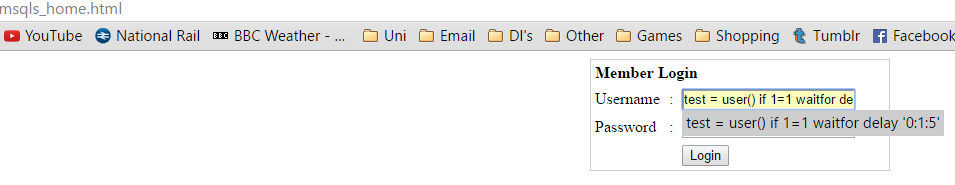
### Oracle Timing Attack



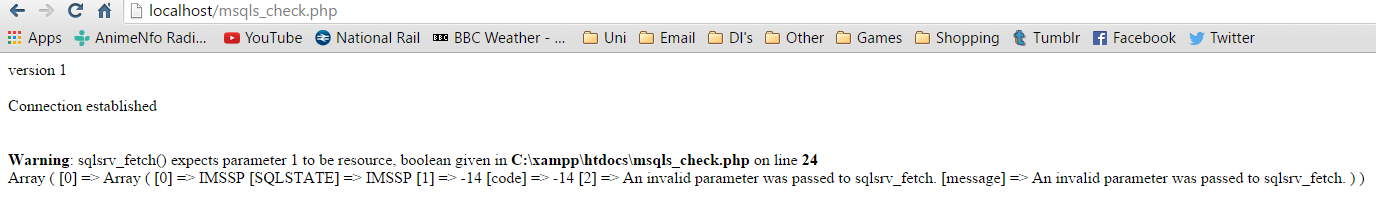
### Oracle Timing Attack results



### MSQLS Timing Attack

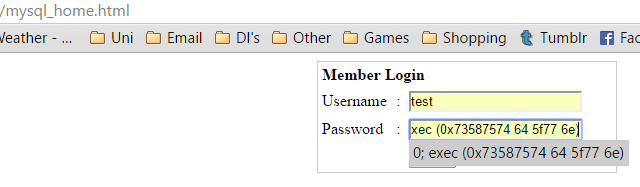


### MSQLS Timing Attack results

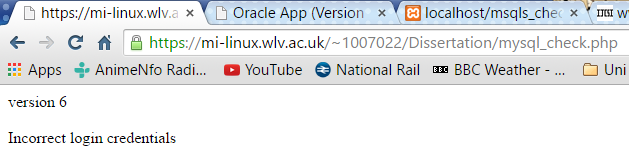


## Alternate Encodings

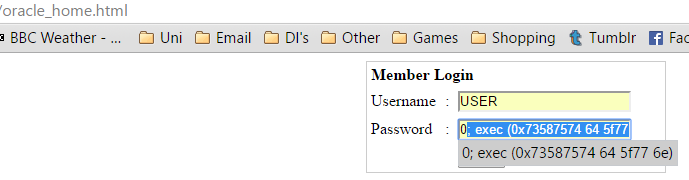
### MySQL Alternate Encodings Attack



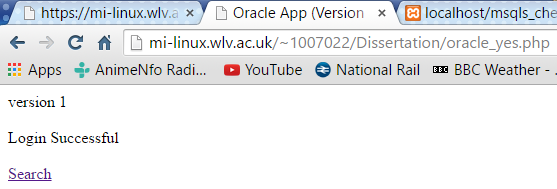
### MySQL Alternate Encodings Attack results



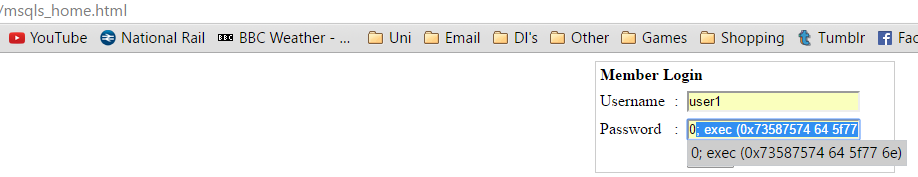
### Oracle Alternate Encodings Attack



### Oracle Alternate Encodings Attack results



### MSQLS Alternate Encodings Attack

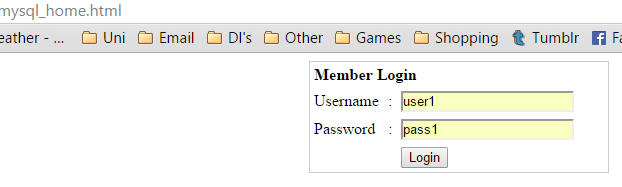


### MSQLS Alternate Encodings Attack results

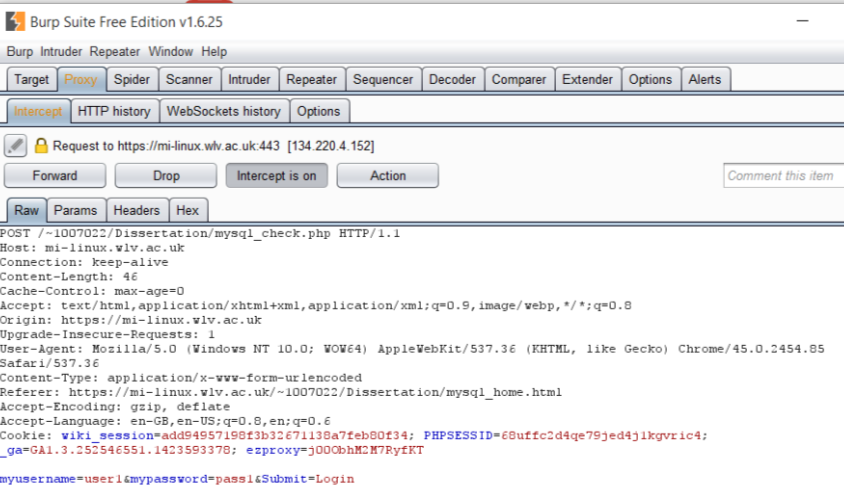
## SQLIA results

# Appendix 5 – Artefact - Burp Suite testing

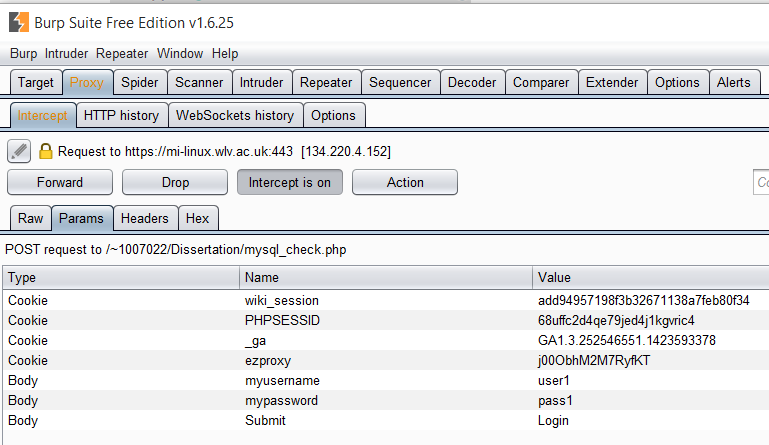
## Login request sent

****

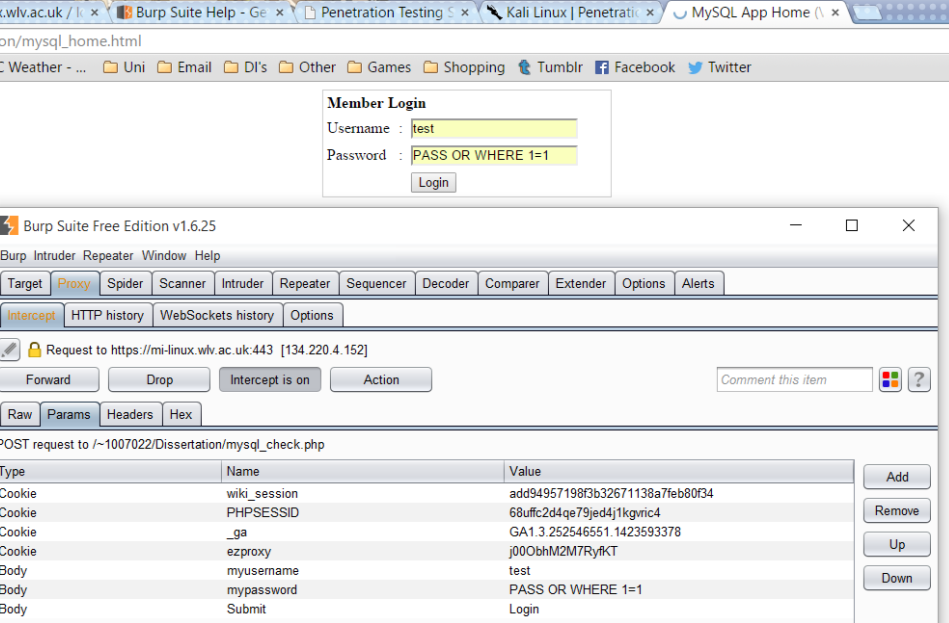
## Request detection



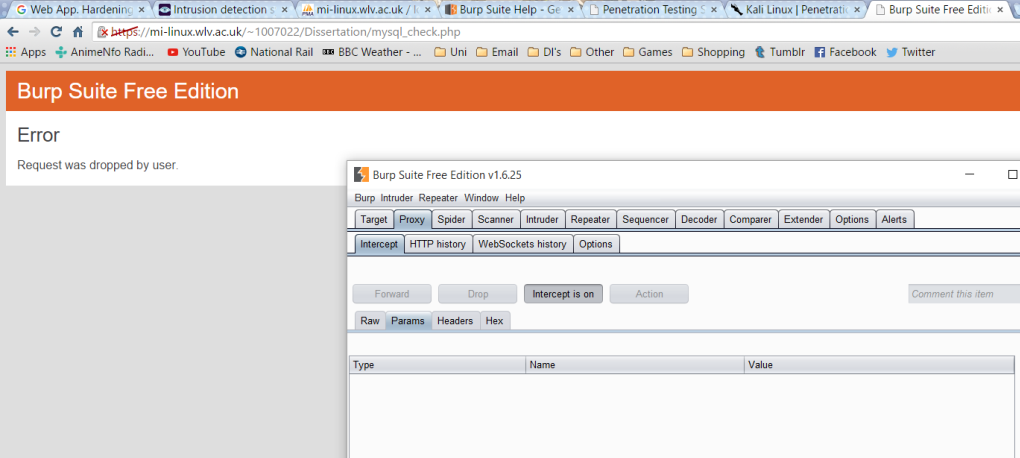
## Input shown



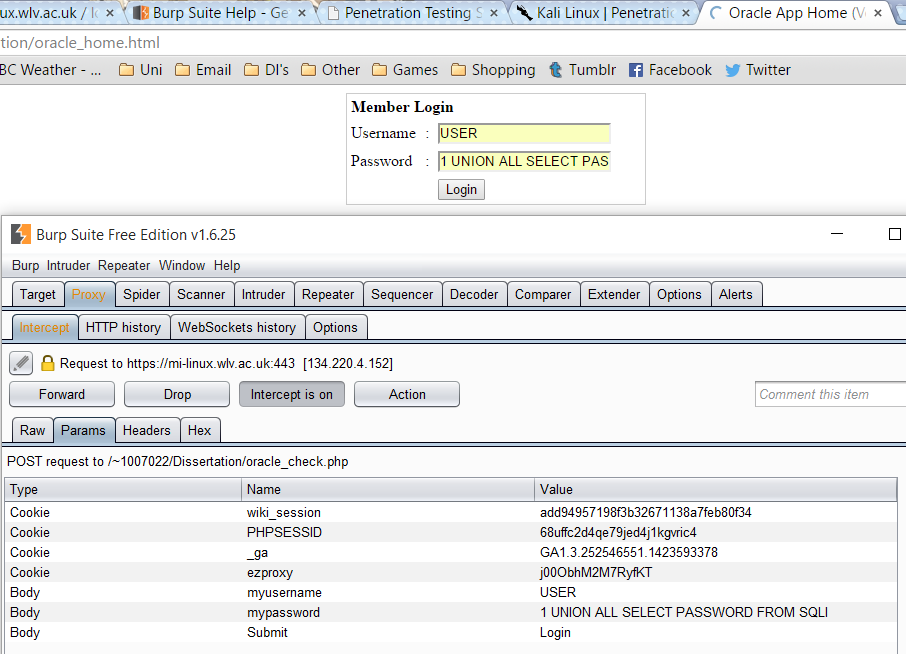
## SQLIA intercepted

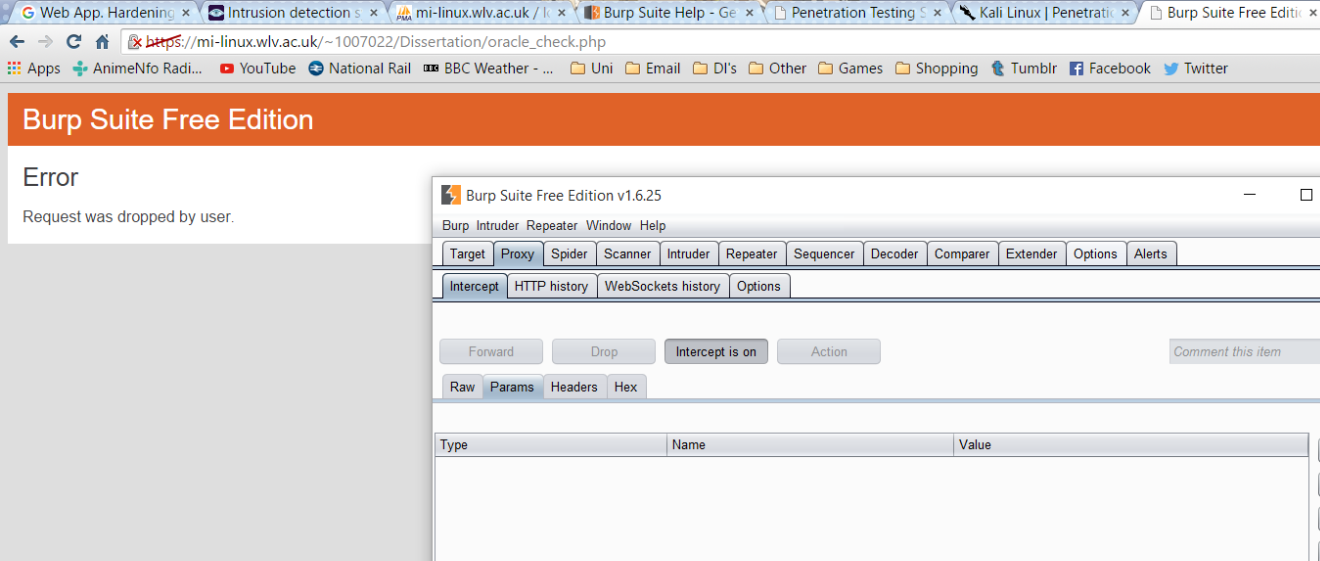


## MySQL SQLIA blocked

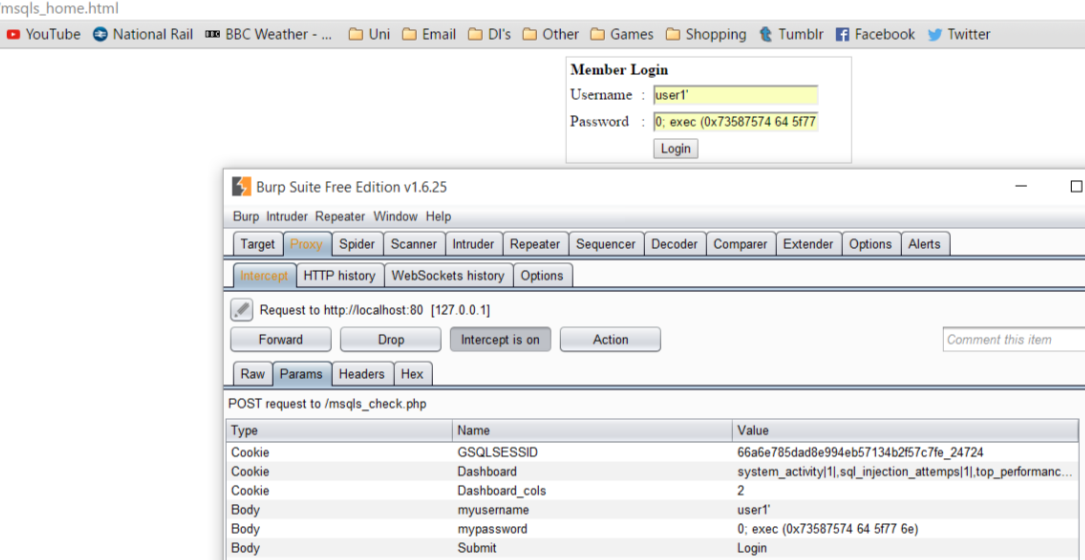


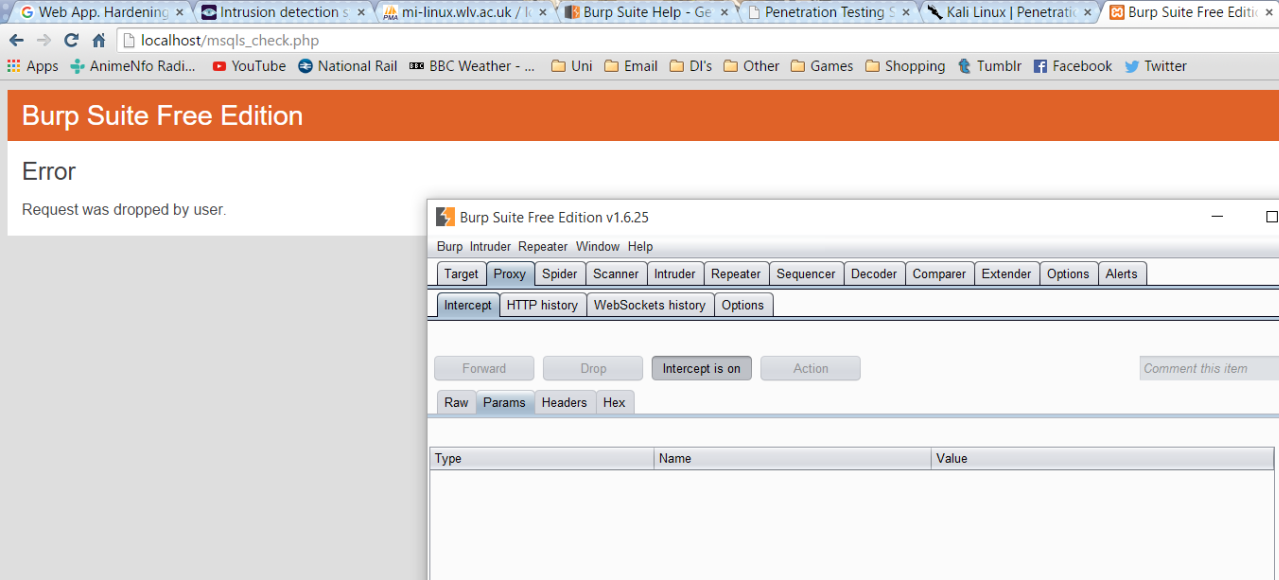
## Oracle SQLIA blocked





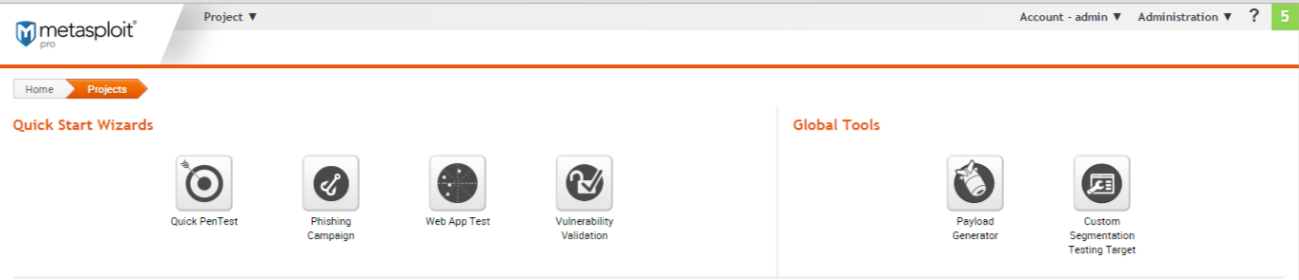
## MSQLS SQLIA blocked



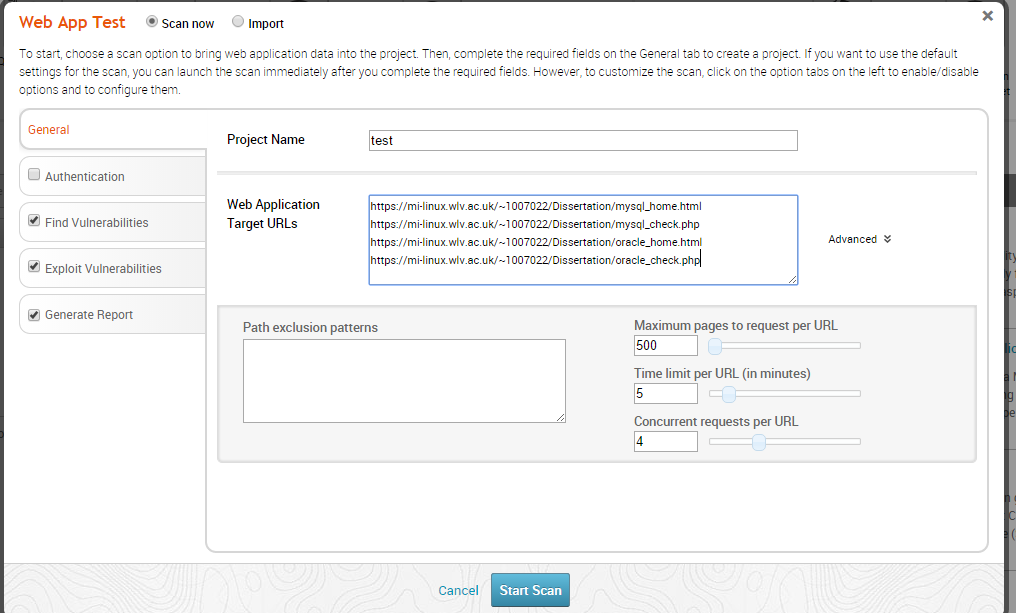


# Appendix 6 – Artefact - Metasploit testing

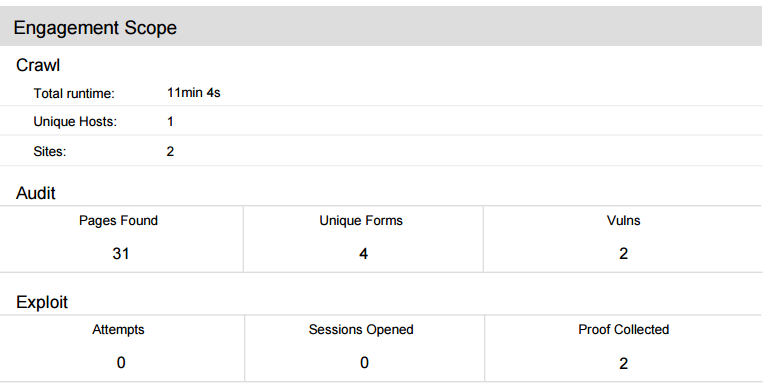
## Metasploit testing options



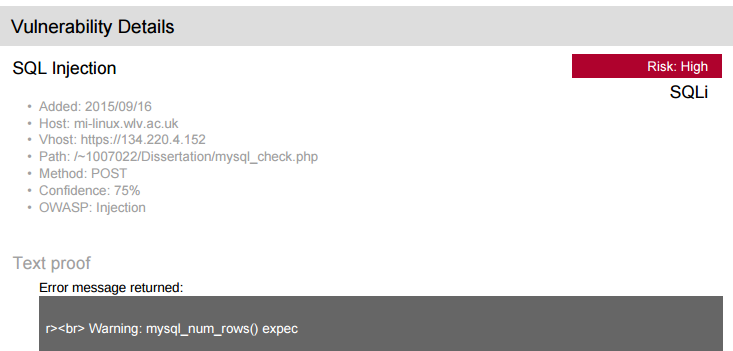
## Web app test configuration

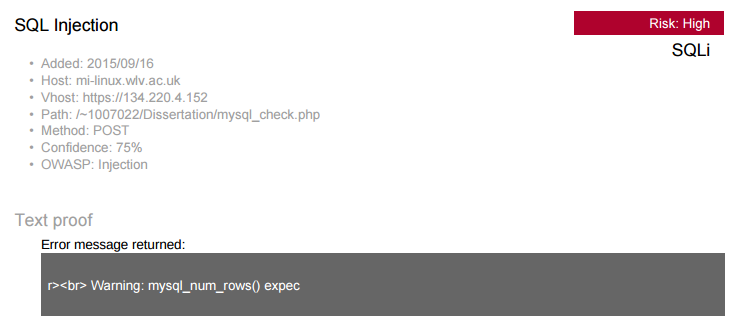


## Metasploit testing results



## Metasploit SQLI vulnerabilities detected





# Appendix 7 – Artefact - Microsoft SQL server application

## msqls\_home.html

<html>

<head><title>MSQLS App Home (Version 1)</title></head>

<body>

<table width="300" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">

<tr><form name="form1" method="post" action="msqls\_check.php"><td>

<table width="100%" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">

<tr><td colspan="3"><strong>Member Login </strong></td></tr>

<tr>

<td width="78">Username</td>

<td width="6">:</td>

<td width="294"><input name="myusername" type="text" id="myusername"></td>

</tr>

<tr>

<td>Password</td>

<td>:</td>

<td><input name="mypassword" type="text" id="mypassword"></td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

<td><input type="submit" name="Submit" value="Login"></td>

</tr></table></td></form></tr></table></body></html>

## msqls\_check.php

<?php

echo "version 1" . "<br><br>";

$serverName = "JAMESBRAZNELL\SQLEXPRESS";

$connectionInfo=array("Database"=>"SQLI");

/\* Connect using Windows Authentication. \*/

$conn = sqlsrv\_connect( $serverName, $connectionInfo);

if( $conn ) {

echo 'Connection established<br><br>';

}else{

echo 'Connection failed<br><br>';

die( print\_r( sqlsrv\_errors(), true));

}

// username and password sent from form

$myusername=$\_POST['myusername'];

$mypassword=$\_POST['mypassword'];

// Performs select query

$sql="SELECT \* FROM USERS WHERE USERNAME='$myusername' and PASSWORD='$mypassword'";

$results=sqlsrv\_query( $conn, $sql);

if( sqlsrv\_fetch( $results ) === false) {

die( print\_r( sqlsrv\_errors(), true));

}

// If there is a result go forward

if($results>=1){

// Go to "msqls\_yes.php"

header("location:msqls\_yes.php");

}

else {

echo "Incorrect login credentials";

}

?>

## msqls\_yes.php

<?php

echo "version 2";

?>

<html>

<head><title>MSQLS App (Version 2)</title></head>

<body><br><br>

Login Successful

<br><br>

<a href= "msqls\_app.php">

Search

</a>

</body>

</html>

## msqls\_app.php

<?php

$serverName = "JAMESBRAZNELL\SQLEXPRESS";

$connectionInfo=array("Database"=>"SQLI");

$conn = sqlsrv\_connect( $serverName, $connectionInfo);

if( $conn ) {

echo 'Connection established<br><br>';

}else{

echo 'Connection failed<br><br>';

die( print\_r( sqlsrv\_errors(), true));

}

$sql = "SELECT \* FROM USERS WHERE USERNAME = 'User1'";

$results = sqlsrv\_query( $conn, $sql);

if( $results === false ) {

die( print\_r( sqlsrv\_errors(), true));

}

if( sqlsrv\_fetch( $results ) === false) {

die( print\_r( sqlsrv\_errors(), true));

}

$user = sqlsrv\_get\_field( $results, 0);

echo "$user: ";

$pass = sqlsrv\_get\_field( $results, 1);

echo $pass;

sqlsrv\_close( $conn );

?>

# Appendix 8 – Artefact - MySQL application

## mysql\_home.html

<html>

<head><title>MySQL App Home (Version 4)</title></head>

<body>

<table width="300" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">

<tr><form name="form1" method="post" action="mysql\_check.php"><td>

<table width="100%" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">

<tr><td colspan="3"><strong>Member Login </strong></td></tr>

<tr>

<td width="78">Username</td>

<td width="6">:</td>

<td width="294"><input name="myusername" type="text" id="myusername"></td>

</tr>

<tr>

<td>Password</td>

<td>:</td>

<td><input name="mypassword" type="text" id="mypassword"></td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

<td><input type="submit" name="Submit" value="Login"></td>

</tr></table></td></form></tr></table></body></html>

## mysql\_check.php

<?php

echo "version 6" . "<br><br>";

$host="localhost";

$username="1007022";

$password="thomas123";

$db\_name="db1007022";

$tbl\_name="SQLI";

mysql\_connect("$host", "$username", "$password")or die("cannot connect");

mysql\_select\_db("$db\_name")or die("cannot select DB");

$myusername=$\_POST['myusername'];

$mypassword=$\_POST['mypassword'];

$sql="SELECT \* FROM $tbl\_name WHERE User='$myusername' and Pass='$mypassword'";

$result=mysql\_query($sql);

$count=mysql\_num\_rows($result);

if($count>=1){

header("location:mysql\_yes.php");

}

else {

echo "Incorrect login credentials";

}

?>

## mysql\_yes.php

<?php

echo "version 3";

?>

<html>

<head><title>MySQL App (Version 3)</title></head>

<body><br><br>

Login Successful

<br><br>

<a href= "https://mi-linux.wlv.ac.uk/~1007022/Dissertation/mysql\_app.php">

Search

</a>

</body>

</html>

## mysql\_app.php

<?php

echo "version 3";

print("<br><br>");

$dbServName = "localhost";

$dbUserName = "1007022";

$dbPassword = "thomas123";

$dbName = "db1007022";

$con = new mysqli($dbServName, $dbUserName, $dbPassword, $dbName);

if ($con->connect\_error) {

die("Connection failed: " . $con->connect\_error);

}

$sql = "SELECT \* FROM `SQLI` WHERE `User` = 'user1'";

$result = $con->query($sql);

if ($result->num\_rows > 0) {

while($row = $result->fetch\_assoc()) {

echo "Name: " . $row["User"]. "<br>" . "Pass: " . $row["Pass"]. "<br>";

}

} else {

echo "0 results";

}

$con->close();

?>

# Appendix 8 – Artefact - Oracle application

## oracle\_home.html

<html>

<head><title>Oracle App Home (Version 1)</title></head>

<body>

<table width="300" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">

<tr><form name="form1" method="post" action="oracle\_check.php"><td>

<table width="100%" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">

<tr><td colspan="3"><strong>Member Login </strong></td></tr>

<tr>

<td width="78">Username</td>

<td width="6">:</td>

<td width="294"><input name="myusername" type="text" id="myusername"></td>

</tr>

<tr>

<td>Password</td>

<td>:</td>

<td><input name="mypassword" type="text" id="mypassword"></td>

</tr>

<tr>

<td>&nbsp;</td>

<td>&nbsp;</td>

<td><input type="submit" name="Submit" value="Login"></td>

</tr></table></td></form></tr></table></body></html>

## oracle\_check.php

<?php

echo "version 1" . "<br><br>";

$username = 'ops$1007022';

$password = 'jbraznell123';

$conn = oci\_connect($username, $password, '(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = oradb-srv.wlv.ac.uk)(PORT = 1522)) (CONNECT\_DATA = (SID=orcle12c)))');

$myusername=$\_POST['myusername'];

$mypassword=$\_POST['mypassword'];

$sql="SELECT \* FROM SQLI WHERE USERNAME='$myusername' and PASSWORD='$mypassword'";

$result=oci\_parse($conn, $sql);

$count=mysql\_num\_rows($result);

if($result>=1){

header("location:oracle\_yes.php");

}

else {

echo "Incorrect login credentials";

}

?>

## oracle\_yes.php

<?php

echo "version 1";

?>

<html>

<head><title>Oracle App (Version 1)</title></head>

<body><br><br>

Login Successful

<br><br>

<a href= "https://mi-linux.wlv.ac.uk/~1007022/Dissertation/oracle\_app.php">

Search

</a>

</body>

</html>

## oracle\_app.php

<?php

echo "version 1" . "<br><br>";

$username = 'ops$1007022';

$password = 'jbraznell123';

$conn = oci\_connect($username, $password, '(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = oradb-srv.wlv.ac.uk)(PORT = 1522)) (CONNECT\_DATA = (SID=orcle12c)))');

$results = oci\_parse($conn, "SELECT \* FROM SQLI where USERNAME = 'user1'");

$r = oci\_execute($results);

while ($row = oci\_fetch\_array($results, OCI\_ASSOC+OCI\_RETURN\_NULLS)) {

foreach ($row as $item) {

print ($item !== null ? htmlentities($item, ENT\_QUOTES) : "&nbsp;") . "\n";

}

}

?>