

|  |
| --- |
| **Web Application Vulnerabilities**    **Luke Burgess – 1703091.uad.ac.uk**  Introduction to Security – CMP110  BSc Ethical Hacking Year 1  2017/18 |

*Note that Information contained in this document is for educational purposes.*

Abstract

This is a report on common web application vulnerabilities which covers both the common exploits used by hackers as well as a few countermeasures to protect applications and databases. These have been researched and tested by the writer with evidence shown throughout the report. It was found that simple methods used can go a long way to preventing theft or loss of data.

Contents

[1 Introduction 1](#_Toc512902696)

[1.1 Background 1](#_Toc512902697)

[1.2 Aim 2](#_Toc512902698)

[2 Procedure 3](#_Toc512902699)

[2.1 Setup 3](#_Toc512902700)

[2.2 Overview of Procedure 6](#_Toc512902701)

[2.3 Procedure part 1 – SQL Injection 1 6](#_Toc512902702)

[2.4 Procedure part 2 - SQL Injection 2 8](#_Toc512902703)

[2.5 Procedure part 3 – Stored XSS (Cross Site Scripting) 12](#_Toc512902704)

[2.6 Procedure part 4 – XSS Reflected (Cross Site Scripting) 13](#_Toc512902705)

[3 Results/Conclusions 15](#_Toc512902706)

[3.1 Results for part 1 15](#_Toc512902707)

[3.2 Results for part 2 15](#_Toc512902708)

[3.3 Results for part 3 15](#_Toc512902709)

[3.4 Results for part 4 17](#_Toc512902710)

[3.5 Countermeasures 17](#_Toc512902711)

[3.6 Conclusions 17](#_Toc512902712)

[3.7 Future Work 18](#_Toc512902713)

[4 Bibliography 19](#_Toc512902714)

# Introduction

## Background

Today many businesses have vastly improved their security infrastructure. However, many companies are exploited by hackers due to simple flaws present in their web applications, software and like products. Large-scale multinational companies such as Sony, Yahoo, and TalkTalk have all fallen victim to exploits including but not limited to SQL injection and XSS (cross-site scripting).

According to (theguardian.com, 2018) TalkTalk was hacked on 15th October 2015 in which 150,000 customers had their personal information leaked by a 19-year-old male called Daniel Kelley. The Guardian goes on to state: “SQL injection is well understood, defenses exist and TalkTalk ought to have known it posed a risk to its data.”.

As stated by (engadget.com, 2018), “When some users changed their password via the TalkTalk website, the new value was stored in the plaintext.” The best practice when it comes to web development is to ‘hash’ and ‘salt’ a user’s password. If a password is ‘hashed’ then as the user creates a new user account or attempts to log in the password inputted goes through a secure hashing function such as the Secure Hashing Algorithm 2 (SHA-2). This function ensures that the password is never stored in plain English within the database. Hashing algorithms such as SHA-2 are known as ‘one-way’ functions. A ‘one-way’ function ensures that a hashed password cannot be reverse engineered to find out the original un-hashed password. Although methods such as ‘dictionary attacks’ or ‘brute force attacks’ do attempt to un-hash an already hashed password. Due to this, the ‘salting’ method adds an extra layer of protection. The ‘salt’ is an extra input added to the password prior to hashing without the user’s knowledge so that two identical un-hashed passwords will be unique once they are stored in the database after being ‘salted' and ‘hashed'. This means that should a hacker attempt to crack a password, the process would take vast amounts more time, which could have given TalkTalk the luxury of informing users of the database breach and could have protected many accounts from harm.

According to (owasp.com, 2018), code injection and XSS (cross-site scripting) are in the Owasp top ten most critical web application security risks. Code injection ranks number one in both 2013 and 2017. XSS ranked third in 2013 and seventh in 2017. OWASP stands for Open Web App Security Project. It is a wiki site designed to be a free and Open Source tool to catalogue web application vulnerabilities.

SQL is called Structured Query Language and is used to create and manipulate the structure and information stored within databases. It uses statements such as SELECT, INSERT, DROP and DELETE to categorise and manipulate information within a relational database management system (RDBMS). SQL is run on a server and is usually combined with a server-side language such as Node.js or PHP.

JavaScript or ECMAScript is a client-side programming language designed to be run on the browser of the user rather than on a server such as PHP. It is an event-driven and procedural programming language very similar to the likes of C++ and C.

## Aim

The aims of this project are to go into detail about common web application exploits and to gain a better understanding of both how they are executed and how they can be prevented to avoid data manipulation, theft or loss.

This will be explored by legally carrying out SQL injection and XSS on a self-run server with materials provided specifically for IT professionals learning how to hack and using online tutorial games designed for educational purposes.

# Procedure

## Setup

The Damn Vulnerable Web App requires an Apache server to run. A free tool called UWamp sets up a local Wamp Server. This can be downloaded directly from UWamp: (https://www.uwamp.com/en/, 2018). This allows the user to set up an apache server with PHP and MySQL which is a dialect of SQL on their localhost (127.0.0.1). Other users on the same network can view files hosted on the WAMP server.

Once the user has downloaded the zip file and extracted it. The extracted folder can be opened, and the executable file can be run. The window shown in figure 0.1 will then show.

The user can be sure the server has been started by checking that the start button on the top left of the screen is greyed out. Figure 0.2 below shows the application should the wamp server of been set up correctly.

The Damn Vulnerable Web App can also be downloaded from its own website: (http://www.dvwa.co.uk/, 2018). Another zip file will be downloaded called DVWA-master.zip. Once DVWA-master is extracted the DVWA file may be copied to UWamp.exe. There should be a folder called www. Place the DVWA folder here. If the steps were followed correctly, the user may type 127.0.0.1/DVWA into a browser and the database set up screen will be displayed as per Figure 0.3.

The Damn Vulnerable Web Application thereafter gives remaining setup instructions to the user.

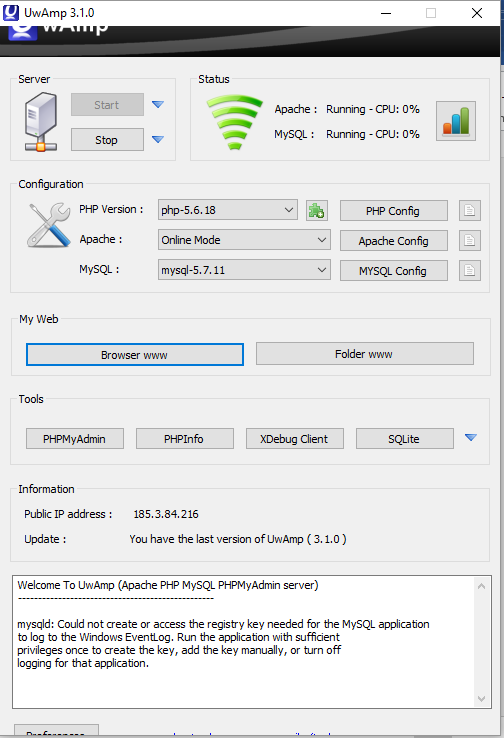


Figure 0.1

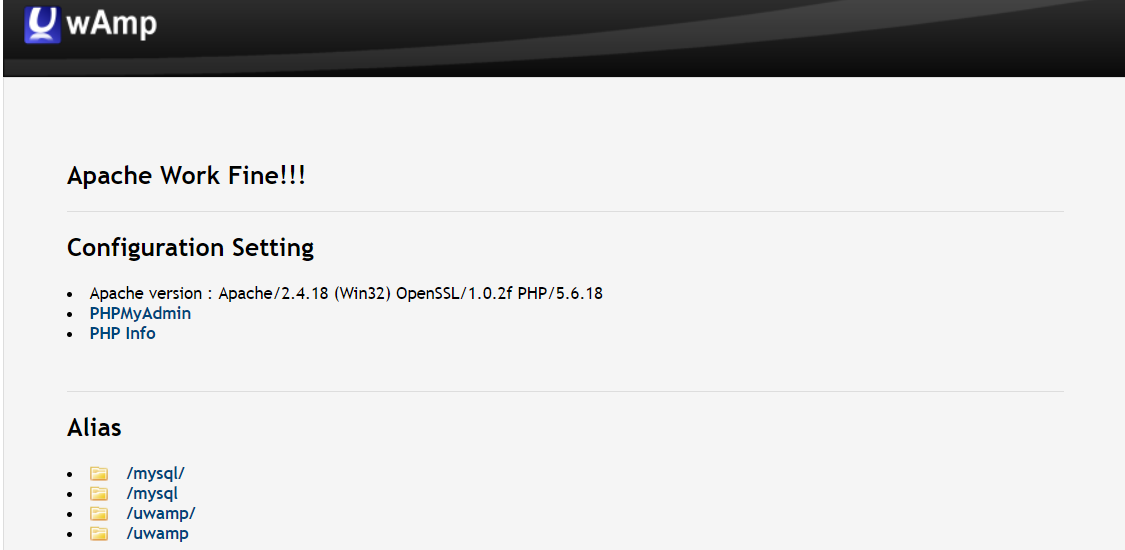


Figure 0.2

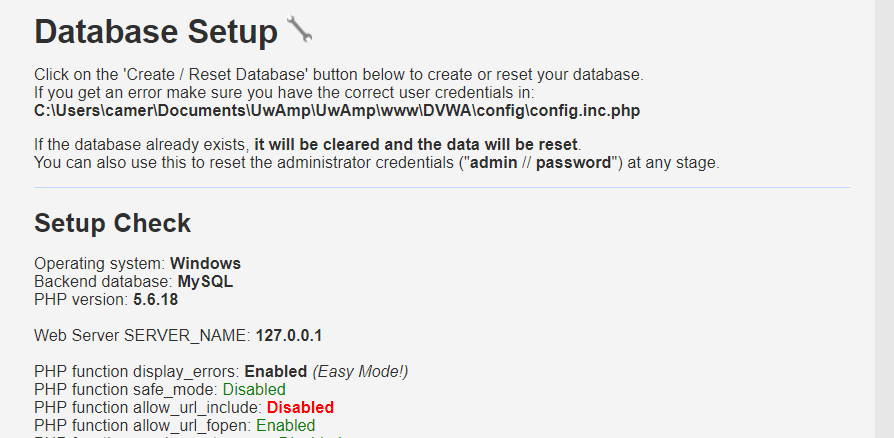


Figure 0.3

## Overview of Procedure

All procedures will take the reader through practical means to carry out SQL Injections and XSS (Cross Site Scripting). This is intended as a step by step guide catered for users with some technical skills and basic knowledge of SQL and JavaScript.

The first procedure uses the SQL injection educational game/tutorial (https://sqlzoo.net/hack/, 2018) which takes the user through the steps required to hack an individual user’s login information.

The second procedure uses Damn Vulnerable Web Application (http://www.dvwa.co.uk/, 2018) via the use of a UWamp wamp server and is used to attempt to manipulate many fields within an SQL database.

## Procedure part 1 – SQL Injection 1

The first exploit makes use of (https://sqlzoo.net/hack/, 2018) and is as follows:

Firstly the single quote ‘ is entered into the name input box to test if the web application will accept SQL statements inputted by the user. If the web application is vulnerable then the user will likely receive a server or SQL syntax error message similar to that shown in figure 1.1 below.

An SQL statement can then be inputted. ‘ OR’ ‘=’ can be used in both name and password fields to end the statement reading user input and adds the condition or true, which in turn means it will always return as one, or true. In this instance, it takes the first user in the database and successfully logs into that user which is "jake”. Please see figure 1.2 for reference.

Now that the username is known, specific statements can be used to find out if the password for the user contains a certain letter e.g. ' OR EXISTS(SELECT \* FROM users WHERE name='jake' AND password LIKE '%w%') AND ''=' which checks if the password contains the letter ‘w'. While it may not be the most efficient method to find a user password, this can be tested against each letter of the alphabet, numbers 0-9 and special characters to determine what characters make up the password for this user. In doing so it was found the password contained the letters "L", "O", "W", "E", "D". This, of course, could contain any number of the same character used e.g. ‘Wooled’. A simple scrabble solver search of “LLOOWWEEDD” finds all dictionary words containing those letters including words with two instances one letter. On testing, none of the dictionary words containing all five letters was the correct password. The next step taken was to use the following command: ' OR EXISTS(SELECT \* FROM users WHERE name='jake' AND password LIKE '\_\_w%') AND ''=' which tests if the third letter of the password was ‘w’ this can be altered to test for any of the five letters in any position of the password. In doing so each position at a time the password can be discovered.

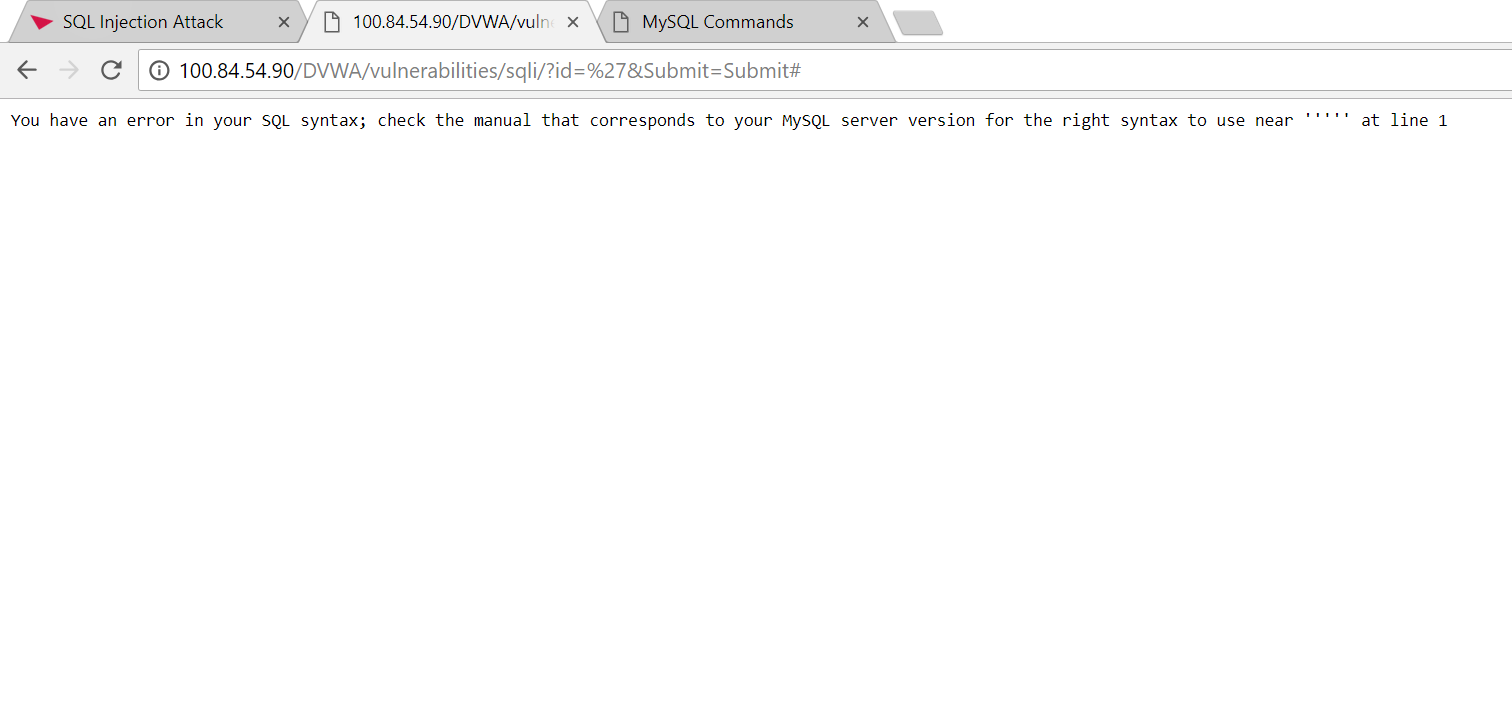


Figure 1.1



Figure 1.2



Figure 1.3

## Procedure part 2 - SQL Injection 2

The second exploit uses Damn Vulnerable Web Application (http://www.dvwa.co.uk/, 2018) and makes use of a UWamp wamp server. Please follow the diagrams below as you read as this aids understanding of each SQL statement and its effect.

Firstly the ‘ OR’ ‘=’ can be used again, this time in just the one available field. This prints all the first and second names of users as shown in figure 2.1. The command is shown in figure 2.2 ' UNION SELECT 1,2 FROM dual# thereafter reveals that the select statement is looking for 2 columns and is a test to ensure that the SQL injection was valid. This is a modified command from the Computerphile youtube channel video (https://www.youtube.com/watch?v=ciNHn38EyRc&t=615s, 2018).

The statement ' UNION SELECT TABLE\_NAME, TABLE\_SCHEMA FROM information\_schema.tables# prints the names of all the tables and the name of the MySQL database they relate to as shown in figure 2.3. The table called “users” stands out as the one most likely to contain personal information such as user ids and passwords. The user table can now be called in a statement to glean further information regarding columns within. The statement 'UNION SELECT COLUMN\_NAME,2 FROM information\_schema.columns WHERE TABLE\_NAME='users'# lists all columns for the “users” table as shown in figure 2.4 and enables the last statement to be used to gain full information about all users. Entering 'UNION SELECT user, password FROM users# as shown in figure 2.5 lists the username and password for every registered person. This completes the exploit as with each command you can access all information relating to this database. Other commands such as DROP TABLE etc. could be used to further alter the database from here.

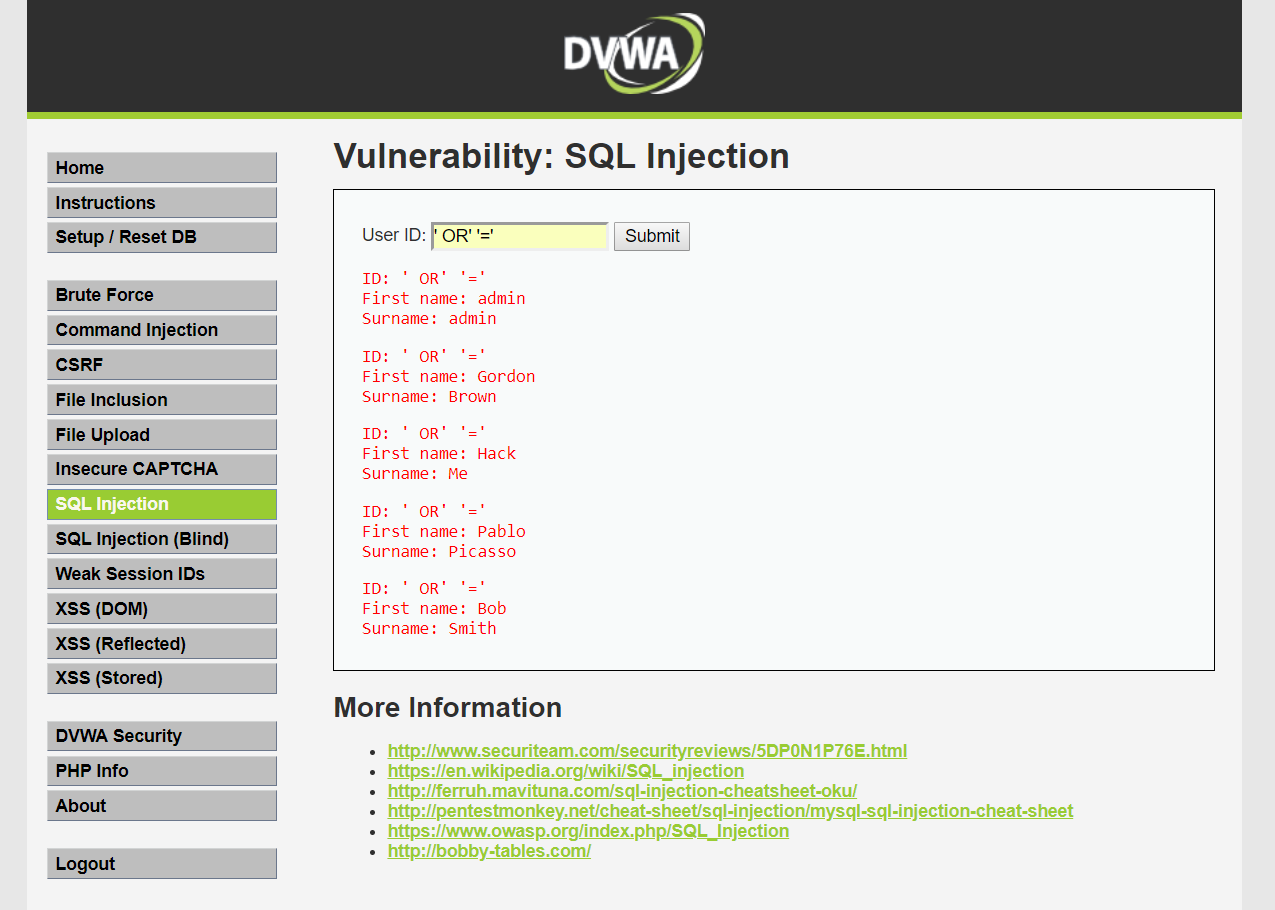


Figure 2.1

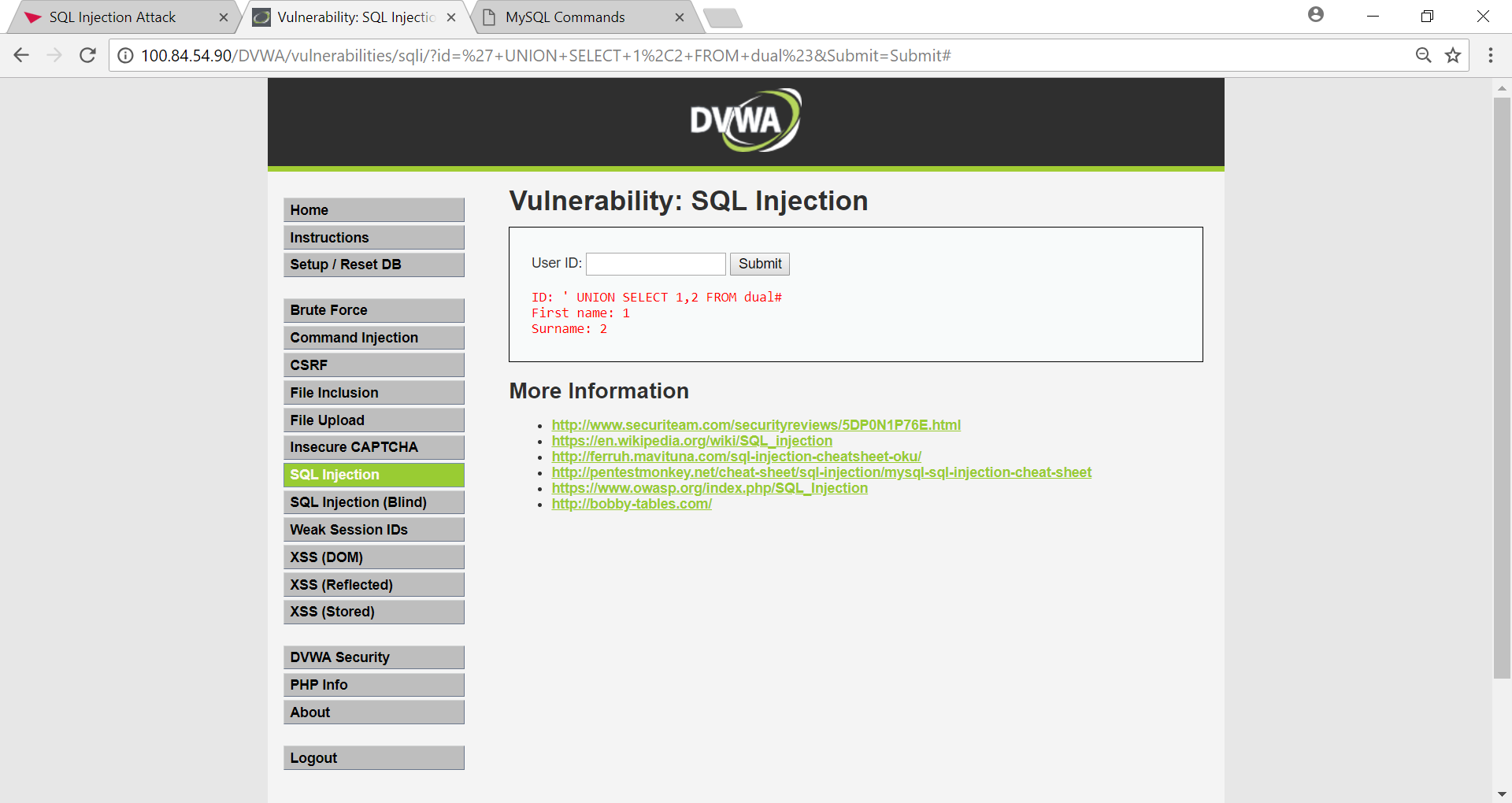


Figure 2.2

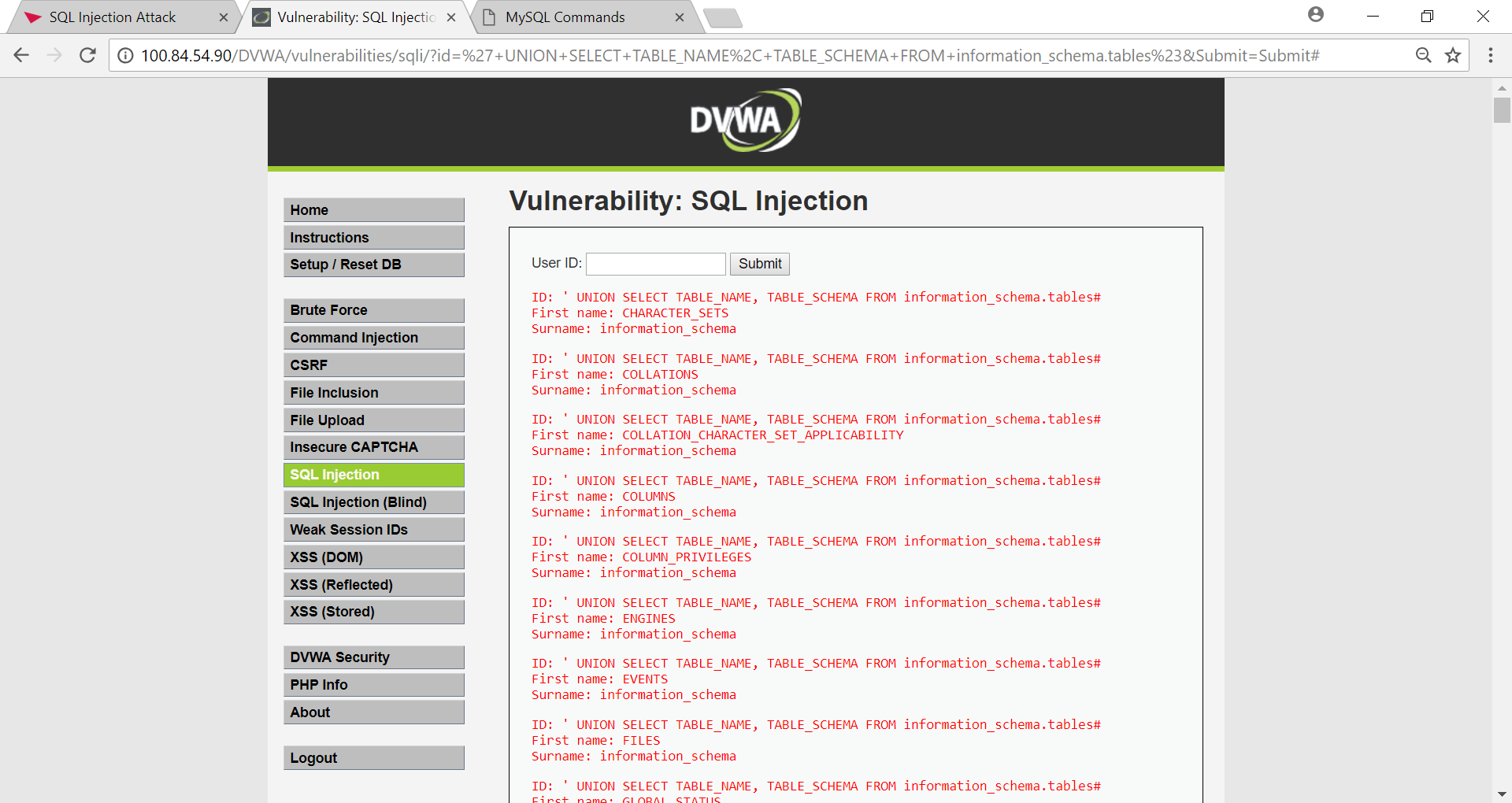


Figure 2.3

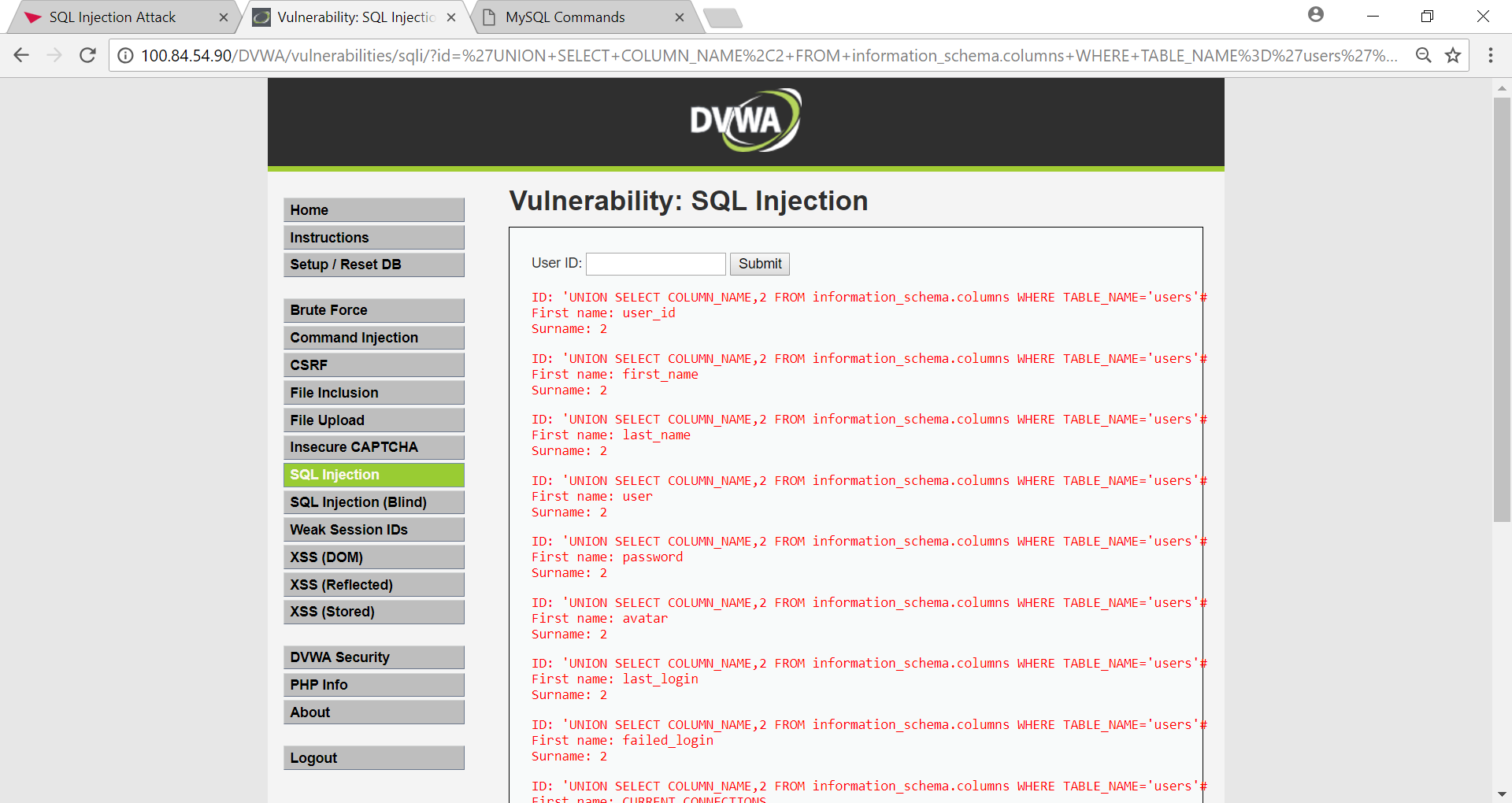


Figure 2.4

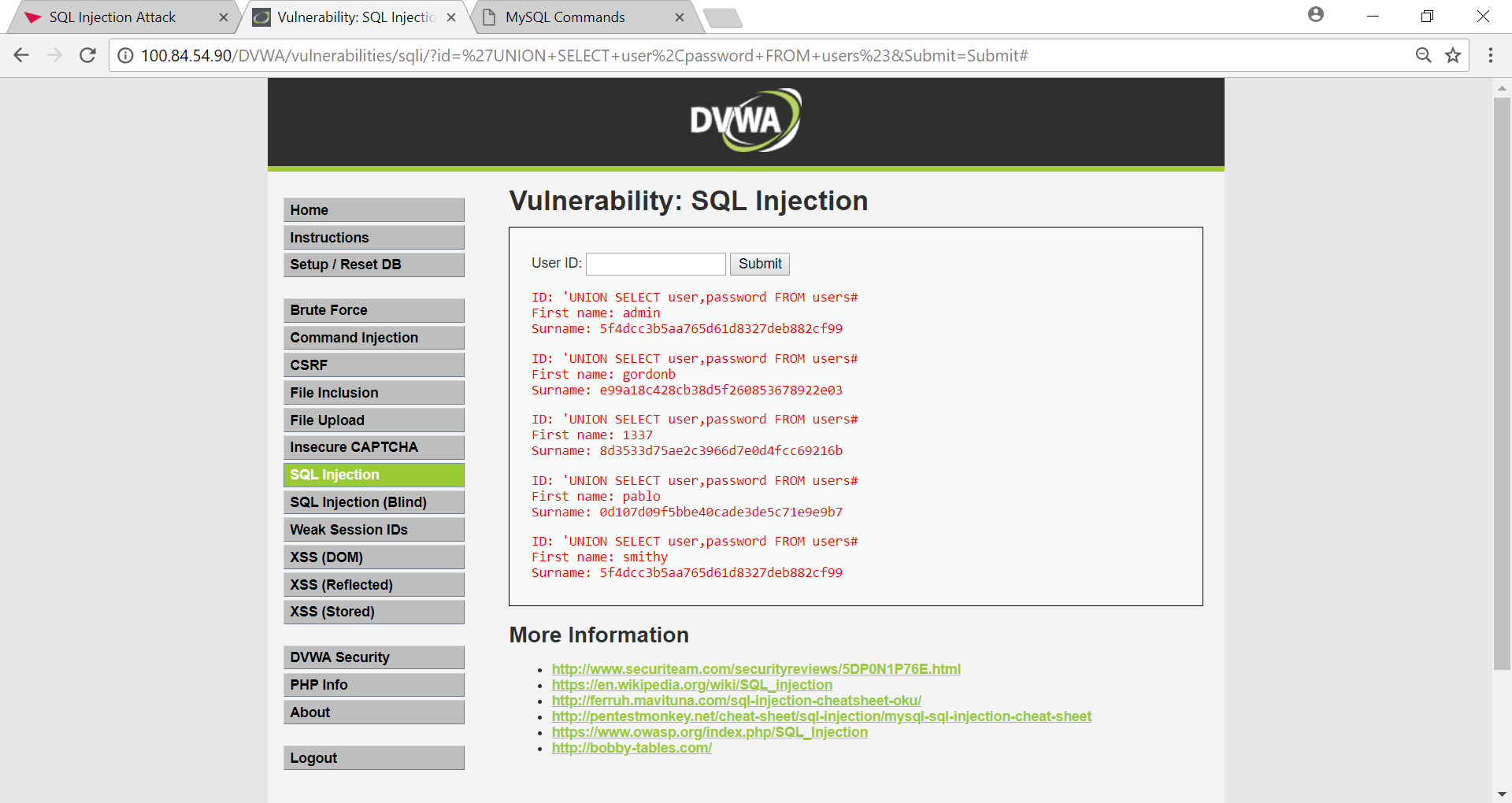


Figure 2.5

## Procedure part 3 – Stored XSS (Cross Site Scripting)

The first XSS exploit uses <http://127.0.0.1/DVWA/vulnerabilities/xss_s/> which is the stored XSS. This means that a hacker has embedded their javascript code into a comment box or other input field where other users can see what the hacker has embedded into this field and the victim's browser will be tricked into running that hackers malicious code.

The first step to this exploit is checking if the comment box is vulnerable to XSS. A HTML tag can be inserted into the comment section to check if the input of this comment box will render the inserted HTML tag as shown in figure 3.1. If rendered, this indicates that the web developer did not properly sanitise input into the comment boxes.

If it is confirmed that the target input field is vulnerable to XSS the next step would be to try and insert <script> tags into the comment box as shown in figure 3.2. This outputs a message box to the screen and further confirms that there is XSS vulnerability in the targeted website. Anything within the <script> tags are not rendered in the browser and are effectively invisible which in turn makes this attack very hard to notice unless you are an admin searching through the database or a technical user studying the source of the website.

Following this, a final payload can be inserted as per figure 3.3 to redirect the user’s browser to Rick Astley’s: ‘Never Gonna Give You Up song’. This is a light-hearted method of showing how a browser can execute code without the user's consent and potentially without the user’s knowledge. This could clearly be adapted for malicious purposes.

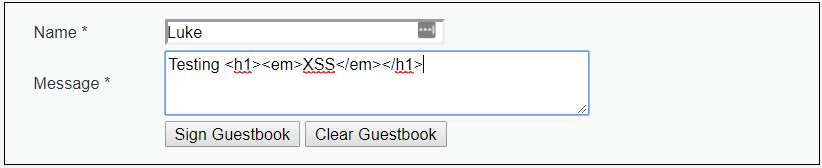


Figure 3.1

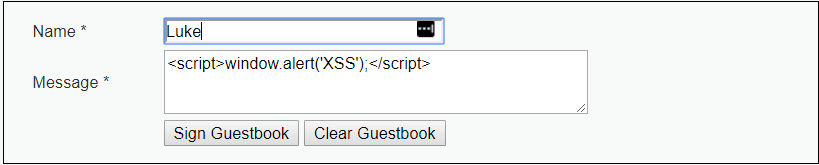


Figure 3.2

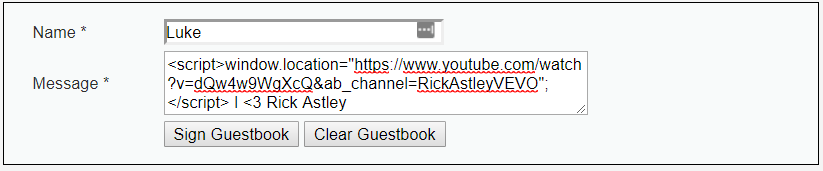


Figure 3.3

## Procedure part 4 – XSS Reflected (Cross Site Scripting)

The second exploit uses <http://127.0.0.1/DVWA/vulnerabilities/xss_r/> which is reflected XSS. This example is an attacker crafting an XSS exploit on the target's browser via a targeted phishing attempt also known as ‘spear phishing'.

The first step to this exploit is to craft the XSS payload. This will print out a message box to the victim’s browser.

Although the full URL can be sent to the victim, it is very common to use a URL shortener to change the link and hide the script tags from the user. The full-length URL can be seen in figure 4.1 and the shortened version is as shown in figure 4.2.

When a victim clicks the malicious link their browser then runs the code and a message box is outputted to the user as expected. Please see figure 4.3 for reference.



Figure 4.1

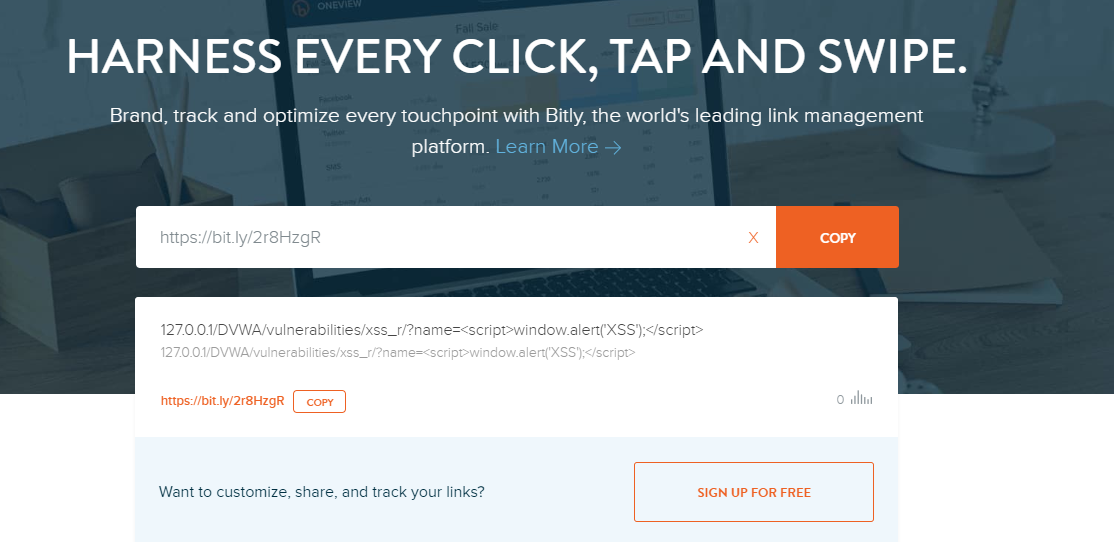


Figure 4.2

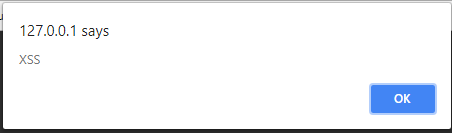


Figure 4.3

# Results/Conclusions

## Results for part 1

For the first part, the goal was to identify a registered user on the database by using SQL injection and thereafter find out their password by using a problem-solving approach to injection. Following all steps taken, the user's individual username was discovered to be "jake” and his password was “elwood”. This task was completed successfully as the intended goal was achieved.

## Results for part 2

For part two, the intentions were to find out personal information for many users without knowing their usernames to begin with. After using different resources to alter SQL statements to make them relevant to this database, it was possible to join tables and discover vast amounts of information relating to the database structure and ultimately all user data including usernames and passwords among many other pieces of personal information.

## Results for part 3

For part three, the intentions were to carry out a stored XSS exploit. The result of the first test is that the text ‘XSS’ is an emphasised header. Meaning that the text XSS should be bigger than the surrounding text and slightly bolder than the surrounding text as shown in figure 5.1.

The result of the second test is to prove that code inside of the <script> tags work. This confirms that the web app is vulnerable to XSS (shown in figure 5.2).

The result of the final test proves that the code inside of the <script> tags work and carry out the task programmed by the hacker. With this being the case, the user will be redirected to a video of Rick Astley’s: Never Gonna Give You Up song (please see figure 5.3). The user is redirected and anything outside of the script tags are treated as a normal comment and is displayed to the user as per figure 5.4.

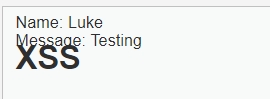


Figure 5.1

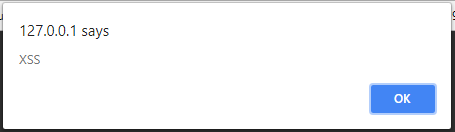


Figure 5.2

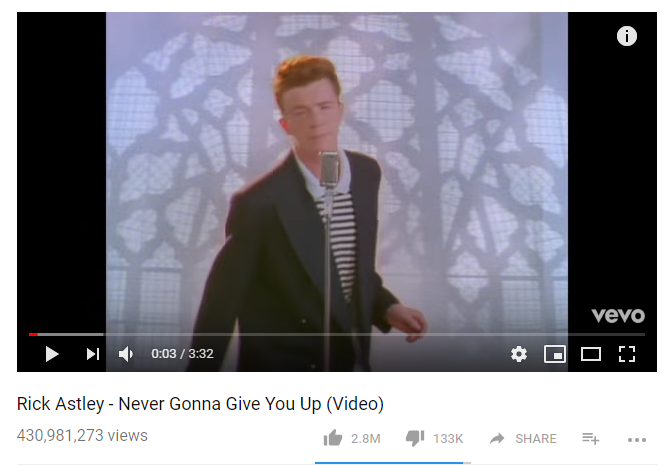


Figure 5.3

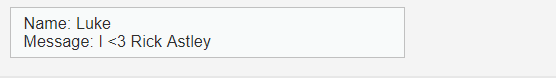


Figure 5.4

## Results for part 4

The XSS Reflected or ‘spear phishing technique was a success. After clicking the shortened version of the URL link, the victim was redirected as expected and a message box opened stating that there is an XSS vulnerability in the website.

## Countermeasures

There are many different countermeasures to protect web applications from different attacks such as SQL injection and Cross-site scripting. Some of commonly used with be listed below.

Prepared statements are a method used in PHP to prevent malicious code from being injected into an input field to access or alter the database. This works by sending the database a template of a query without the actual specific data. The database then has this information and the specific data for the query can be sent later which can be executed once or multiple times with the same structure. The application will take a user input and store it as a variable before use within an SQL statement. A validation function (e.g. mySQLi: real\_escape\_string) will be used to ensure this doesn’t contain special characters that may be read as code.

Escaping input is a method to prevent cross-site scripting techniques from tricking the victim’s browser into executing malicious javascript code. This changes HTML tags such as < and > to &lt and &gt. This means that the browser will not see the angle brackets as code as opposed to the literal characters.

You can also validate the code, not just the client side but also on the server side. This is because if you validate only on client side then the user can turn off the scripts that are used to validate the input by disabling javascript. The server-side validation would reject any invalid input.

## Conclusions

The work carried out forming this report proves how easy it can be to learn basic Injection techniques to compromise the security of a web application. It is a stark reminder of how vital a strong security infrastructure is for every web application accessible by the internet or internal networks. In addition to this, the number of companies that do not take this seriously enough is very alarming.

The report highlights the need for stricter regulations to be enforced for all web applications and related products owned by businesses.

## Future Work

With more time, further work could be put into practicing countermeasure tactics to test how effective they are and what weaknesses they may have despite greatly improving the security of a web application.

Further research into blind SQL injection to develop these specific hacking skills and understanding would be ideal, as well as looking into XSS Document Object Model (DOM) which involves modifying elements of a web application.

# Bibliography

engadget.com, 2018. *TalkTalk hack: what you need to know.* [Online]   
Available at: https://www.engadget.com/2015/10/23/talktalk-hack-explainer/  
[Accessed 30 04 2018].

http://www.dvwa.co.uk/, 2018. *DVWA - Damn Vulnerable Web Application.* [Online].

https://sqlzoo.net/hack/, 2018. *SQL Injection Attack.* [Online].

https://www.uwamp.com/en/, 2018. *UwAmp Wamp Server - Apache MySQL PHP.* [Online].

https://www.youtube.com/watch?v=ciNHn38EyRc&t=615s, 2018. *Running an SQL Injection Attack - Computerphile - YouTube.* [Online].

owasp.com, 2018. *OWASP Top Ten.* [Online]   
Available at: https://www.owasp.org/images/7/72/OWASP\_Top\_10-2017\_%28en%29.pdf.pdf  
[Accessed 30 04 2018].

theguardian.com, 2018. *TalkTalk hit with record £400k fine over cyber-attack | Business | The Guardian.* [Online]   
Available at: https://www.theguardian.com/business/2016/oct/05/talktalk-hit-with-record-400k-fine-over-cyber-attack