Vulnerability Scanner

G1 - b
Under Supervision of:
Dr. A.L. Sangal

SIDHARTH SINGH (14103014)
SRIJANI SEN (14103075)

Web Vulnerability

- Vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application.
- Stakeholders include the application owner, application users, and other entities that rely on the application

Why are web applications vulnerable?

- Websites and web applications are easily available via the internet 24 hours a day, 7 days a week to customers, employees, suppliers and therefore also hackers.
- Web applications often have direct access to backend data such as customer databases.
- Firewalls provide no protection against web application hacking, simply because access to the website has to be made public.

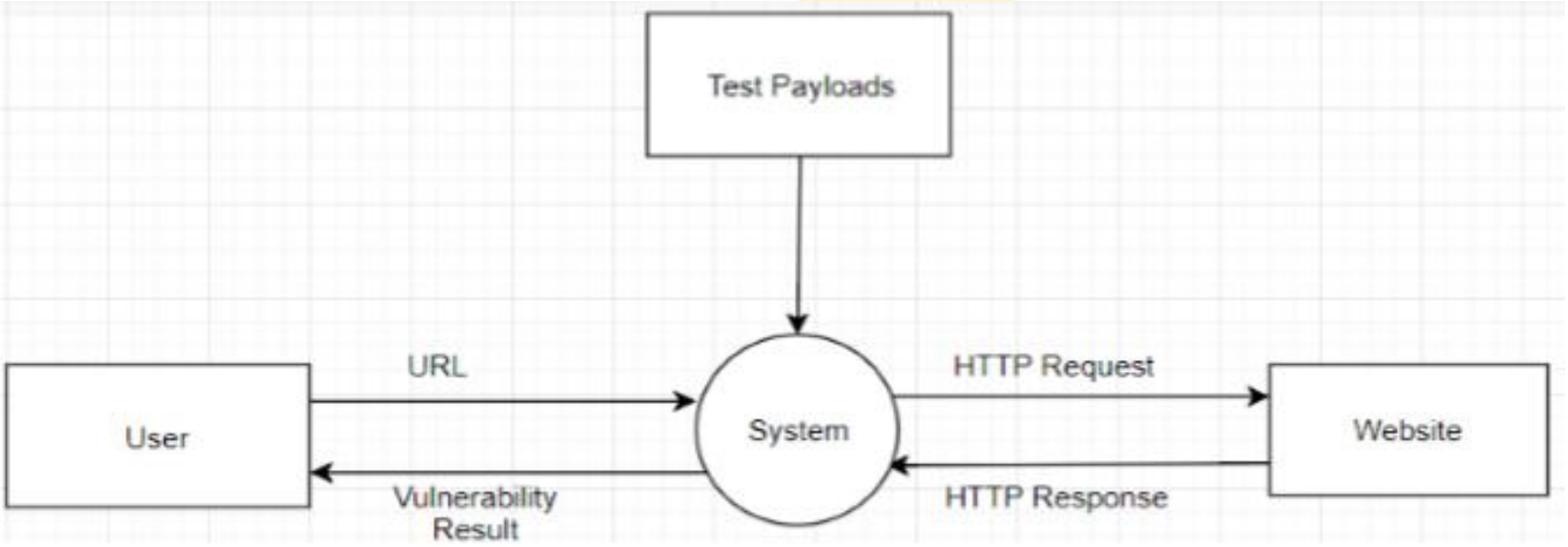
Vulnerability Scanners

- Vulnerability scanners are automated tools that scan hosts and networks for known vulnerabilities and weaknesses
- It can assess a variety of vulnerabilities across information systems that may have originated from a vendor, system administration activities, or general day-to-day user activities
- Administrators need to conduct a scan and fix problems before an Attacker
- Both administrators and attackers can use the same tool for fixing or exploiting a system

Need of Vulnerability Scanner

- Very good at checking for hundreds (or thousands) of potential problems quickly
  - Automated
  - Regularly
- Manual vulnerability auditing of all web applications is complex and time-consuming, since it generally involves processing a large volume of data
- Hackers are constantly finding new ways to exploit web application, which means that we need to constantly monitor the security communities, and find new vulnerabilities in web application code before hackers discover them.
- Early Detection and Defense in depth
- Within minutes, an automated web application scanner can scan web application and find vulnerabilities easily.

Architecture:



- The client will provide url of the website to be tested and what kind of test to be performed on it.
- The system will build the crafted request with the help of test criteria.
- The build crafted request is send to the server and scanner analyzes the response return by the web server and report will be send to client.

Objective

Development of Scanner for detecting vulnerabilities to protect the website from web-based attacks

1. To find out whether a host is alive or not.

2. To find out all the services running on that host.

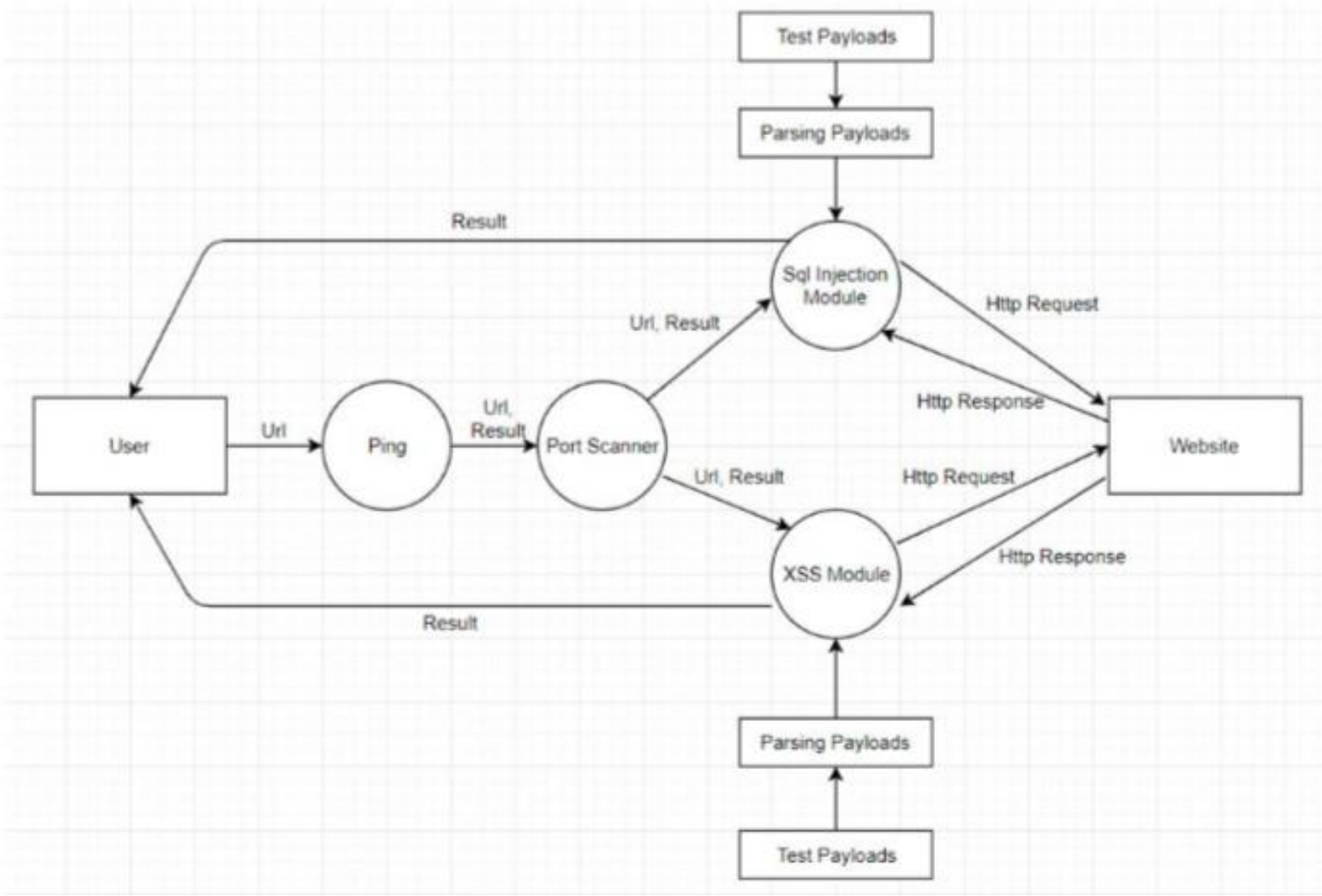
3. Finds out whether a particular host is running a web server or not.

4. Performs SQL Injection Vulnerability Testing

5. Performs XSS Vulnerability Testing

6. Low False Positive Rates: Detection of inexistent vulnerabilities are a nightmare to deal with. False positives reduce confidence in automated security testing and waste the developers' time trying to find and fix vulnerabilities.

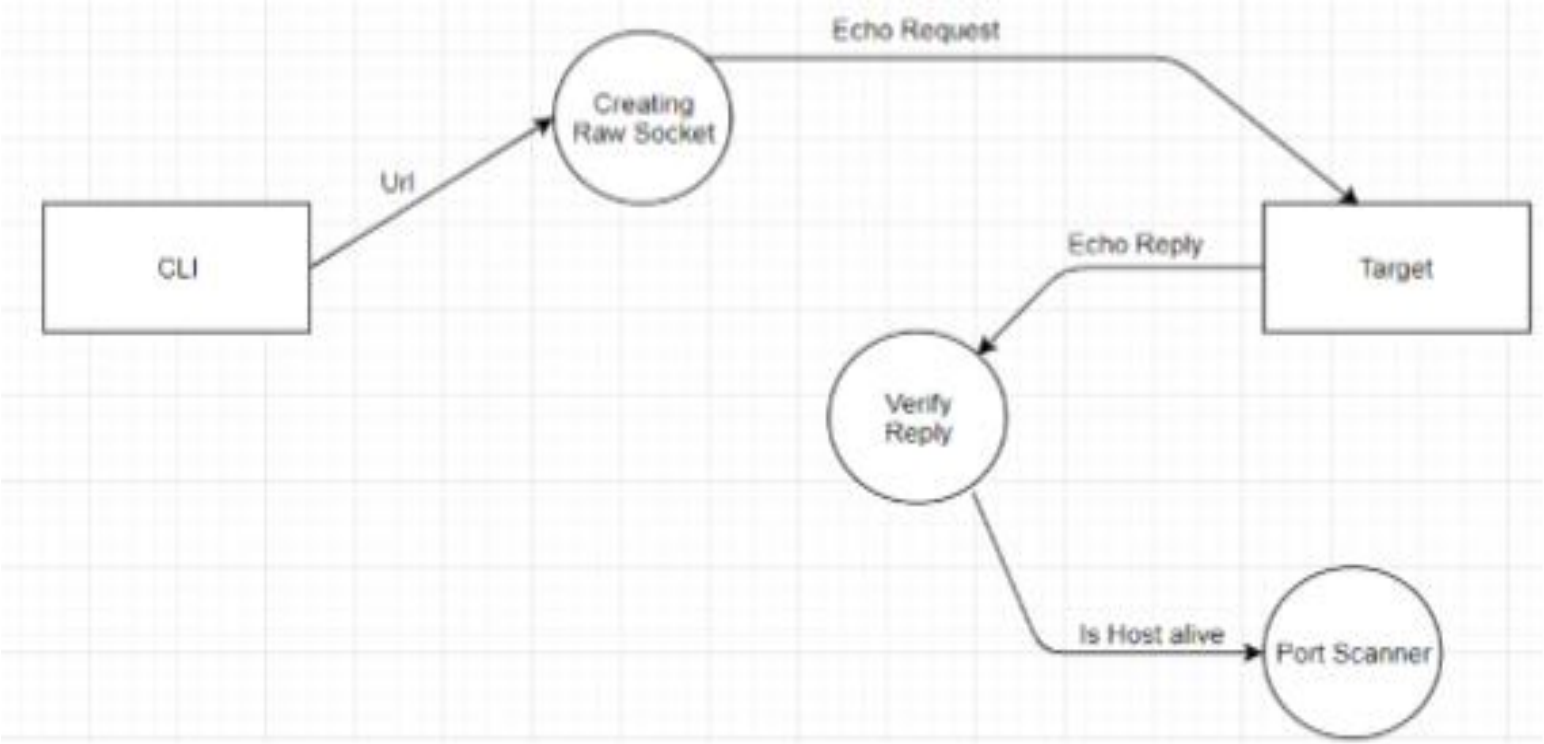
Data flow diagram :



Checking if the Host is alive

- Send ICMP ECHO_REQUEST packet to the network host
- To determine whether host(s) is up and running
- It is implemented using Raw Sockets
- Checked reply on the socket. If the id of the reply matches the packet id , we mark it as reachable.
- If a target doesn't reply within a certain time limit it is marked unreachable

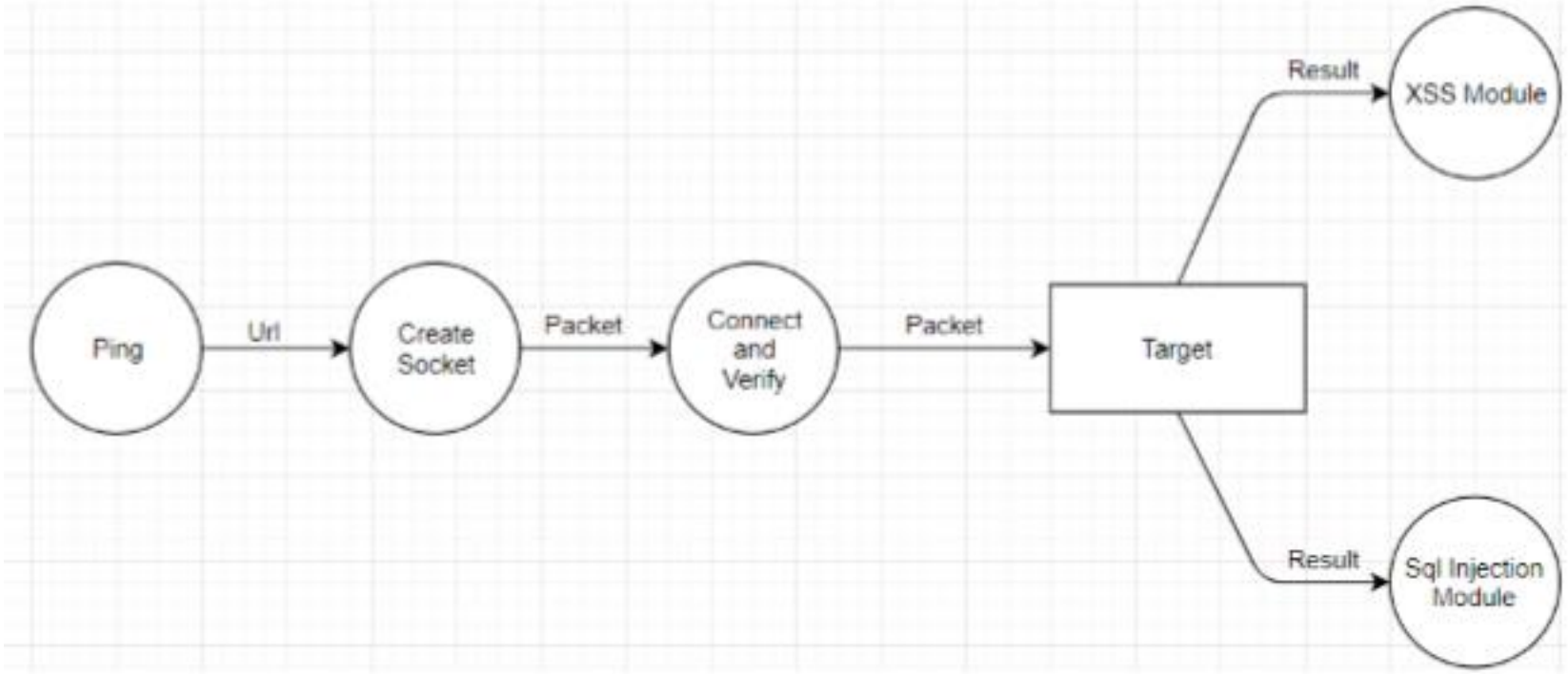
Checking if the Host is alive



Port Scanner

- Scans the host for open ports and services visible to the attacker
- Creates a socket and connects to every port on the target address
- It then finds which service is open.

Port Scanner



What is Cross Site Scripting?

XSS vulnerabilities target scripts embedded in a page that are executed on the client side

Attackers can use XSS to execute malicious scripts on the users

Since the browser cannot know if the script is trusty or not, the script will be executed, and the attacker can hijack session cookies, deface websites, or redirect the user to an unwanted and malicious websites.

Vulnerable Objects

- Input Fields
- URLs

What is Cross Site Scripting?

The following server-side pseudo-code is used to display the most recent comment on a web page.

```
print "<html>"

print "<h1>Most recent comment</h1>"

print database.latestComment

print "</html>"
```

The above script is simply printing out the latest comment from a comments database and printing the contents out to an HTML page, assuming that the comment printed out only consists of text.

What is Cross Site Scripting?

The above page is vulnerable to XSS because an attacker could submit a comment that contains a malicious payload such as <script>doSomethingEvil();</script>.

Users visiting the web page will get served the following HTML page.

<html>

<h1>Most recent comment</h1>

<script>doSomethingEvil();</script>

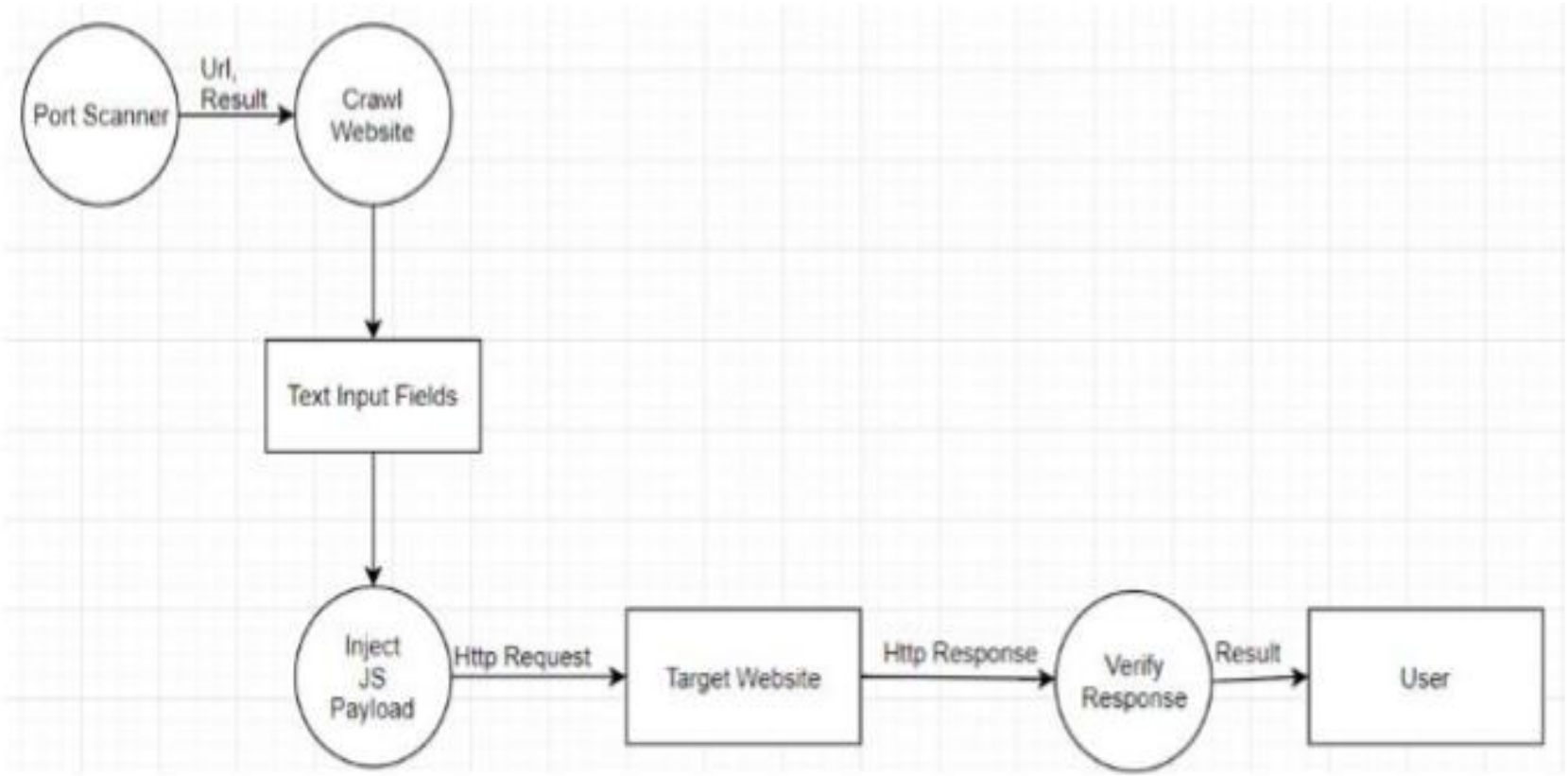
</html>

When the page loads in the victim's browser, the attacker's malicious script will execute, most often without the user realizing or being able to prevent such an attack.

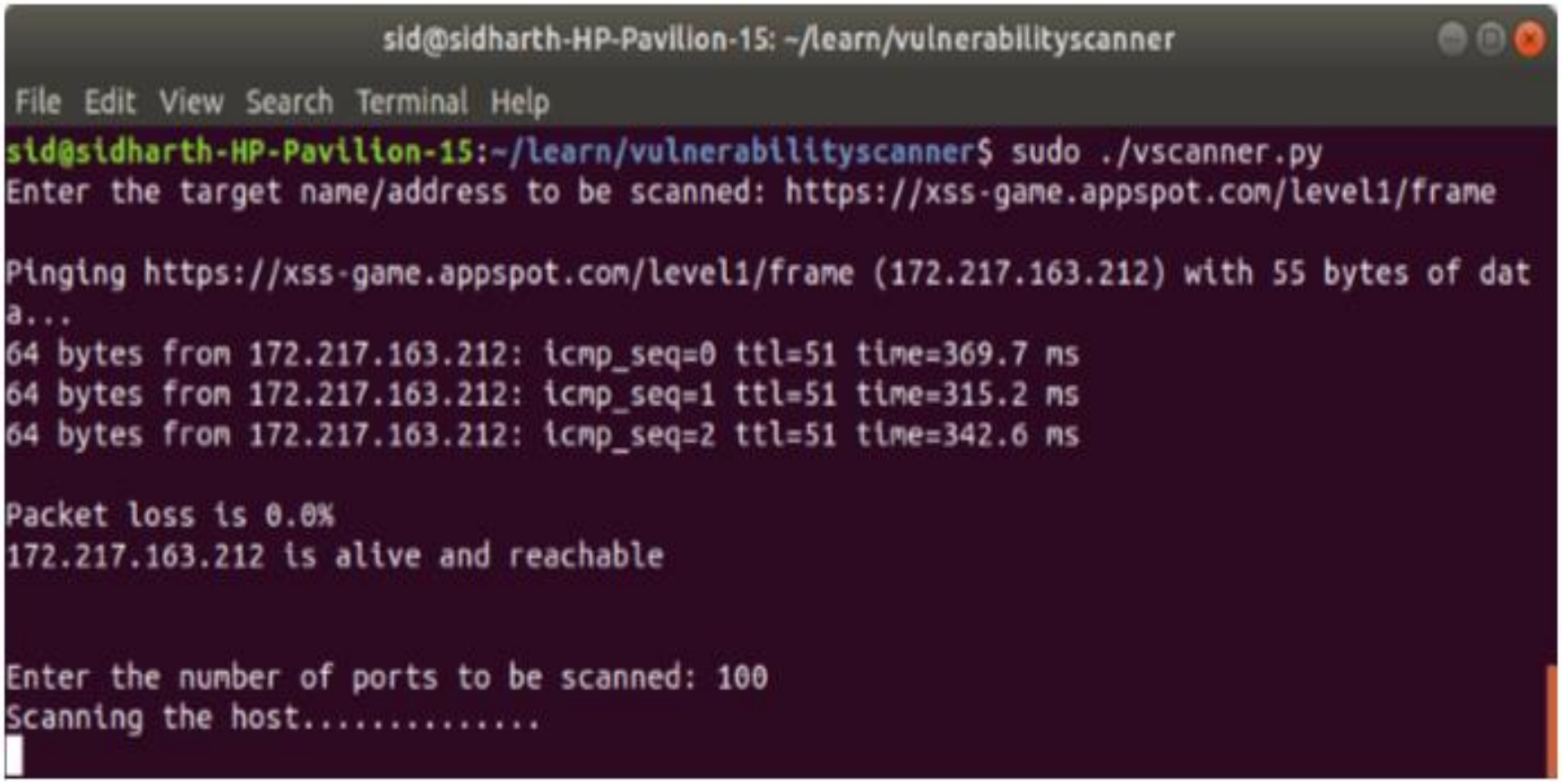
Cross Site Scripting Scanner

- Automatic Insertion of “<script>alert('Test')</script>", "<script>alert('Hello')</script>" in every textbox available in the web form.
- After preparing the test cases, cases are passed to every links and web forms of inputted website
- With these test cases, the test will be performed on the website.
- Scanner will build the HTTP request to send it to the respective URL and in response URL will give HTTP response.
- The response will contain all the details and the test will be performed on these details. If the same inputted content <script>alert('Test')</script>", "<script>alert('Hello')</script>" is found in response text then that particular webform or link is vulnerable to cross site scripting

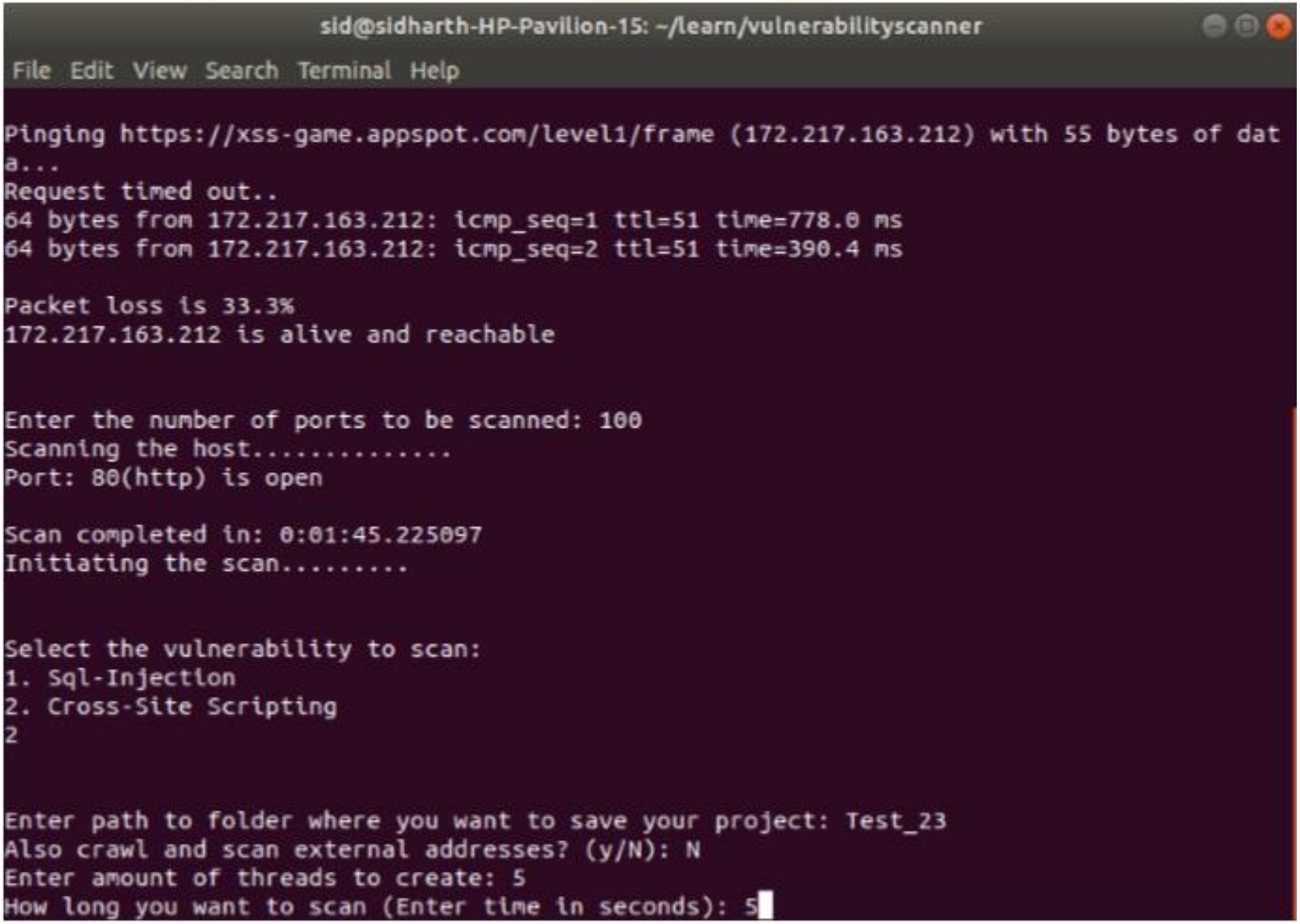
Cross Site Scripting Scanner



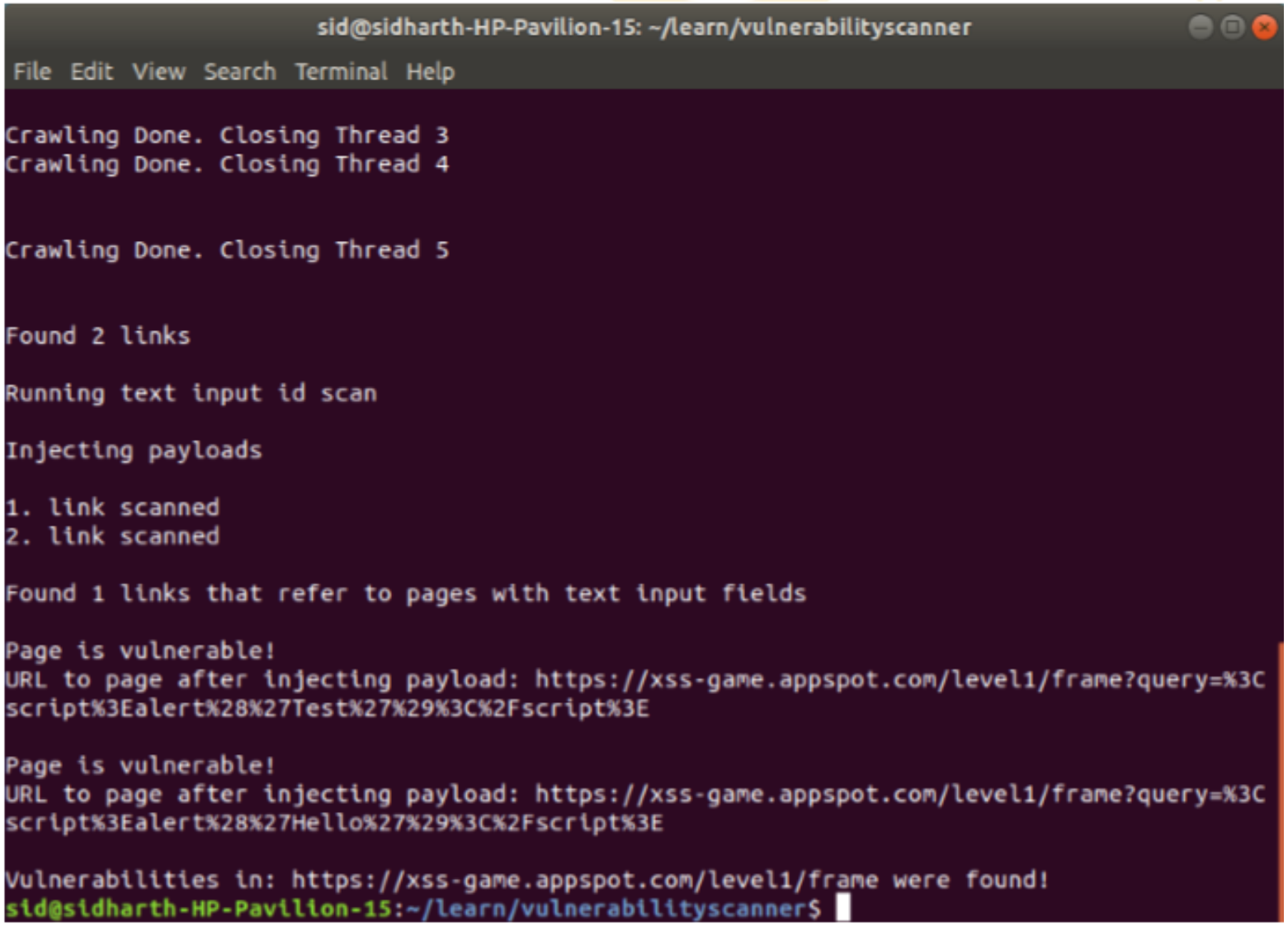
Screenshots:



Screenshots:



Screenshots:



What is SQL Injection?

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database

An attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database.

SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity.

An SQL Injection needs just two conditions to exist – a relational database that uses SQL, and a user controllable input which is directly used in an SQL query.

SQL Injection Example

Here is an example of a user login on a web site:

Username:

John Doe

Password:

myPass

example:

```
uName = getRequestString("username");
uPass = getRequestString("userpassword");

sql = 'SELECT * FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'
```

result:

```
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"
```

SQL Injection Example

A hacker might get access to user names and passwords in a database by simply inserting " OR ""=" into the user name or password text box:

User Name:

" or ""="

Password:

" or ""="

The code at the server will create a valid SQL statement like this:

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

The SQL above is valid and will return all rows from the "Users" table, since OR ""="" is always TRUE.

SQL Injection Scanner

- To check for SQL injection the scanner crawls the website for input forms and fields. It then generates various payloads like:

- Automatic Insertion of 1’ in every textbox available in the web form.
- Modifying the URL by replacing = with =’

- After preparing the test payloads, these are passed to every links and web forms of inputted website.

- With these test cases, the test will be performed on the website. Scanner will build the HTTP request to send it to the respective URL and in response URL will give HTTP response.

- The response will contain all the details and the test will be performed on these details.

SQL Injection Scanner

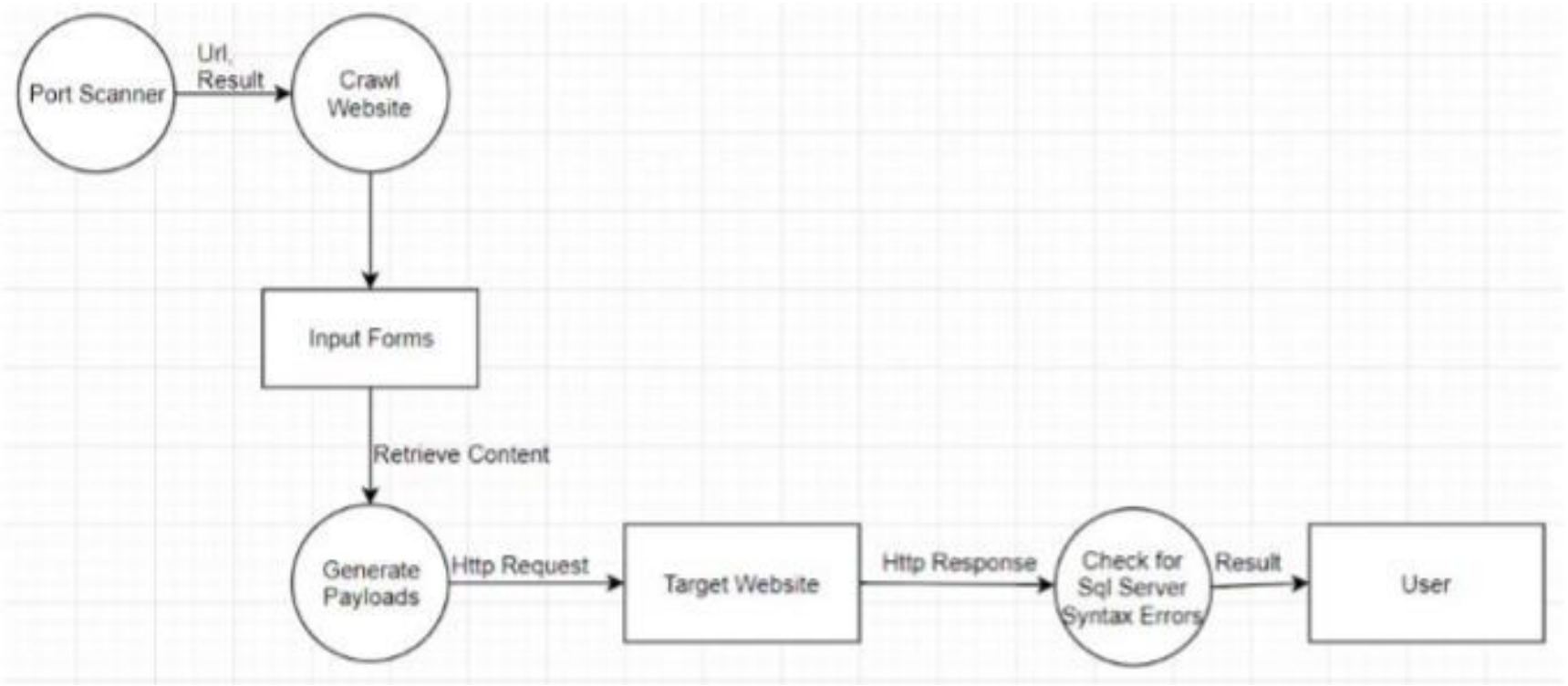
- If any kind of syntax error is found as per a database server in response text or any internal error in a webform is found then that particular webform or link is vulnerable to SQL Injection.
- To check for the syntax error the scanner has all the patterns of syntax error.
- If any of the pattern found in response text is matched with the patterns in the scanner then the particular webform is vulnerable to SQL Injection

List of patterns:

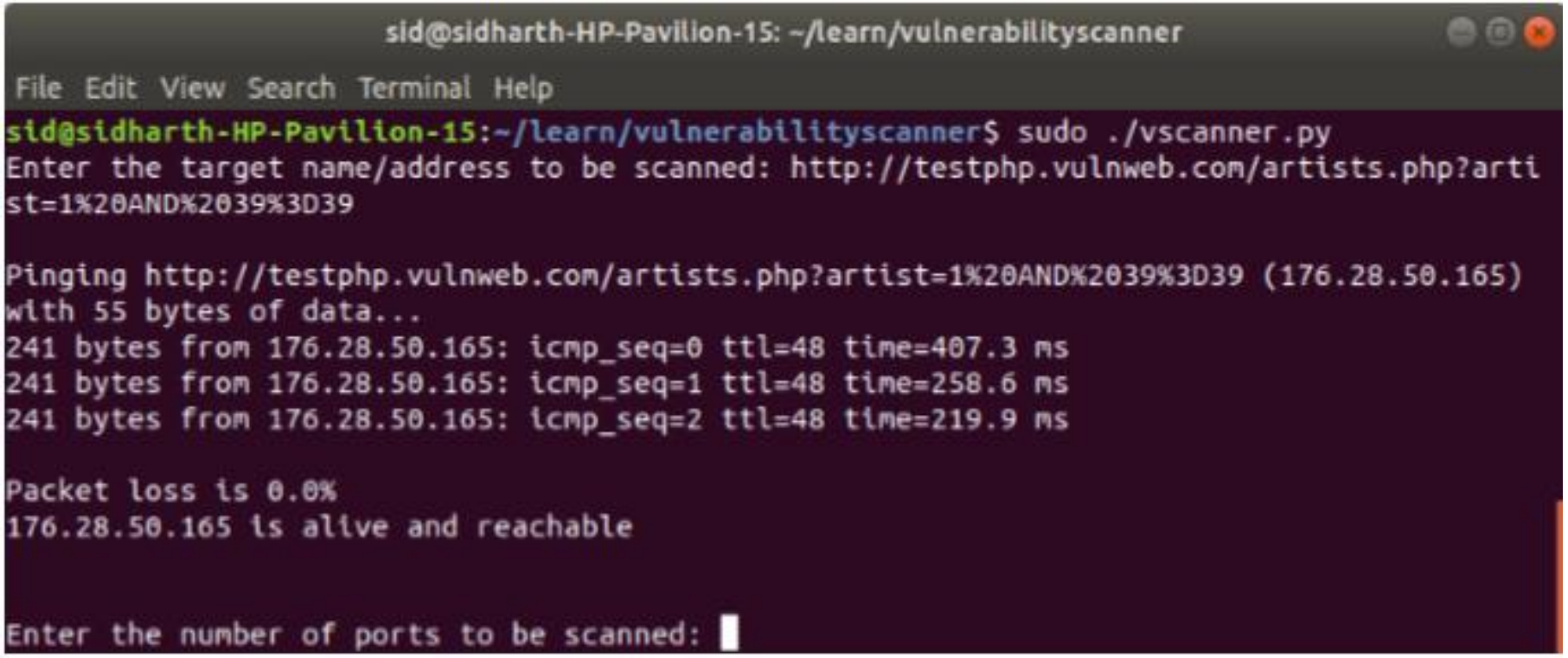
Few examples :

- "MySQL": (r"SQL syntax.*MySQL", r"Warning.*mysql_.*", r"valid MySQL result", r"MySqlClient\.")
- "PostgreSQL": (r"PostgreSQL.*ERROR", r"Warning.*\Wpg_.*", r"valid PostgreSQL result", r"Npgsql\.")
- "Microsoft SQL Server": (r"Driver.* SQL[\-_\ ]*Server", r"OLE DB.* SQL Server", r"(\W|\A)SQL Server.*Driver", r"Warning.*mssql_.*", r"(\W|\A)SQL Server.*[0-9a-fA-F]{8}", r"(?s)Exception.*\WSystem\.Data\.SqlClient\.", r"(?s)Exception.*\WRoadhouse\.Cms\.")

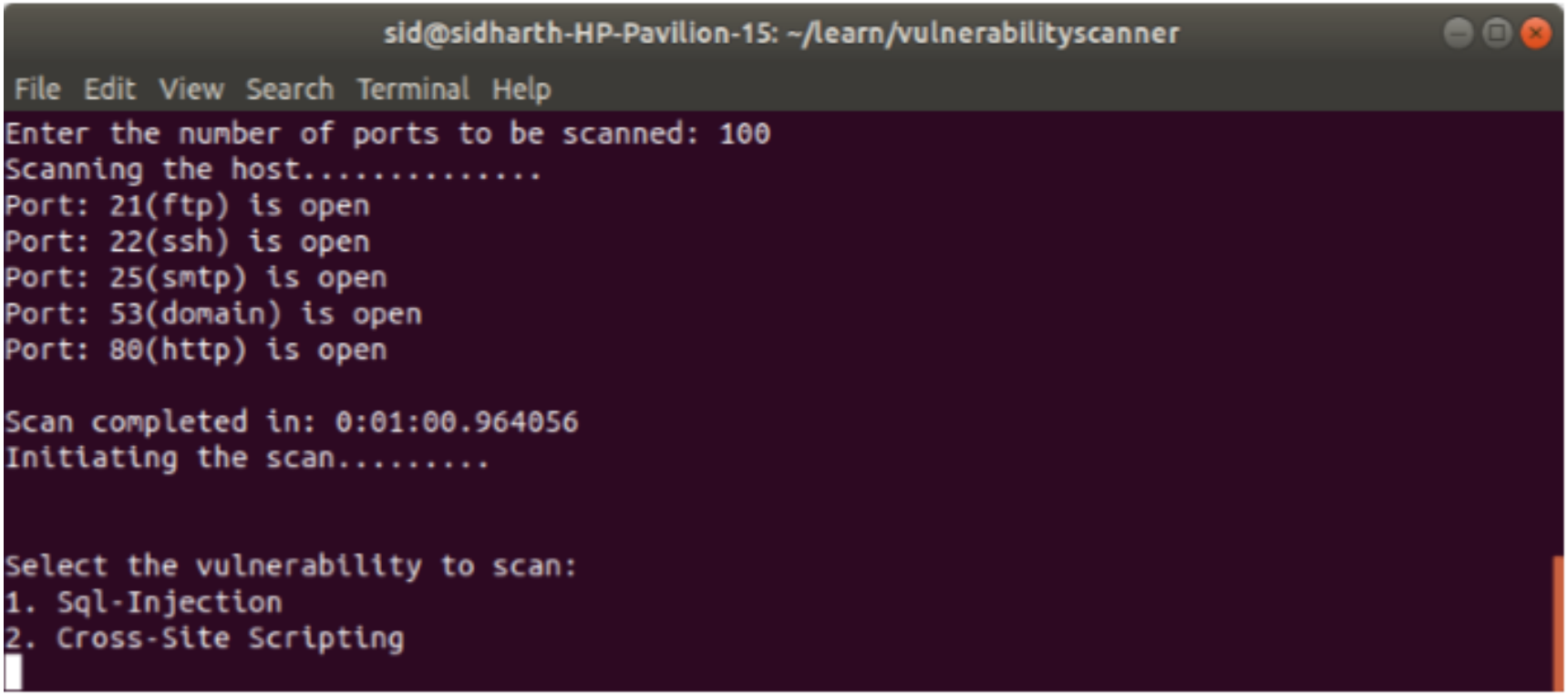
SQLi Module:


Port Scanner
Url, Result
Crawl Website
Input Forms
Retrieve Content
Generate Payloads
Http Request
Target Website
Http Response
Check for Sql Server Syntax Errors
Result
User

Screenshots:



Screenshots:



Screenshots:

```
1


* scanning GET parameter 'artist'
 (i) GET parameter 'artist' appears to be error SQLi vulnerable (MySQL)
 (i) GET parameter 'artist' appears to be blind SQLi vulnerable (e.g.: 'http://testphp.vulnweb.com/artists.ph
p?artist=1%20AND%2088%3D88')

scan results: possible sqli vulnerabilities found
sid@sidharth-HP-Pavilion-15:~/learn/vulnerabilityscanner$
```

Conclusion and Future Scope

- We described how the website is vulnerable to SQL Injection and XSS based on automatically generation of specially crafted request allowing the successful exploitation of detected vulnerabilities.
- The performance of the system is found to be satisfactory.
- Rich area for future work is exploration of many more web vulnerabilities.
- We can detect other network vulnerabilities like DOS.

Thank You