Introduction

Penetration testing or Pen Testing is a security exercise practiced by Software Engineer or Cyber Security experts - attempts to find and exploit vulnerabilities in a system. As pen testing broadly consists of 5 different phases which are coined as: -

- Planning & Reconnaissance
- Scanning & Discovery
- Exploitation
- Risk Analysis & Suggestions
- Report Generation

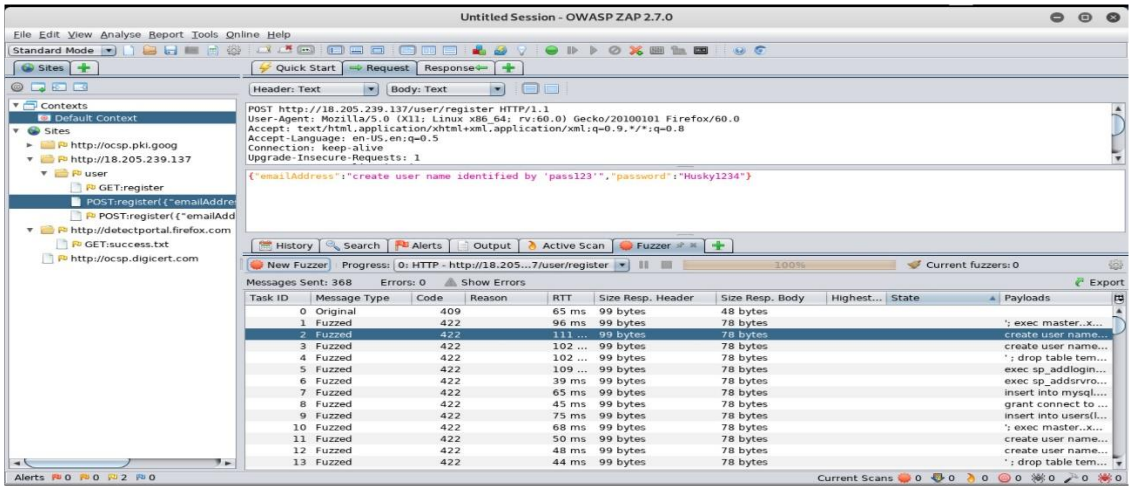
Penetration testing comprises of 5 types and could be done in done in different ways to exploit the vulnerabilities and to check out the level of damage it could cause to the system.

Web Application Penetration Testing checks the vulnerabilities associated with the web-based application which involved Java Applets, API, SQL Injection, etc. We have a web application which can register the user. The primary purpose of the application is for users to keep a NOTE by also giving an option to upload the file associated with the NOTE. A registered user can perform CRUD operations on a NOTE that he created as well as attach files to it. There is also an option to reset the user password and for a registered user to fetch the current time by providing authentication in the request header.

Attack Vectors

Below are Attack Vectors used to perform penetration testing on the application.

1. **SQL Injection Attack**: - It is a technique where an attacker executes malicious SQL queries that control a web application's database. With the right set of queries, a user can gain access to information stored in databases. We have used OWASP Tool to tests whether a HTTP request parameter is vulnerable to SQL Injection.



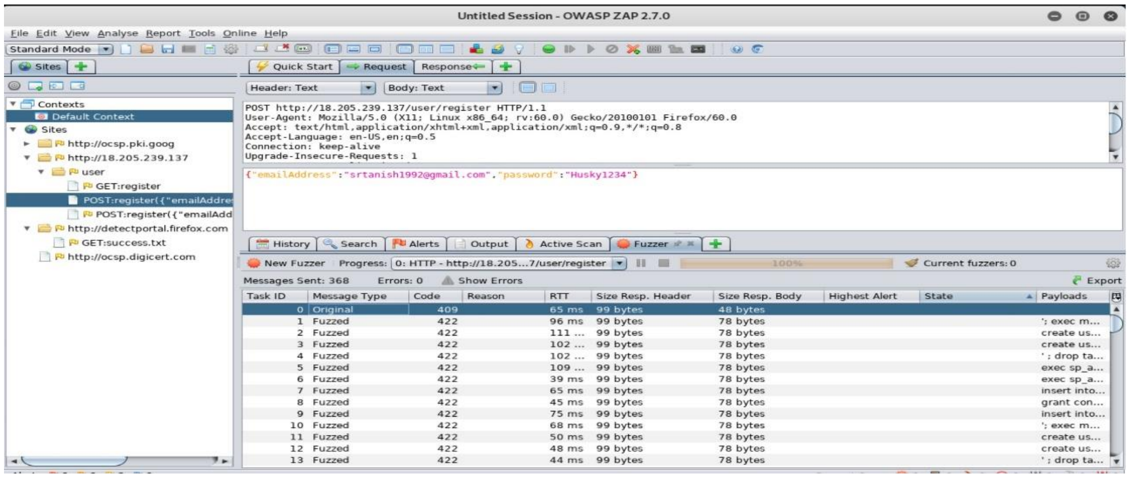
```
POST http://18.205.239.137/user/register HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
{"emailAddress":"create user name identified by 'pass123'","password":"Husky1234"}
```

Progress: 0: HTTP - http://18.205...7/user/register 100%

Messages Sent: 368 Errors: 0 Show Errors

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest...	State	Payloads
0	Original	409		65 ms	99 bytes	48 bytes			
1	Fuzzed	422		96 ms	99 bytes	78 bytes			'; exec master..x...
2	Fuzzed	422		111 ...	99 bytes	78 bytes			create user name...
3	Fuzzed	422		102 ...	99 bytes	78 bytes			create user name...
4	Fuzzed	422		102 ...	99 bytes	78 bytes			' ; drop table tem...
5	Fuzzed	422		109 ...	99 bytes	78 bytes			exec sp_addlogin...
6	Fuzzed	422		39 ms	99 bytes	78 bytes			exec sp_addsrvro...
7	Fuzzed	422		65 ms	99 bytes	78 bytes			insert into mysql....
8	Fuzzed	422		45 ms	99 bytes	78 bytes			grant connect to ...
9	Fuzzed	422		75 ms	99 bytes	78 bytes			insert into users(l...
10	Fuzzed	422		68 ms	99 bytes	78 bytes			'; exec master..x...
11	Fuzzed	422		50 ms	99 bytes	78 bytes			create user name...
12	Fuzzed	422		48 ms	99 bytes	78 bytes			create user name...
13	Fuzzed	422		44 ms	99 bytes	78 bytes			' ; drop table tem...



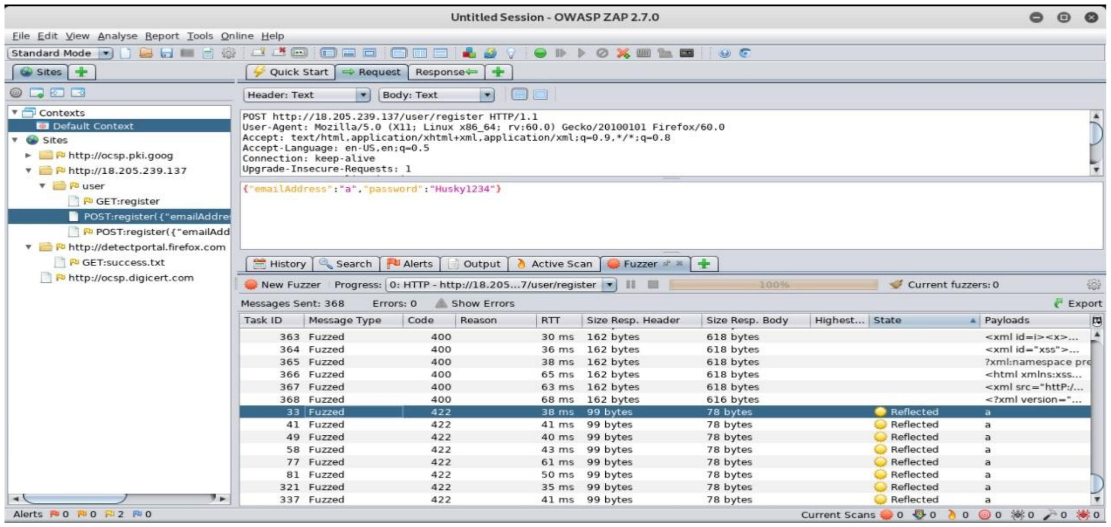
```
POST http://18.205.239.137/user/register HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
{"emailAddress":"srtanish1992@gmail.com","password":"Husky1234"}
```

Progress: 0: HTTP - http://18.205...7/user/register — Current fuzzers: 0

Messages Sent: 368 Errors: 0

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	409		65 ms	99 bytes	48 bytes			
1	Fuzzed	422		96 ms	99 bytes	78 bytes			'; exec m...
2	Fuzzed	422		111 ...	99 bytes	78 bytes			create us...
3	Fuzzed	422		102 ...	99 bytes	78 bytes			create us...
4	Fuzzed	422		102 ...	99 bytes	78 bytes			' ; drop ta...
5	Fuzzed	422		109 ...	99 bytes	78 bytes			exec sp_a...
6	Fuzzed	422		39 ms	99 bytes	78 bytes			exec sp_a...
7	Fuzzed	422		65 ms	99 bytes	78 bytes			insert into...
8	Fuzzed	422		45 ms	99 bytes	78 bytes			grant con...
9	Fuzzed	422		75 ms	99 bytes	78 bytes			insert into...
10	Fuzzed	422		68 ms	99 bytes	78 bytes			'; exec m...
11	Fuzzed	422		50 ms	99 bytes	78 bytes			create us...
12	Fuzzed	422		48 ms	99 bytes	78 bytes			create us...
13	Fuzzed	422		44 ms	99 bytes	78 bytes			' ; drop ta...



```
POST http://18.205.239.137/user/register HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
{"emailAddress":"a","password":"Husky1234"}
```

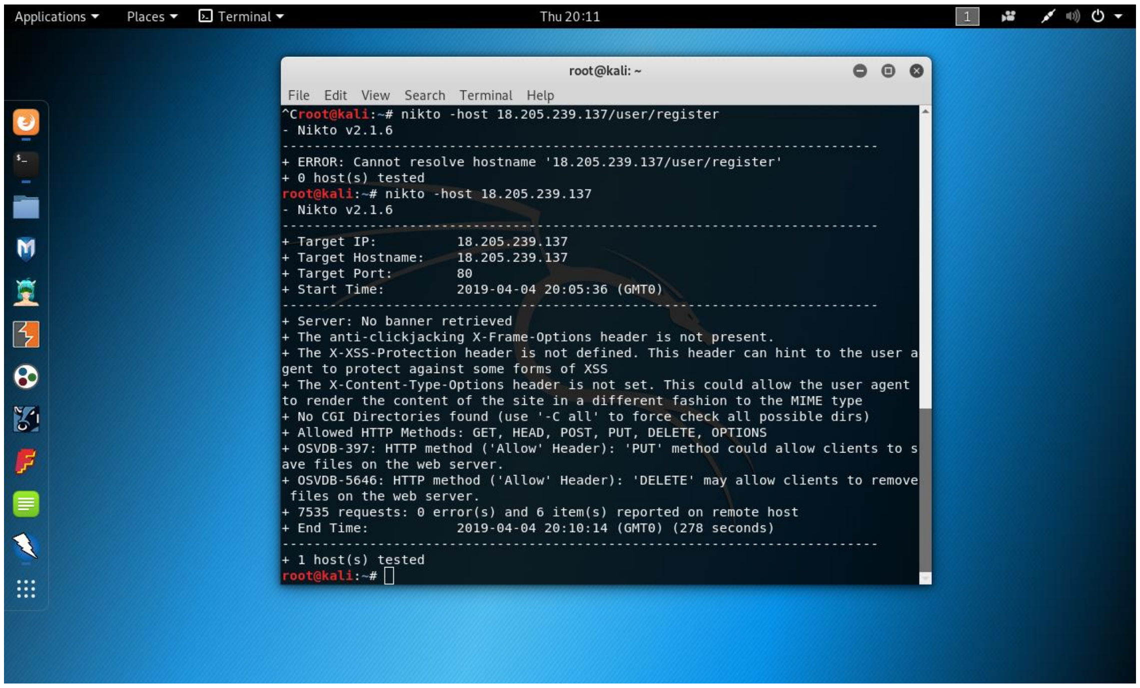
Progress: 0: HTTP - http://18.205...7/user/register — Current fuzzers: 0

Messages Sent: 368 Errors: 0

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest...	State	Payloads
363	Fuzzed	400		30 ms	162 bytes	618 bytes			<xml id=i><x>...
364	Fuzzed	400		36 ms	162 bytes	618 bytes			<xml id="xss">...
365	Fuzzed	400		38 ms	162 bytes	618 bytes			?xml:namespace pre
366	Fuzzed	400		65 ms	162 bytes	618 bytes			<html xmlns:xss...
367	Fuzzed	400		63 ms	162 bytes	618 bytes			<xml src="httP:/...
368	Fuzzed	400		68 ms	162 bytes	616 bytes			<?xml version="...
33	Fuzzed	422		38 ms	99 bytes	78 bytes		Reflected	a
41	Fuzzed	422		41 ms	99 bytes	78 bytes		Reflected	a
49	Fuzzed	422		40 ms	99 bytes	78 bytes		Reflected	a
58	Fuzzed	422		43 ms	99 bytes	78 bytes		Reflected	a
77	Fuzzed	422		61 ms	99 bytes	78 bytes		Reflected	a
81	Fuzzed	422		50 ms	99 bytes	78 bytes		Reflected	a
321	Fuzzed	422		35 ms	99 bytes	78 bytes		Reflected	a
337	Fuzzed	422		41 ms	99 bytes	78 bytes		Reflected	a

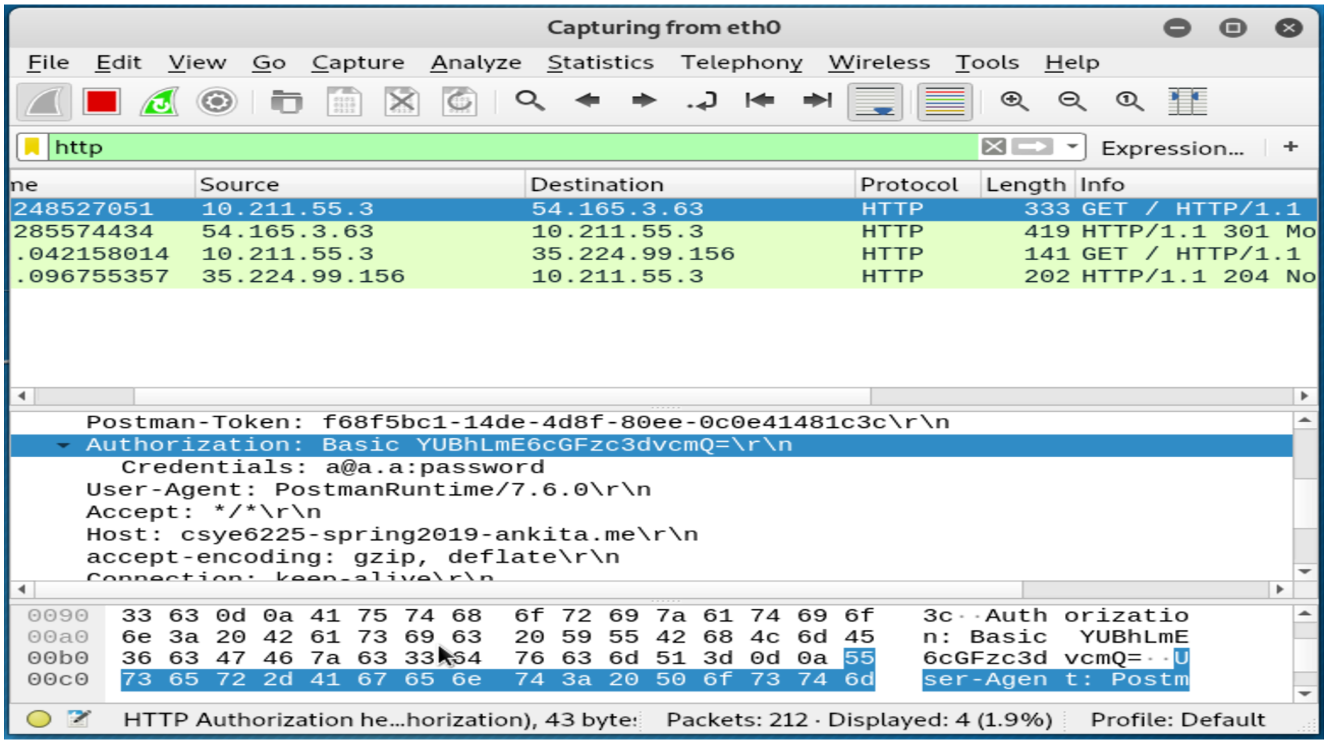
We've used the Fuzzing for SQL Injection Flaws with OWASP ZAP. We hit the one end with POST request - /user/register. The POST request had been executed, and now attacked by dozens of SQL requests that may be potentially dangerous for your application. Requests with the 'Reflected' status in the 'State' column are safe for the application – the rest, however, may be not.

2. **<u>Unprotected APIs: -</u>** A part of web-application pen testing - checks the potential vulnerabilities in the APIs. Nikto Scanner checks the malicious end points and give a short report about it. We could also save the result in a file so that we could refer in the future.



```
^Croot@kali:~# nikto -host 18.205.239.137/user/register
- Nikto v2.1.6
---------------------------------------------------------------------------
+ ERROR: Cannot resolve hostname '18.205.239.137/user/register'
+ 0 host(s) tested
root@kali:~# nikto -host 18.205.239.137
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          18.205.239.137
+ Target Hostname:    18.205.239.137
+ Target Port:        80
+ Start Time:         2019-04-04 20:05:36 (GMT0)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user a
gent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to s
ave files on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove
 files on the web server.
+ 7535 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2019-04-04 20:10:14 (GMT0) (278 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~# 
```

3. **<u>Sensitive Data Exposure: -</u>** Sensitive Data Exposure occurs when a web application does not adequately protective sensitive information. The data can vary anything from passwords, session tokens, credit card data to private health data. We have used the Wireshark application used to expose or intercept Web Application modules like Basic Authentication in the request and fetch the username and password of the user. The image below describes this situation.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...

```
ne               Source            Destination       Protocol  Length Info
248527051        10.211.55.3       54.165.3.63       HTTP         333 GET / HTTP/1.1
285574434        54.165.3.63       10.211.55.3       HTTP         419 HTTP/1.1 301 Mo
.042158014       10.211.55.3       35.224.99.156     HTTP         141 GET / HTTP/1.1
.096755357       35.224.99.156     10.211.55.3       HTTP         202 HTTP/1.1 204 No
```

```
   Postman-Token: f68f5bc1-14de-4d8f-80ee-0c0e41481c3c\r\n
 ▾ Authorization: Basic YUBhLmE6cGFzc3dvcmQ=\r\n
      Credentials: a@a.a:password
   User-Agent: PostmanRuntime/7.6.0\r\n
   Accept: */*\r\n
   Host: csye6225-spring2019-ankita.me\r\n
   accept-encoding: gzip, deflate\r\n
   Connection: keep-alive\r\n
```

```
0090  33 63 0d 0a 41 75 74 68  6f 72 69 7a 61 74 69 6f   3c··Auth orizatio
00a0  6e 3a 20 42 61 73 69 63  20 59 55 42 68 4c 6d 45   n: Basic  YUBhLmE
00b0  36 63 47 46 7a 63 33 64  76 63 6d 51 3d 0d 0a 55   6cGFzc3d vcmQ=··U
00c0  73 65 72 2d 41 67 65 6e  74 3a 20 50 6f 73 74 6d   ser-Agen t: Postm
```

HTTP Authorization he...horization), 43 byte: Packets: 212 · Displayed: 4 (1.9%) Profile: Default