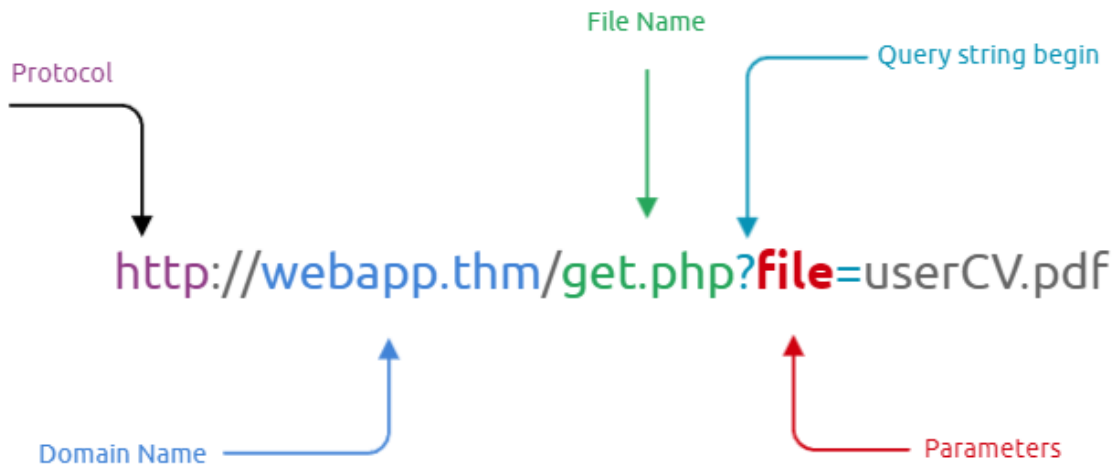
TryHackMe - File Inclusion

This room introduces file inclusion vulnerabilities, including Local File Inclusion (LFI), Remote File Inclusion (RFI), and directory traversal.

Task 1: Introduction

What is File inclusion?

The following diagram breaks down the essential parts of a URL.



Why do File inclusion vulnerabilities happen?

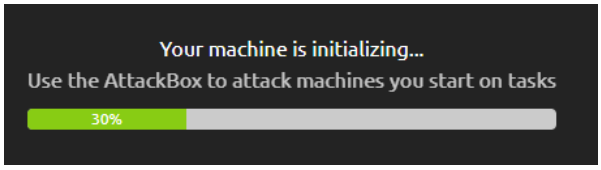
File inclusion vulnerabilities are commonly found and exploited in various programming languages for web applications, such as PHP that are poorly written and implemented. The main issue of these vulnerabilities is the input validation, in which the user inputs are not sanitized or validated, and the 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?

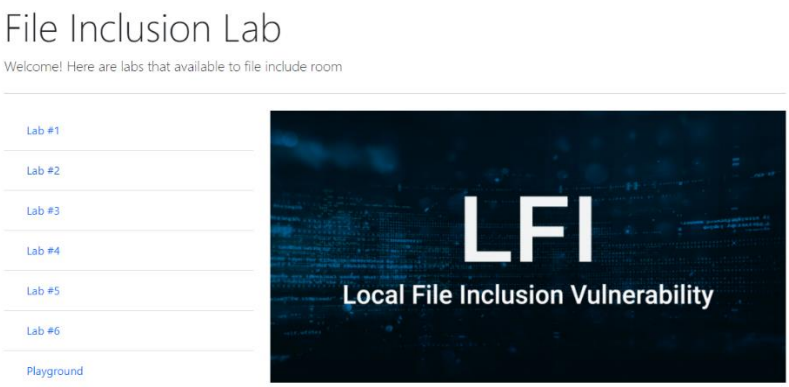
By default, an attacker can leverage file inclusion vulnerabilities to leak data, such as code, credentials or other important files related to the web application or operating system. Moreover, if the attacker can write files to the server by any other means, file inclusion might be used in tandem to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

Task 2: Deploy the VM

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



Please visit the link http://MACHINE_IP/ which should look as follows,



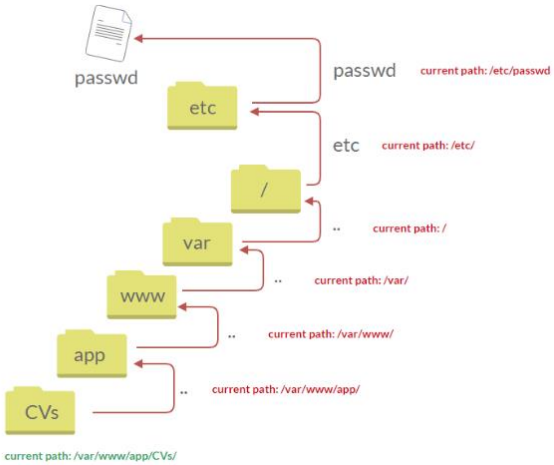
Task 3: Path Traversal

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.

Example of how directory traversal looks like:



Question 1: What function causes path traversal vulnerabilities in PHP?

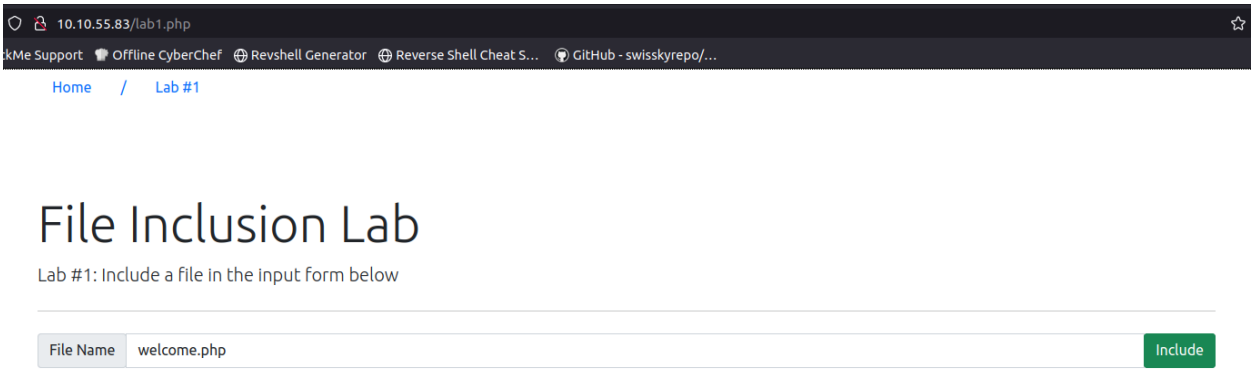
Answer: *file_get_contents*

Task 4: Local File Inclusion - LFI

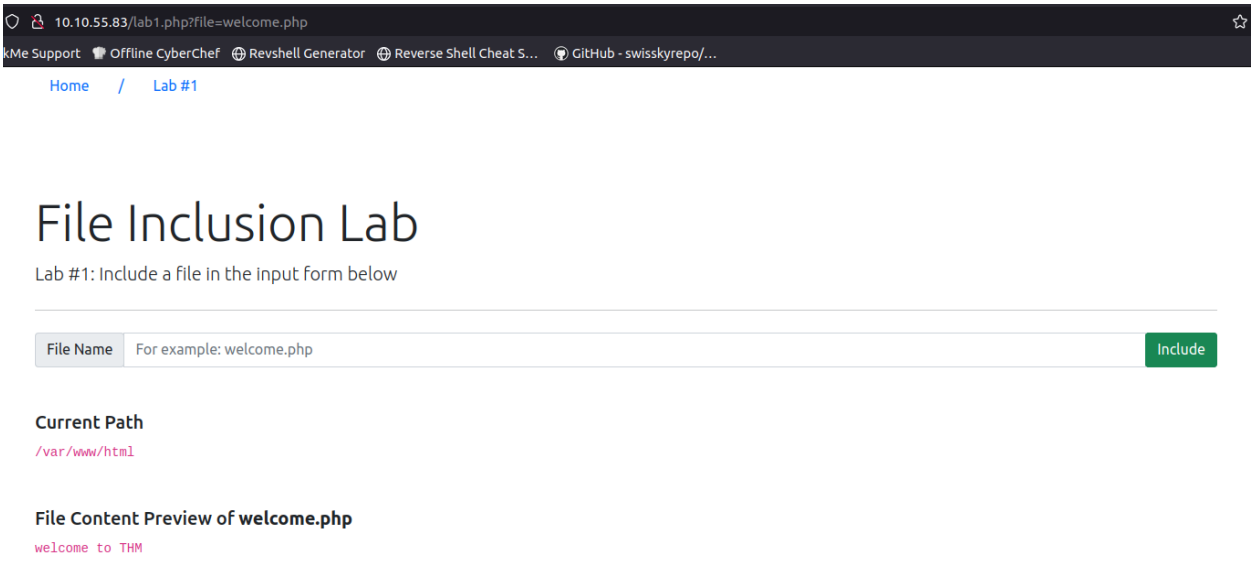
LFI attacks against web applications are often due to a developers' lack of security awareness. With PHP, using functions such as include, require, include_once, and require_once often contribute to vulnerable web applications. In this room, we'll be picking on PHP, but it's worth noting LFI vulnerabilities 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.

Lab #1

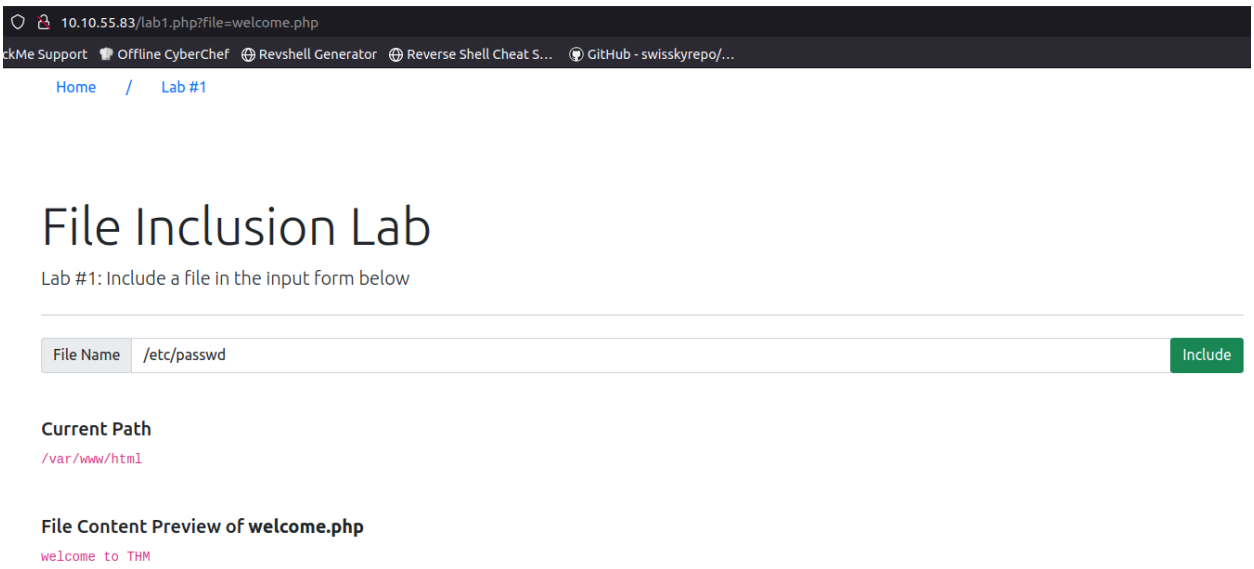
Type “welcome.php”.



The following image is displayed:



We need to go to **/etc/passwd**



This is displayed:

File Content Preview of /etc/passwd

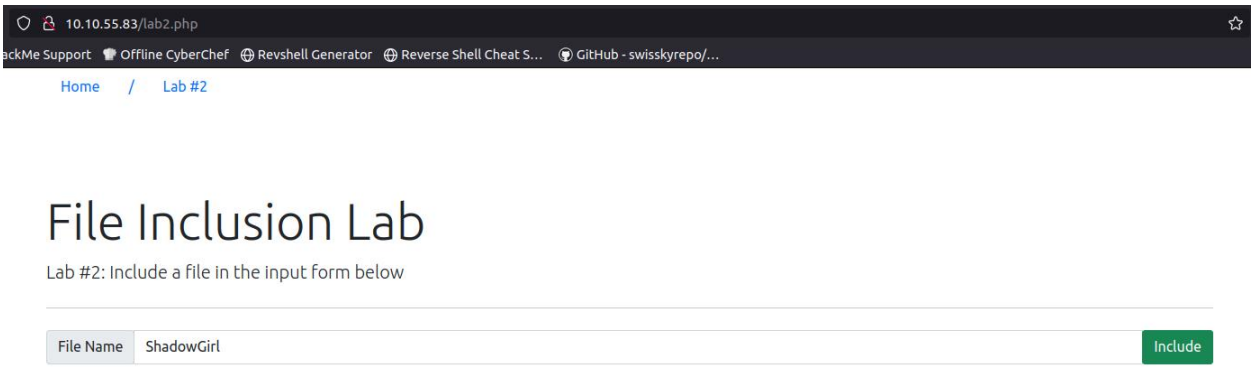
```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:
/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
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
```

Question 2: Give Lab #1 a try to read /etc/passwd. What would the request URI be?

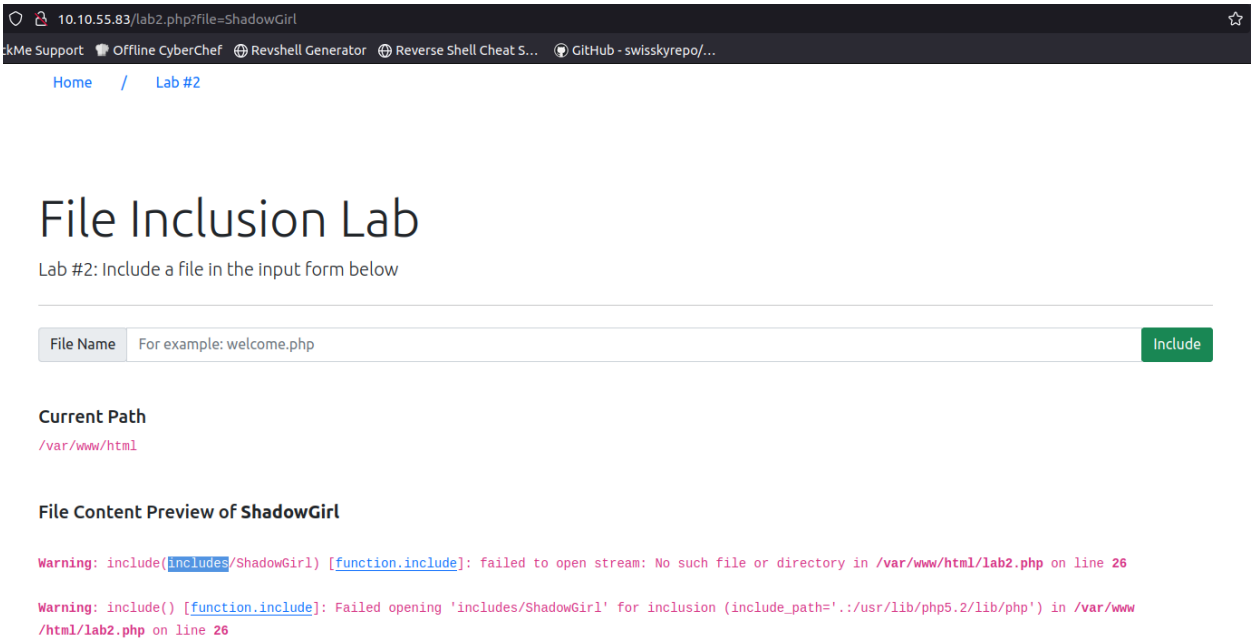
Answer: *./lab1.php?file=/etc/passwd*

Lab #2

Type “ShadowGirl”.



The following image is displayed:



Question 3: In Lab #2, what is the directory specified in the include function?

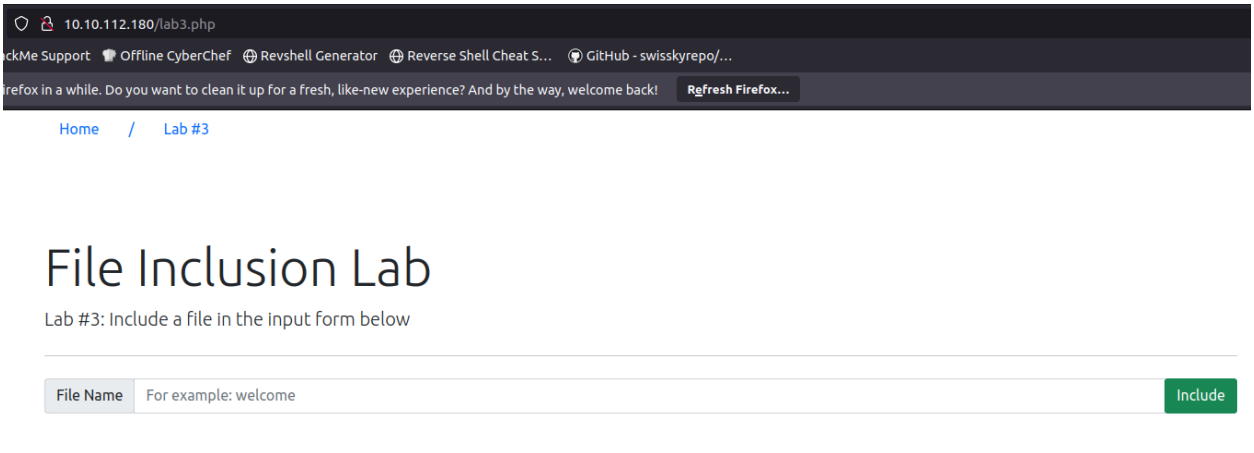
Answer: *includes*

Task 5: Local File Inclusion - LFI #2

Question 4: Give Lab #3 a try to read /etc/passwd. What is the request look like?

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

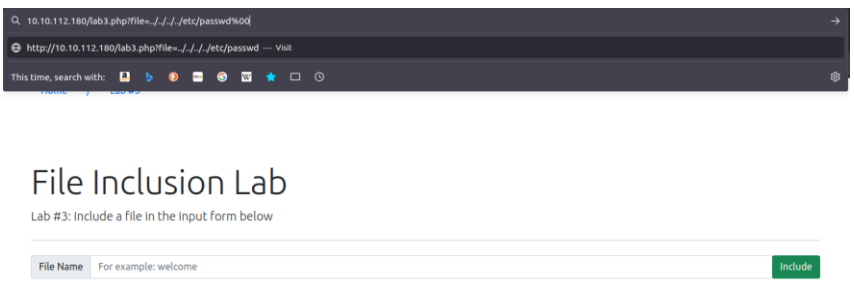
Lab #3



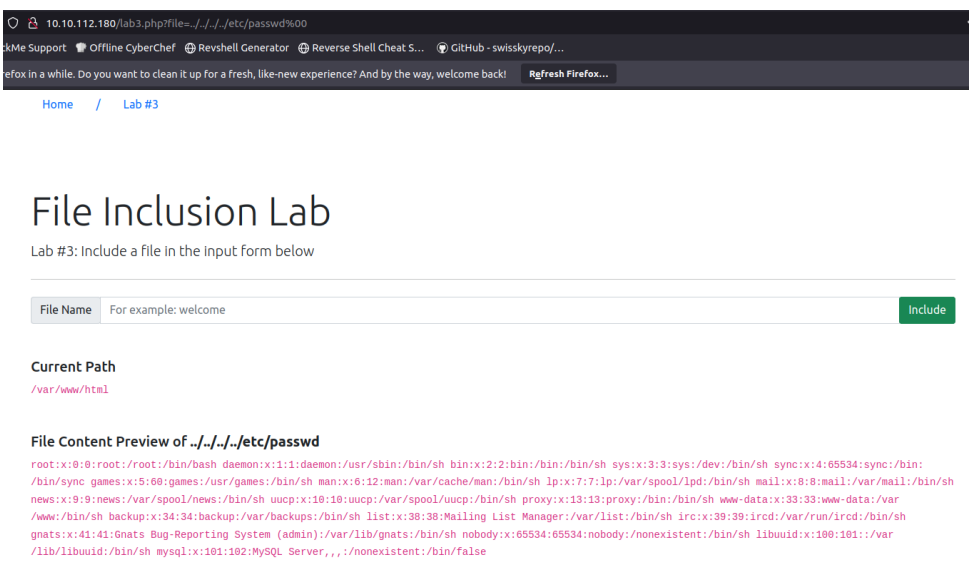
We need to inject the payload:

http://<MACHINE-IP>/lab3.php?file=../../../../etc/passwd%00

In our case, will be http://10.10.112.180/lab3.php?file=../../../../etc/passwd%00



The following page is displayed:

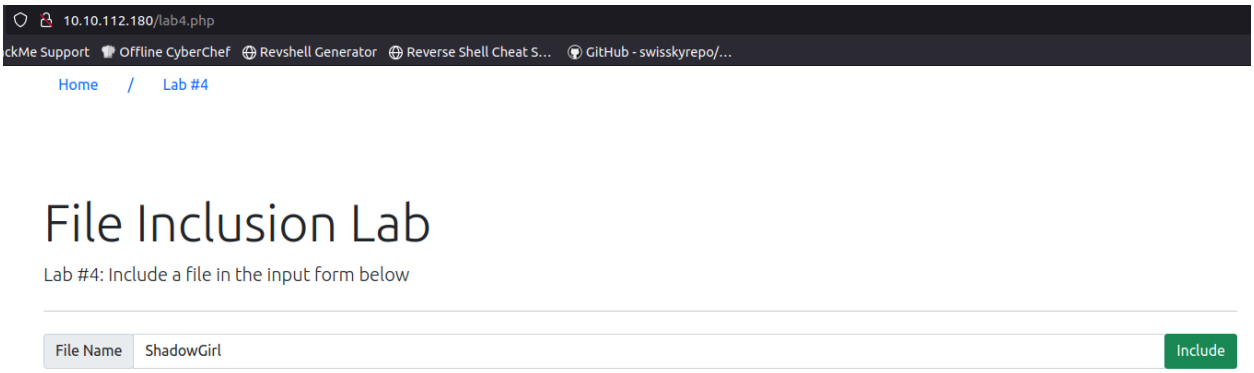


Question 5: Which function is causing the directory traversal in Lab #4?

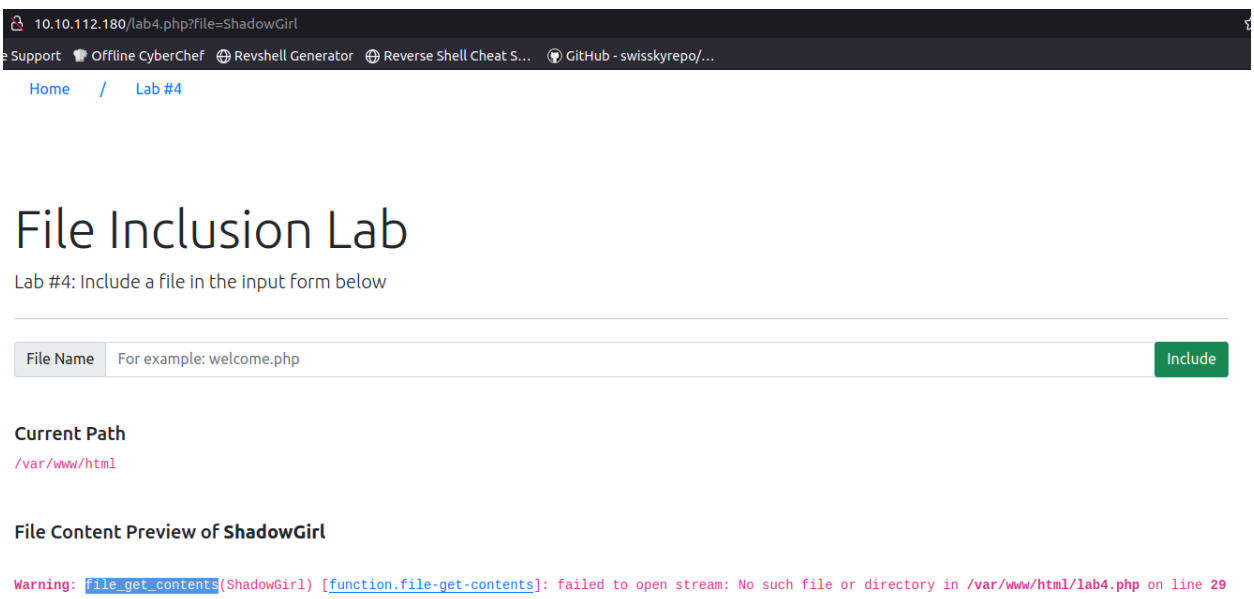
Answer: *file_get_contents*

Lab #4

Type “ShadowGirl”.



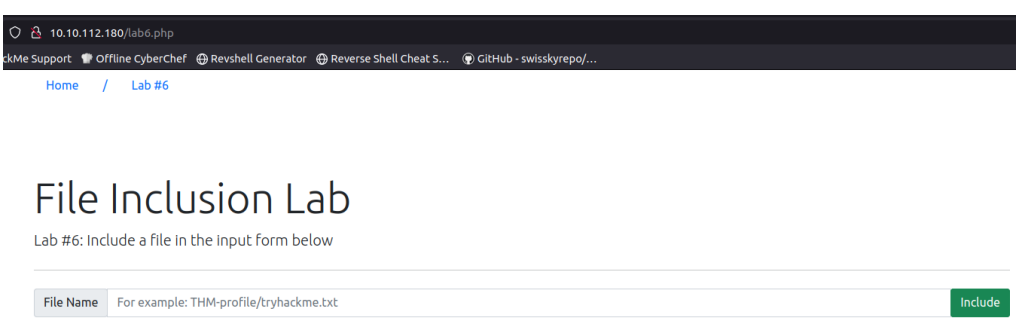
The following image is displayed:



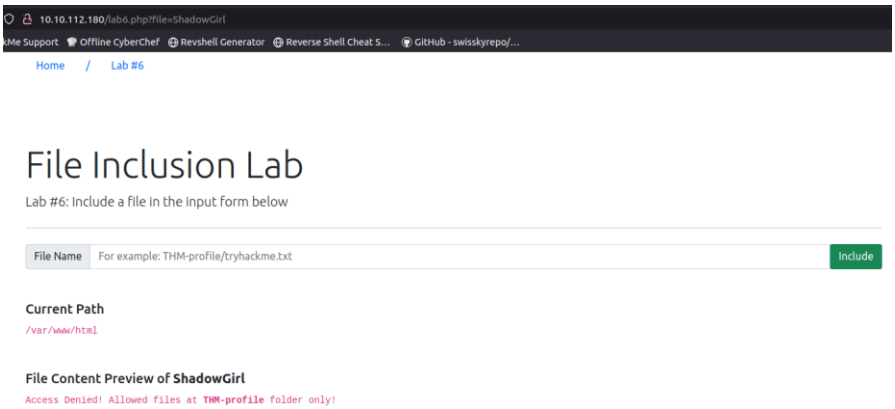
Question 6: Try out Lab #6 and check what is the directory that has to be in the input field?

Answer: *THM-profile*

Lab #6



Type “ShadowGirl”.

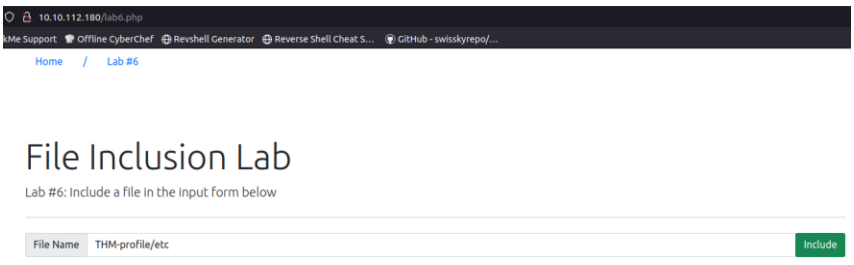


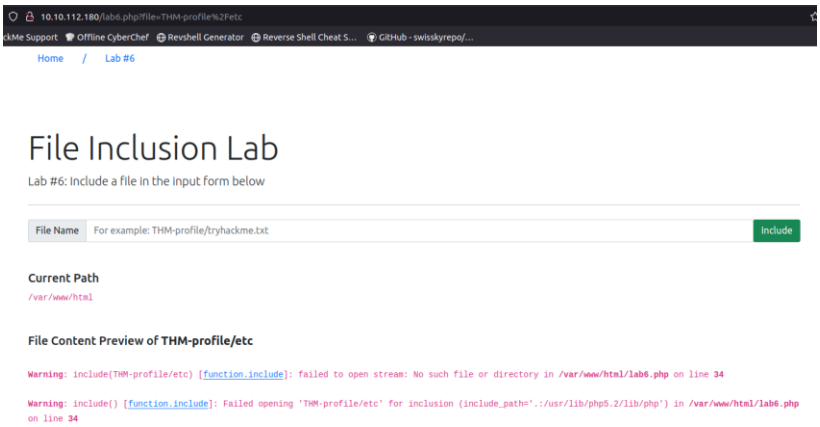
The directory is “THM-profile”.

Question 7: Try out Lab #6 and read /etc/os-release. What is the VERSION_ID value?

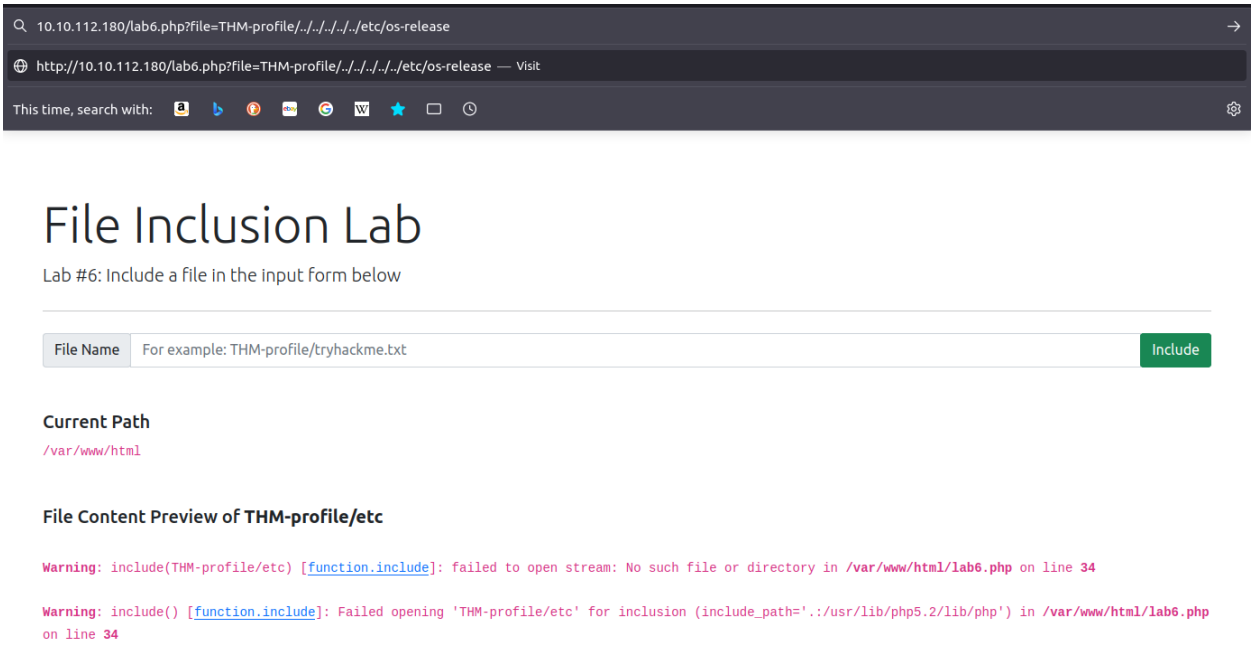
Answer: *12.04*

Type “THM-profile/etc”.

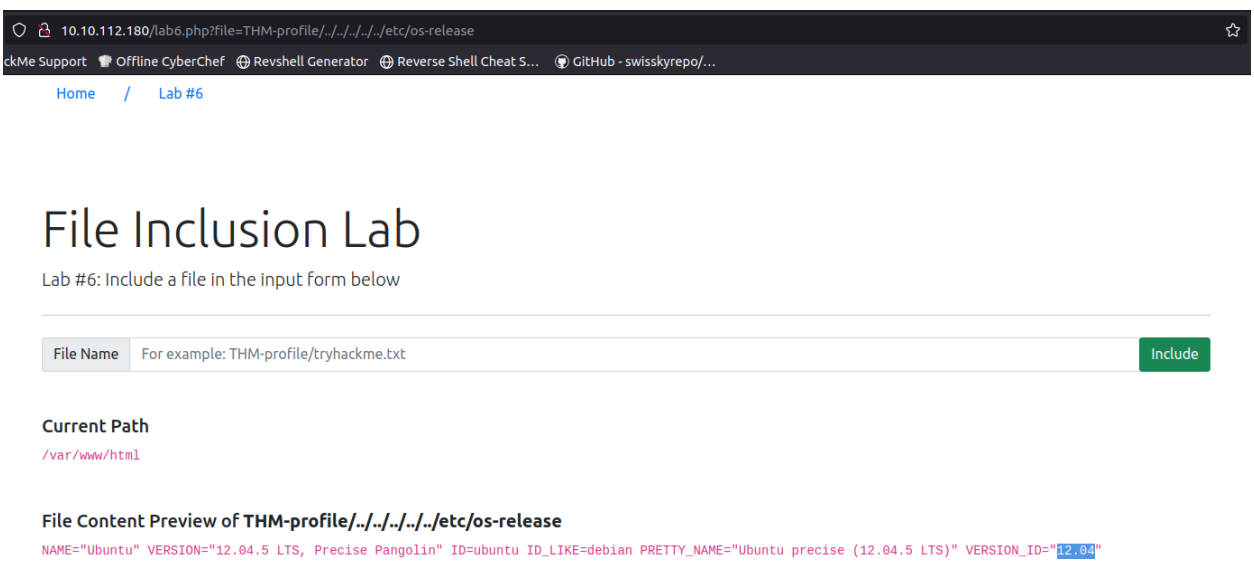




Then, go to the URL and after “THM-profile”, add “/../../../../../etc/os-release”.



The VERSION_ID is "12.04".



Task 6: Remote File Inclusion (RFI)

Remote File Inclusion (RFI) is a technique to include remote files and 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

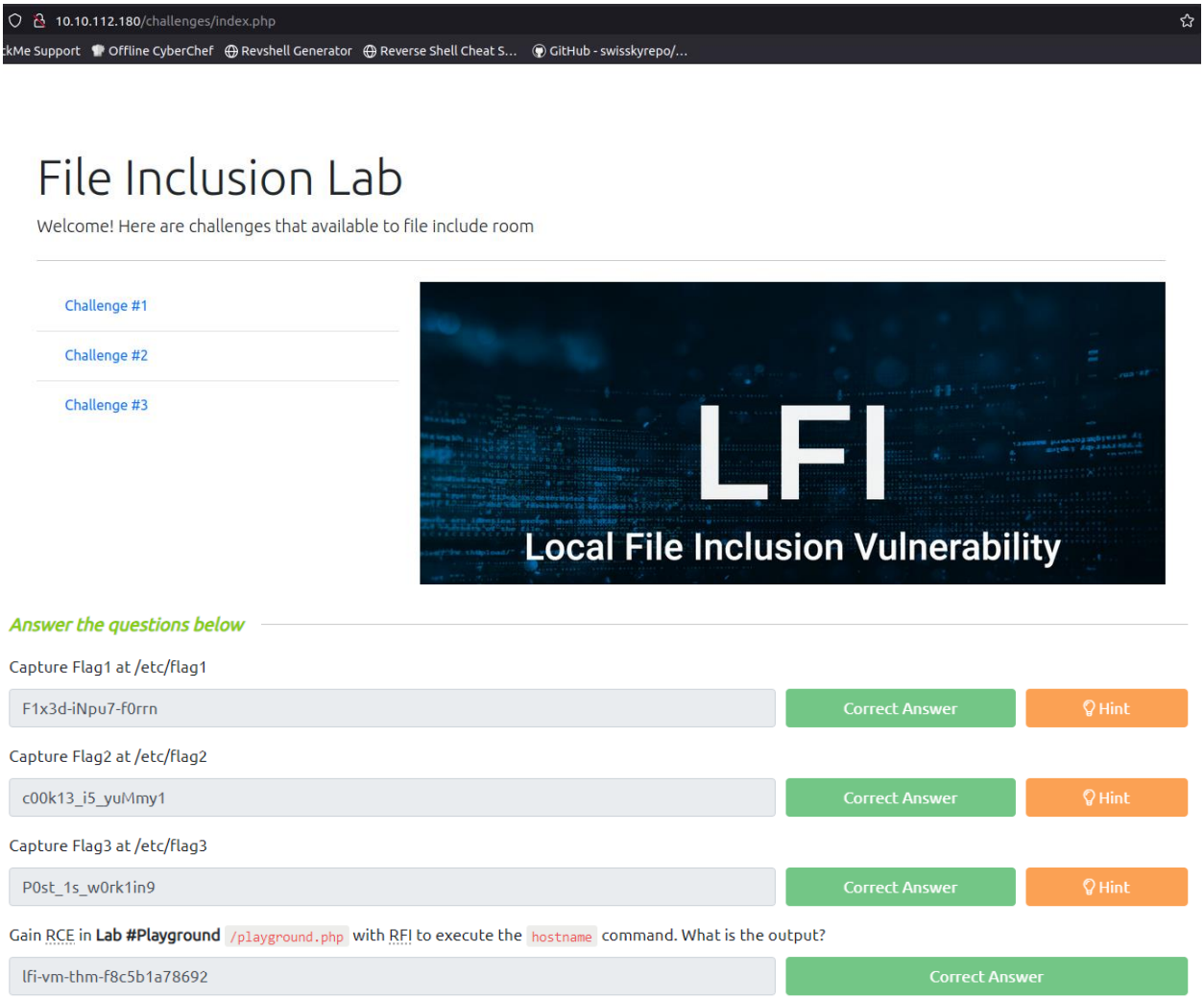
- Cross-site Scripting (XSS)
- Denial of Service (DoS)

Task 7: 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 are provided in the TryHackMe room.

Task 8: Challenge

Make sure the attached VM is up and running then visit: http://10.10.112.180/challenges/index.php



Happy Hacking! 😸

> *Thanks and Regards,*
>
> *ShadowGirl* 🦹‍♀️ 😉