**Bypass Disable Functions -TryHackMe**

This is a Informational room on TryHackMe that allows a specific upload vector to be learned and practiced using a intentionally vulnerable machine. I’ve found a lot of the available write-ups where they were simply a copy and paste from the Original room and barely made any sense; unless you wanted to just finish without gaining an understanding. Hopefully, this writeup will give an understanding.

------------------------------------------------------

**The Informational section**

In web applications that use PHP, there are functions that are considered dangerous due to their nature, specifically allowing code to be executed on the server. Naturally, we do not want random people uploading their own code onto a server and executing it. The functions that can be exploited are well known such as *system*() or *shell\_exec*(). As these are well known attack accomplices, they are commonly disabled as a security measure.

Another security measure that can be implemented on the web-server is restricting the base directory. This is means that even if a malicious script managed to slip though, its execution is only limited to the base directory.

However, addressed in 2008 was a PHP attack vector that made use of the putenv() and the mail() functionalities. They are known vectors but they are not usually not disabled. Putenv() allows for environment variables to be modified; specifically a .so library can be set to be pre-loaded using the LD\_PRELOAD so that if any program executes, the .so library is called first. Windows and Unix environments run differently so this technique better suits Unix environments due to a problem that Unix solves, even if a script is uploaded, it still needs a program to be called in order to run; in Unix, the mail() runs as a binary meaning it can be invoked.

So, in our upcoming lab, we have a server that restricts the base directory and does not allow scripts to be exeuted. Enter Chankro. Chankro is a Python based tool that allows for functions that have been disabled to be bypassed ( *disable\_functions*) along with the base directory restrictions. (*open\_basedir*). It is available on Github. <https://github.com/TarlogicSecurity/Chankro> and is straightforward to install and run. What it does is create a shared library on the server (.so library) and the binary/ script that we want to be executed. The *putenv*() and *mail*() are then called in order to launch the process.

Hopefully, now we understand, how this room works, the knowledge needed and we can successfully complete the machine.

------------------------------------------------------

**The Lab**

As this room is intended to specifically practice this attack vector it is correct to assume it is all web-based so we can skip the Port Scan as it will only confirm this. Browsing through the webapge, we find that we have the ability to upload our CV but as a image. There are two pieces of information that we need to know.

*#1: Where are the uploads stored?*

*#2: What are the upload restrictions in place?*

*#3: What is the server configuration?*

In order to answer the first question, we need to brute-force directories. It is also advised to look for files that have a .php extension seeing as we are dealing with a PHP web application.

*ffuf -u http://$IP\_ADDRESS/FUZZ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -c*

We found the files and the directories

*/uploads*

*/phpinfo.php*

Now we know where our uploaded files are located. Visiting in the browser we find directory listing is enabled meaning that we can open files directly. Now, our second question. The restriction is only images are allowed to be uploaded. There are different ways to try and bypass this restriction but none of the usual work except one using Burp proxy in conjunction with a form of magic bytes.

The extension used is GIF. An image with this extension is allowed to be uploaded and we can see it in the /uploads directory. We need to trick the server into thinking that our .php file is actually a GIF file. Keeping this information, we look at the phpinfo.php file in order to see if there is any other information concerning the server configuration. We find that the document root is located at */var/www/html/fa5fba5f5a39d27d8bb7fe5f518e00db*. Great, useful information.

Using our Chankro tool, we create our payload using

python2 chankro.py –arch 64 –input reverse.sh –output reverse.php –path /var/www/html/fa5fba5f5a39d27d8bb7fe5f518e00db

The reverse.sh is simply a bash srcipt that creates a reverse connection to a listener on our attacking machine.

Having Burp Proxy running and with the Intercept on, we upload the .php extension. In the intercept, we add the sequence **GIF87a** before the <?php –. This represents a GIF file and tricks the security measure into thinking it is of the allowed type.

The modified request is forwarded and thefile can be located in the /uploads directory. Make sure a listener is running on the machine with the Port set to match the port in the reverse shell script. Upon, opening the file in the browser, we should receive a reverse connection back.

Looking around the machine, the flag required is found in the /home directory.