TokyoGHOUL - CTF

Date Attempted - 10th July 2023 | 10:15am

Task 1

Deploying machine

Task 2

Scanning ports and checking out what os the machine is running.

Task 3

1. Inspecting can help alot. especially using 'view-page:source'
2. inspecting says anonymous users allowed in its own way, ftp port is open, login to ftp using anonymous. get all the files.
    1. make the file executable. run it. it asks for a password/paraphrase.
    2. Can use 'strings' or 'rabin2' (in some cases) to check the binary conversions. I used strings which gave me a bunch of data/commands as well as a paraphrase hidden between the same data/commands.
    3. when you enter the paraphrase after running the executable it gives you a paraphrase again.
    4. HInt is given that something is hidden in the file (.jpg) we got from ftp
    5. we can use steghide to check what all is hidden. it needs a paraphrase. so i used what i got from the executable and got the hidden text file.

Task 4

1. After opening the text file, it had morse code. I used cyberchef to automatically decode it. got the data in the form of hex values. decoded that again, got base64 encoding. decoded it again. in the wrote it was written that after all the decoding we get the hidden directory. So after going to URL/{hidden_directory} we get a page where it says scan me.
2. Scan me literally translates to scan the directories from my exp while doing CTFs so i quickly scanned the directories in **"URL/hidden_directory_we_found_earlier/"**
3. we get a hidden directory and we proceed to look into it. It treats us with an index.php page where it asks us to go back to MAIN PAGE, NO, YES to a question. MAIN PAGE let us go back to index.php page but option NO and YES redirects to a specific URL.
4. This is where LFI (Local File Intrusion) comes into play.

LFI (Local File Intrusion)

An attacker can use Local File Inclusion (LFI) to trick the web application into exposing or running files on the web server. An LFI attack may lead to information disclosure, remote code execution, or even Cross-site Scripting (XSS). Typically, LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement.

Local File Inclusion is very similar to Remote File Inclusion (RFI). However, an attacker using LFI may only include local files (not remote files like in the case of RFI).

The following is an example of PHP code that is vulnerable to LFI.

```
/**
* Get the filename from a GET input
* Example - http://example.com/?file=filename.php
*/
$file = $_GET['file'];

/**
* Unsafely include the file
* Example - filename.php
*/
include('directory/' . $file);
```

In the above example, an attacker could make the following request. It tricks the application into executing a PHP script such as a web shell that the attacker managed to upload to the web server.

```
http://example.com/?file=../../uploads/evil.php
```

In this example, the file uploaded by the attacker will be included and executed by the user that runs the web application. That would allow an attacker to run any server-side malicious code that they want.

This is a worst-case scenario. An attacker does not always have the ability to upload a malicious file to the application. Even if they did, there is no guarantee that the application will save the file on the same server where the LFI vulnerability exists. Even then, the attacker would still need to know the disk path to the uploaded file.

Directory Traversal

Even without the ability to upload and execute code, a Local File Inclusion vulnerability can be dangerous. An attacker can still perform a Directory Traversal / Path Traversal attack using an LFI vulnerability as follows

```
http://example.com/?file=../../../../etc/passwd
```

In the above example, an attacker can get the contents of the /etc/passwd file that contains a list of users on the server. Similarly, an attacker may leverage the Directory Traversal vulnerability to access log files (for example, Apache access.log or error.log), source code, and other sensitive information. This information may then be used to advance an attack.

Bypassing and exploiting LFI if ../etc/passwd or other directories are blocked

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/File%20Inclusion#lfi-to-rce-via-procfd

https://book.hacktricks.xyz/pentesting-web/file-inclusion#lfi-2-rce

after encoding a lot (URL Encoding):

```
%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2
E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2Fetc%2Fpasswd
```

and using it we get, the password and username for the ssh. Below is the password hash:

```
$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it
8r.jbrlpfZeMdwD3B0fGxJI0
```

5. after finding out what kind of hash this was, we can use john (john the ripper) to quickly crack the hash. wordlist used = rockyou.txt
6. We can now ssh into the machine and get the user.txt flag.
7. After that we can check what all commands can the user run as sudo using

```
sudo -l
```

we see the user can execute python files as sudo and one specific python file called jail.py but we cannot modify it.

8. So after taking a look at the python file using cat command we see that the python file will not allow any imports (importing a module), or anything else, we have to use python jailbreak. Using builtin commands/instructions to either escalate privileges or read the file.
9. Command used:

```
__builtins__.__dict__['__IMPORT__'.lower()
('PTY'.lower()).__dict__['SPAWN'.lower()]('/bin/bash')
```

this specific command uses builtin function to call dict function which imports PTY module which can be used to spawn a bash shell as sudo/root.
the above command translates to:

```
import pty
pty.spawn('/bin/bash')
```

which is executed as root. This gives a root bash shell.

10. Now we can navigate to the root directory and grab the root.txt flag.