EHAX RECRUITMENT CTF WRITE UP

Challenge 1: Music_Pages
Category: Web Exploitation
Description: "Aur bhi pages hai life mein"

- Divyansh Singh (divxcode/dsunreal)

>> THOUGHT PROCESS:

1. Considering the description, it is evident that we have to find more pages on the server than the ones which are visible on the homepage.

2. There are a couple of things we can do - As in any web exploit, we would obviously start by looking at the page source code and check for hidden comments in the html file or any hint to the hidden page.

3. Next we can use developer tools to check for anything else which is not in source code (like cookies, sources, network activity, network requests etc.)
   side note: I didn't explain these terms as this challenge is not about them

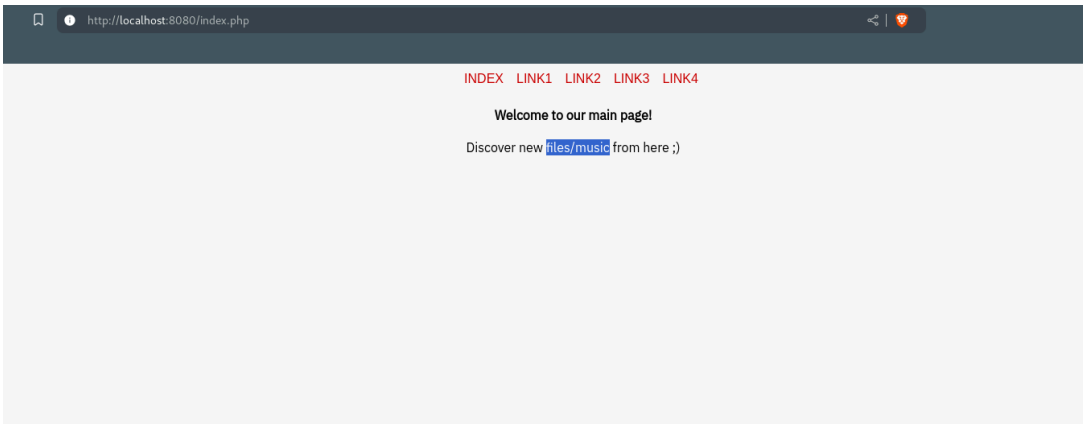
4. The other thing that we can work on is - after looking at the url carefully (http://localhost:8080/index.php) - we can tell that the server uses php at that backend. PHP has a lot of known vulnerabilities. Maybe something on those lines?

   // How would we navigate the page? there are only a few buttons (INDEX, LINK 1, LINK2 and so on.. There nothing on the page to click to go to)

   // We don't need buttons to go to a page on a website. We typically navigate a webpage using the hyperlinks (links/buttons) present on the page. But we can also navigate it by directly typing the url in the search bar of the browser. The hyperlinks basically do the same thing (take us to a url).

   // If the page is there on the server, and accessible, then it will show otherwise, it will give a 403 (Forbidden) or 404 (Not Found) Error or any other depending on the condition.

5. We want to find the hidden “files” somehow on the server which are accessible to us. That hidden file must contain our flag. (the index page says “files/music”, but on links only music is there, so there must be files as well, yet another hint confirming our hunch)



>> APPROACH

- Reading Source Code - This approach failed as there was nothing special in the page source. No hints/hidden comments

- Using Developer Tools - This approach also failed. No useful information could be gathered using the developer tools/inspector window.

- Discovering LFI Vulnerability - I searched for PHP server vulnerabilities and read through some articles, asked gpt and also watched a few videos on the topic of LFI.
  https://www.youtube.com/watch?v=yq2rq50IMSQ
  https://www.youtube.com/watch?v=rAeM33MNAUo

  During the exploration, I noticed that the ‘view’ parameter in the URL was used to include the files on the server (music1,music2 etc). This was talked about in the above videos. This thing hinted at a possible Local File Inclusion (LFI) vulnerability.

// LFI is a type of exploit when server takes user input (like the 'view' parameter in this case) and directly uses it to determine which files to include, without proper checking

// We can see this in the index.php source code as well (given with dockerfile). It takes input from the user & sets its value. When a user accesses the page with a URL like index.php?view=music1.html, the value of $_GET['view'] becomes music1.html.

// The include($_GET['view']); line will then effectively become include('music1.html'); causing the server to include the content of music1.html and display it on the page

//-! The problem with this is that it doesn't check the user input. AND! Whenever we take input from a user (a string or anything) and don't filter it, it always makes that code vulnerable. This same concept is also used in buffer overflow attack, where the gets() function takes user input and doesn't check bounds.

```
16          </div>
17 <?php
18
19 if(!isset($_GET['view']) || ($_GET['view']=="index.php")) {
20     echo" <p><b>Welcome to our main page!</b><br><br>Discover new files/music
   from here ;)</p>";
21 }
22 else {
23          echo "<p>";
24          // include("/var/www/html/" .$_GET['view']);
25          include($_GET['view']);
26          echo "</p>";
27 }
28 ?>
29     </body>
30 </html>
```

Knowing that this is most probably an LFI vulnerability. My next move was clear, Trying to access files by passing parameters in the view function.

I observed the pattern in url naming. The links follow a simple sequence - music1,music2,music3 etc.
http://localhost:8080/index.php?view=music3.html;
http://localhost:8080/index.php?view=music4.html;

So, I Tried other patterns like music5, music6 and so on… Maybe those pages existed but weren't listed in the index page. I also tried file1, file2 and so on but it didn't work. Moreover, how many could I try manually?

So, I created a python script (*with some help from gpt as I know only c & cpp. I only know python's basics*) to automate this brute forcing attack.

```
import requests

# Define the base URL of the site and possible extensions
base_url = 'http://98.70.76.80:8006/index.php?view='
extensions = [".html", ".php", ".txt", ".asp", ".jsp", ".py"]
words_to_try = ["music", "video", "files", "docs"]  # Add other words to
try

# Function to check if the page contains the text "ehax"
def check_for_ehax(page_content):
    return "ehax" in page_content

# Iterate through different words and file extensions
for word in words_to_try:
    for num in range(1, 20):
        for ext in extensions:
            url = f"{base_url}{word}{num}{ext}"
            try:
                response = requests.get(url)
                if response.status_code == 200:  # Check if page exists
                    print(f"Found something at: {url}")
                    if check_for_ehax(response.text):
                        print("Page contains 'ehax'")
                    else:
                        print("Page does not contain 'ehax'")
            except requests.RequestException as e:
                print(f"Request failed: {e}")
```

This code automates bruteforcing various possible words with sequence numbers and extensions. Not only this, it also scans that page for the keyword "ehax" which is part of the flag. (Its like 2 shots with 1 bullet, not only I search for hidden pages but also find if it contains the flag)

But, This approach failed and I was thinking of doing gobuster but that would take time and it was not guaranteed that it would work.

// Gobuster is a tool that does almost exactly this but instead of manually supplying each and every word, it utilizes a word list that contains common words. It is a brute force attack tool and if used excessively it may trigger websites' DOS (Denial of Service) or Intrusion Detection systems and block the ip from which we are sending these requests.

- Later, the hint “git repository” was dropped in the discord channel. Then, It occurred to me that I could have tried git’s pages also instead of just file1 file2 etc.

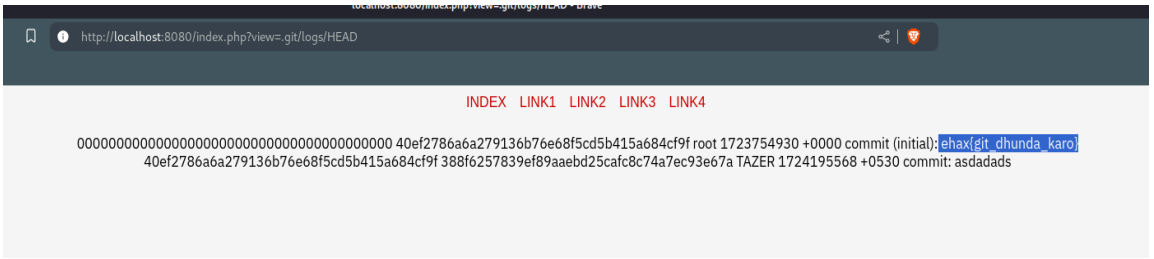
  // git is a version-control software which basically allows many coders to collaborate on a single project and also keep track of changes made in the project throughout its development.

  // A folder that is tracked by git for changes is called a git repository, and where exactly does this software store all these changes’ data? In the same repository itself in the hidden “.git” folder.
  Every “.git” folder has the same structure of folders/files inside it. It contains data pertaining to the repository which are stored in files, namely -
  1. *config*
  2. *description*
  3. *HEAD*
  4. *index*
  5. *logs > HEAD*
  6. *logs > refs > …*
  7. *info > …*

     And so on

  // This is not a secret and known to everyone who uses git. This structure of files is standard and public.

- From here on things were easy. I just needed to filter through the git’s files and look for the flag. Now, it was just a matter of time 🙂



- The flag was finally found in the .git/logs/HEAD file (http://localhost:8080/index.php?view=.git/logs/HEAD)
- `>> FLAG: ehax{git_dhunda_karo} <<`

>> FINAL-THOUGHTS / CONCLUSION

This was quite an easy challenge *considering* that it is discovered, that we had to explore git pages. Without the hint it would be like shooting arrows blindly. Although, when I later tried Go buster, it surprisingly discovered the .git/config page! Maybe, I should have tried it earlier on instead of doubting it.
