WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Sruthi Sridhar Bopparthi

Email: bopparsr@mail.uc.edu

Figure 1: Sruthi's Headshot

Repository Information

Repository's URL: https://github.com/SruthiAelay/waph-bopparsr.git

This is a private repository which is used to store all the codes related to course Topics in Computer Systems. The structure of this repository is as mentioned below.

Lab 4 - A Secure Login System with Session Authentication

Lab's overview

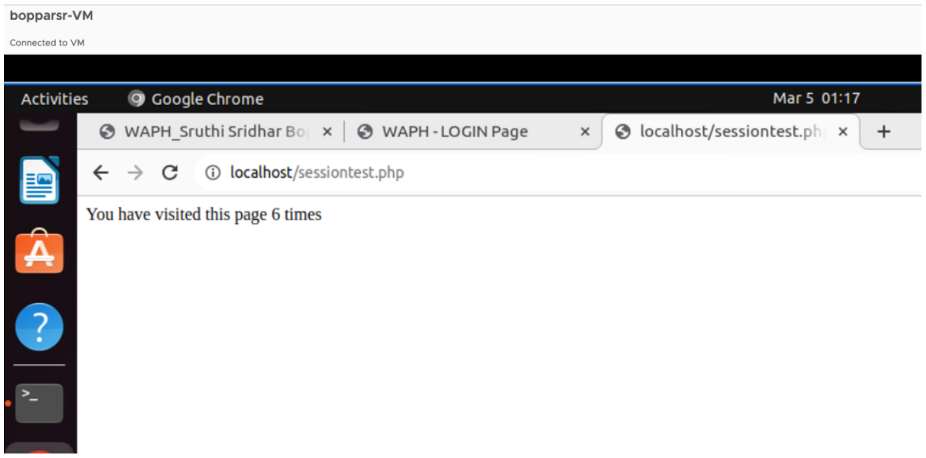
In this lab, we focus on implementing and securing session management in PHP web applications. Our goals include deploying and testing sessiontest.php, understanding session handling processes through Wireshark, and identifying and mitigating session hijacking vulnerabilities. By completing this lab, we, as students, aim to gain practical experience in deploying session management, observing web traffic, and implementing secure authentication measures. The tasks involve revising the login system, simulating session hijacking attacks, and enhancing security with measures like HTTPS. Through hands-on exercises, we strengthen our understanding of PHP web application security, ensuring we can effectively protect against session hijacking and other potential vulnerabilities. The lab aims to provide us with valuable insights and practical skills in securing web applications.

Link to Lab4 code : https://github.com/SruthiAelay/waph-bopparsr/tree/main /labs/lab4

Task 1: Understanding Session Management in a PHP Web Application

a) Deploy and test sessiontest.php

In this sub-task, we need to clone the course repository, make necessary revisions to the sessiontest.php file, deploy it to the web server, and then access it using different web browsers like Chrome and Firefox. I have tested in both the browsers and according to screenshots it shows that each browser has it own session while interacting with the web browsers.



You have visited this page 6 times

Figure 2: Sessiontest in Chrome

b) Observe the Session-Handshaking using Wireshark

To perform the Wireshark sub-task, we began by downloading and installing Wireshark from the official website. After opening the application, we selected the network interface connecting our computer to the internet and initiated packet capturing. Prior to accessing sessiontest.php, we ensured the browser's cookies were cleared to maintain a clean session. Upon accessing the page, we stopped the packet capture in Wireshark. To isolate relevant packets, we

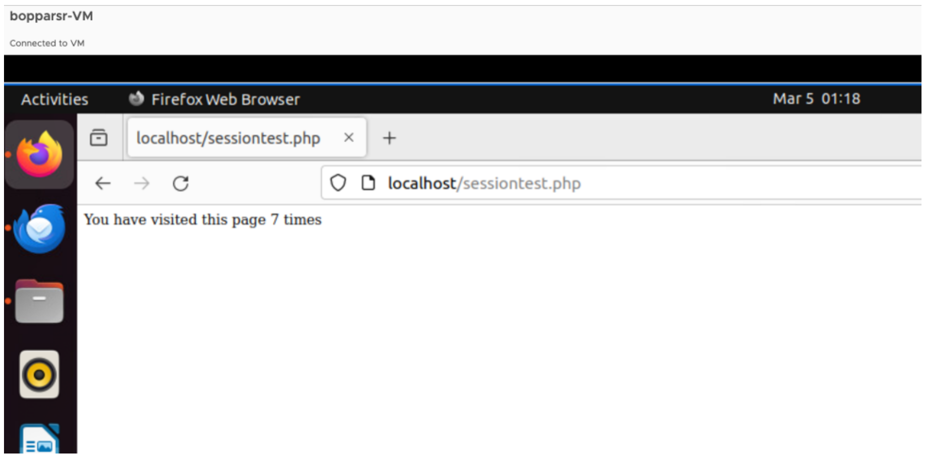

You have visited this page 7 times

Figure 3: Sessiontest in Firefox

applied a display filter focusing on HTTP requests and responses related to sessiontest.php. By analyzing these packets, we gained insights into the session handshaking process, identifying key information exchanged. We can observe how cookies were set and maintained.

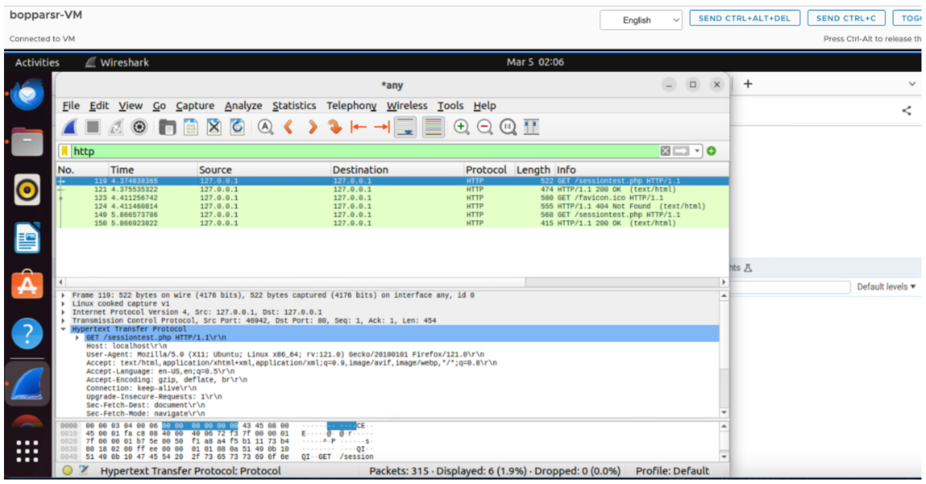
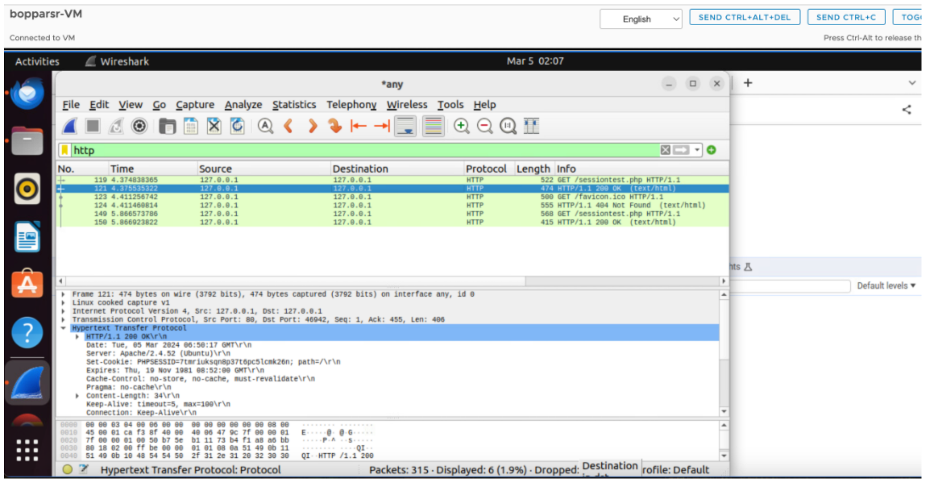

```
No.  Time         Source     Destination  Protocol  Length  Info
119  4.374838365  127.0.0.1  127.0.0.1    HTTP      522     GET /sessiontest.php HTTP/1.1
121  4.375535322  127.0.0.1  127.0.0.1    HTTP      474     HTTP/1.1 200 OK  (text/html)
123  4.411256742  127.0.0.1  127.0.0.1    HTTP      500     GET /favicon.ico HTTP/1.1
124  4.411460814  127.0.0.1  127.0.0.1    HTTP      555     HTTP/1.1 404 Not Found  (text/html)
149  5.866573786  127.0.0.1  127.0.0.1    HTTP      568     GET /sessiontest.php HTTP/1.1
150  5.866923822  127.0.0.1  127.0.0.1    HTTP      415     HTTP/1.1 200 OK  (text/html)

Frame 119: 522 bytes on wire (4176 bits), 522 bytes captured (4176 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 46942, Dst Port: 80, Seq: 1, Ack: 1, Len: 454
Hypertext Transfer Protocol
  GET /sessiontest.php HTTP/1.1\r\n
  Host: localhost\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Sec-Fetch-Dest: document\r\n
  Sec-Fetch-Mode: navigate\r\n

0000  00 00 03 04 00 06 00 00  00 00 00 00 43 45 08 00   ············CE··
0010  45 00 01 fa c8 08 40 00  40 06 72 f3 7f 00 00 01   E·····@·@·r·····
0020  7f 00 00 01 b7 5e 00 50  f1 a8 a4 f5 b1 11 73 b4   ·····^·P······s·
0030  80 18 02 00 ff ee 00 00  01 01 08 0a 51 49 0b 10   ············QI··
0040  51 49 0b 10 47 45 54 20  2f 73 65 73 73 69 6f 6e   QI··GET /session

Hypertext Transfer Protocol: Protocol    Packets: 315 · Displayed: 6 (1.9%) · Dropped: 0 (0.0%)    Profile: Default
```

Figure 4: Http Request

c) Understanding Session Hijacking

Performing a session hijacking attack involves exploiting vulnerabilities in the session management of a web application. In this case, following the steps outlined in the lecture, we simulated a session hijacking attack on sessiontest.php.



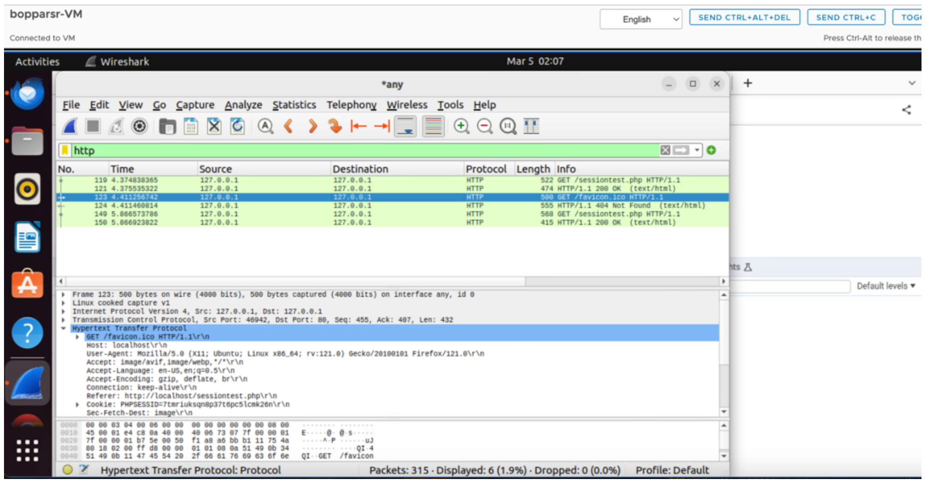
```
No.  Time         Source     Destination  Protocol  Length  Info
119  4.374838365  127.0.0.1  127.0.0.1    HTTP      522 GET /sessiontest.php HTTP/1.1
121  4.375535322  127.0.0.1  127.0.0.1    HTTP      474 HTTP/1.1 200 OK  (text/html)
123  4.411256742  127.0.0.1  127.0.0.1    HTTP      500 GET /favicon.ico HTTP/1.1
124  4.411460814  127.0.0.1  127.0.0.1    HTTP      555 HTTP/1.1 404 Not Found  (text/html)
149  5.866573786  127.0.0.1  127.0.0.1    HTTP      568 GET /sessiontest.php HTTP/1.1
150  5.866923822  127.0.0.1  127.0.0.1    HTTP      415 HTTP/1.1 200 OK  (text/html)
```

```
Frame 121: 474 bytes on wire (3792 bits), 474 bytes captured (3792 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 46942, Seq: 1, Ack: 455, Len: 406
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Tue, 05 Mar 2024 06:50:17 GMT\r\n
    Server: Apache/2.4.52 (Ubuntu)\r\n
    Set-Cookie: PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n; path=/\r\n
    Expires: Thu, 19 Nov 1981 08:52:00 GMT\r\n
    Cache-Control: no-store, no-cache, must-revalidate\r\n
    Pragma: no-cache\r\n
    Content-Length: 34\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 01 ca f3 8f 40 00  40 06 47 9c 7f 00 00 01   E·····@· @·G·····
0020  7f 00 00 01 00 50 b7 5e  b1 11 73 b4 f1 a8 a6 bb   ·····P·^ ··s·····
0030  80 18 02 00 ff be 00 00  01 01 08 0a 51 49 0b 11   ········ ····QI··
0040  51 49 0b 10 48 54 54 50  2f 31 2e 31 20 32 30 30   QI··HTTP /1.1 200
```

Hypertext Transfer Protocol: Protocol   Packets: 315 · Displayed: 6 (1.9%) · Dropped: Profile: Default

Figure 5: Http Request Set Cookie



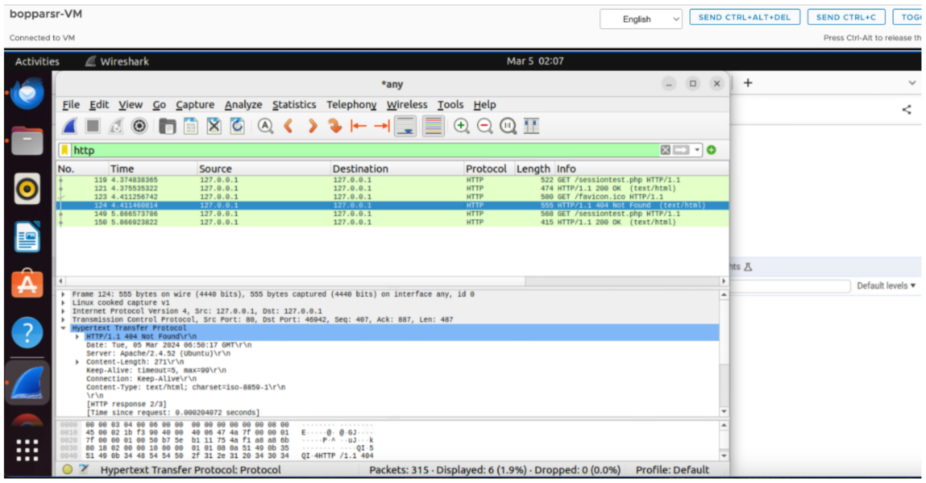
```
No.  Time         Source     Destination  Protocol  Length  Info
119  4.374838365  127.0.0.1  127.0.0.1    HTTP      522 GET /sessiontest.php HTTP/1.1
121  4.375535322  127.0.0.1  127.0.0.1    HTTP      474 HTTP/1.1 200 OK  (text/html)
123  4.411256742  127.0.0.1  127.0.0.1    HTTP      500 GET /favicon.ico HTTP/1.1
124  4.411460814  127.0.0.1  127.0.0.1    HTTP      555 HTTP/1.1 404 Not Found  (text/html)
149  5.866573786  127.0.0.1  127.0.0.1    HTTP      568 GET /sessiontest.php HTTP/1.1
150  5.866923822  127.0.0.1  127.0.0.1    HTTP      415 HTTP/1.1 200 OK  (text/html)
```

```
Frame 123: 500 bytes on wire (4000 bits), 500 bytes captured (4000 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 46942, Dst Port: 80, Seq: 455, Ack: 407, Len: 432
Hypertext Transfer Protocol
    GET /favicon.ico HTTP/1.1\r\n
    Host: localhost\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
    Accept: image/avif,image/webp,*/*\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate, br\r\n
    Connection: keep-alive\r\n
    Referer: http://localhost/sessiontest.php\r\n
    Cookie: PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n\r\n
    Sec-Fetch-Dest: image\r\n
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 01 e4 c8 0a 40 00  40 06 73 07 7f 00 00 01   E·····@· @·s·····
0020  7f 00 00 01 b7 5e 00 50  f1 a8 a6 bb b1 11 75 4a   ·····^·P ······uJ
0030  80 18 02 00 ff d8 00 00  01 01 08 0a 51 49 0b 34   ········ ····QI·4
0040  51 49 0b 11 47 45 54 20  2f 66 61 76 69 63 6f 6e   QI··GET  /favicon
```

Hypertext Transfer Protocol: Protocol   Packets: 315 · Displayed: 6 (1.9%) · Dropped: 0 (0.0%)   Profile: Default

Figure 6: Http Request with Cookie



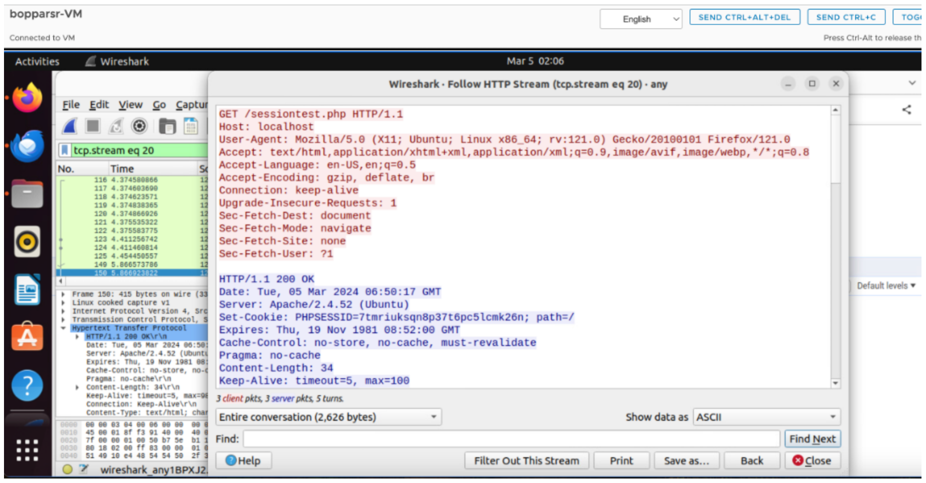
```
No.  Time          Source     Destination  Protocol  Length  Info
119  4.374838365   127.0.0.1  127.0.0.1    HTTP      522     GET /sessiontest.php HTTP/1.1
121  4.375535322   127.0.0.1  127.0.0.1    HTTP      474     HTTP/1.1 200 OK  (text/html)
123  4.411256742   127.0.0.1  127.0.0.1    HTTP      500     GET /favicon.ico HTTP/1.1
124  4.411460814   127.0.0.1  127.0.0.1    HTTP      555     HTTP/1.1 404 Not Found  (text/html)
149  5.866573786   127.0.0.1  127.0.0.1    HTTP      568     GET /sessiontest.php HTTP/1.1
150  5.866923822   127.0.0.1  127.0.0.1    HTTP      415     HTTP/1.1 200 OK  (text/html)
```

```
Frame 124: 555 bytes on wire (4440 bits), 555 bytes captured (4440 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 46942, Seq: 407, Ack: 887, Len: 487
Hypertext Transfer Protocol
  HTTP/1.1 404 Not Found\r\n
  Date: Tue, 05 Mar 2024 06:50:17 GMT\r\n
  Server: Apache/2.4.52 (Ubuntu)\r\n
  Content-Length: 271\r\n
  Keep-Alive: timeout=5, max=99\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=iso-8859-1\r\n
  \r\n
  [HTTP response 2/3]
  [Time since request: 0.000204072 seconds]
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 02 1b f3 90 40 00  40 06 47 4a 7f 00 00 01   E·····@· @·GJ····
0020  7f 00 00 01 00 50 b7 5e  b1 11 75 4a f1 a8 a8 6b   ·····P·^ ··uJ···k
0030  80 18 02 00 00 10 00 00  01 01 08 0a 51 49 0b 35   ········ ····QI·5
0040  51 49 0b 34 48 54 54 50  2f 31 2e 31 20 34 30 34   QI·4HTTP /1.1 404
```

Hypertext Transfer Protocol: Protocol   Packets: 315 · Displayed: 6 (1.9%) · Dropped: 0 (0.0%)   Profile: Default

Figure 7: Http Request with No Cookie



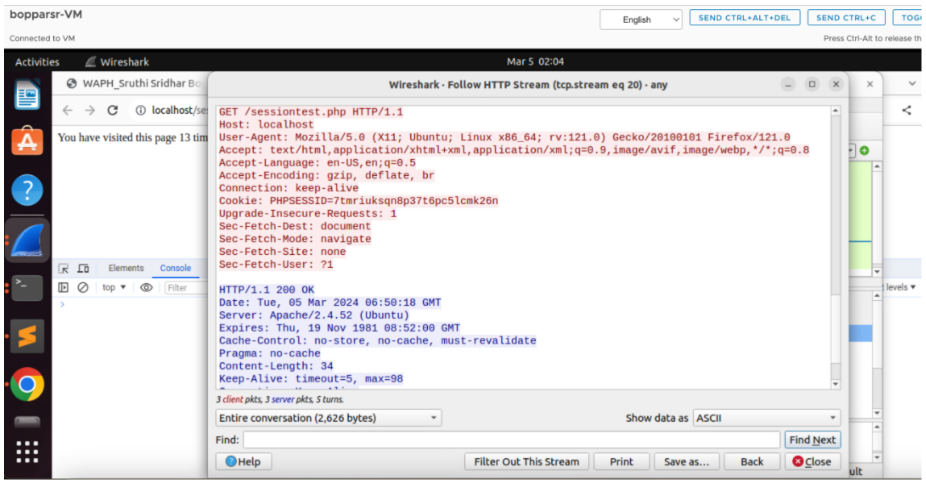
Wireshark · Follow HTTP Stream (tcp.stream eq 20) · any

```
GET /sessiontest.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

HTTP/1.1 200 OK
Date: Tue, 05 Mar 2024 06:50:17 GMT
Server: Apache/2.4.52 (Ubuntu)
Set-Cookie: PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 34
Keep-Alive: timeout=5, max=100
```

3 client pkts, 3 server pkts, 5 turns.

Entire conversation (2,626 bytes)   Show data as ASCII

Figure 8: Http 1st Request Response

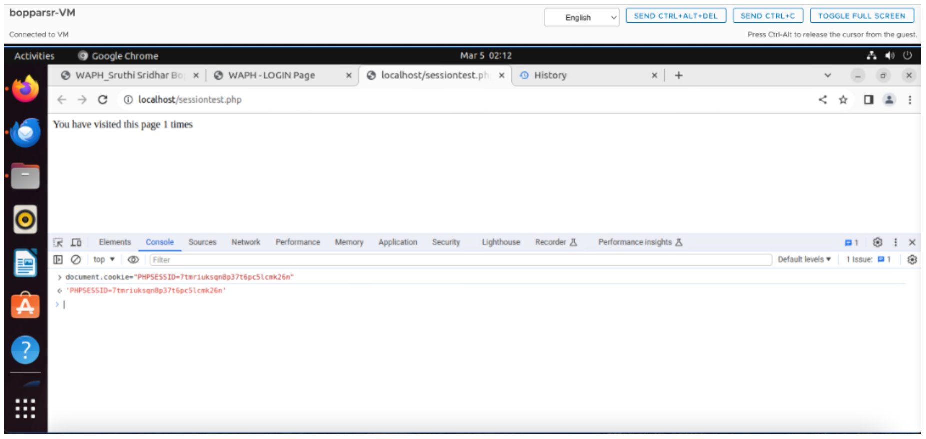


```
GET /sessiontest.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1

HTTP/1.1 200 OK
Date: Tue, 05 Mar 2024 06:50:18 GMT
Server: Apache/2.4.52 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 34
Keep-Alive: timeout=5, max=98
```

You have visited this page 13 tim

Figure 9: Http after 1st Request Response

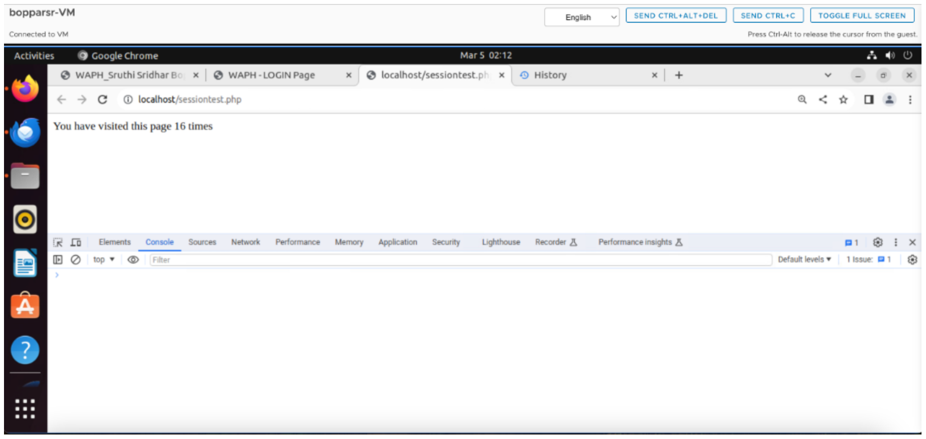
We intercepted the session cookie of a legitimate user by capturing session cookies and replicate session cookies. Once the session cookie was obtained, we used it to impersonate the legitimate user by injecting the captured session ID into another browser. By accessing sessiontest.php with the hijacked session, we demonstrated unauthorized access to the user's session, showcasing the potential risks associated with session hijacking.



You have visited this page 1 times

```
> document.cookie="PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n"
<- 'PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n'
> |
```

Figure 10: Copied session cookie into another browsers



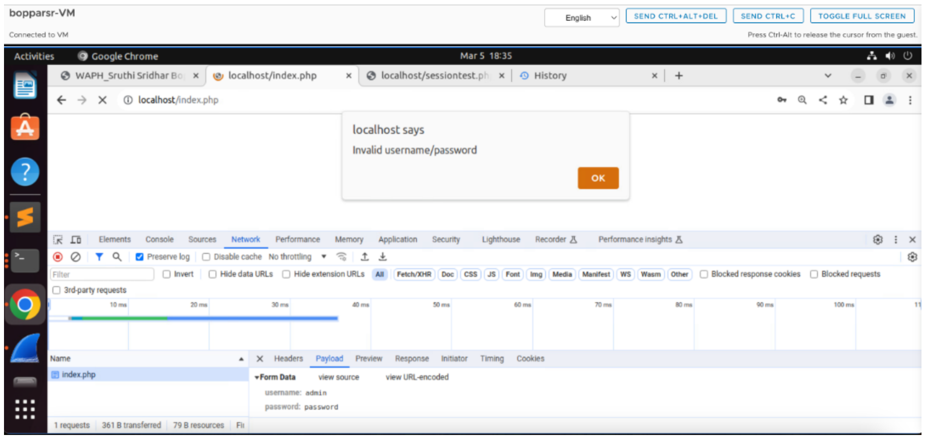
You have visited this page 16 times

Figure 11: Session is Hijacked

Task 2: Insecure Session Authentication

a) Revised Login System with Session Management

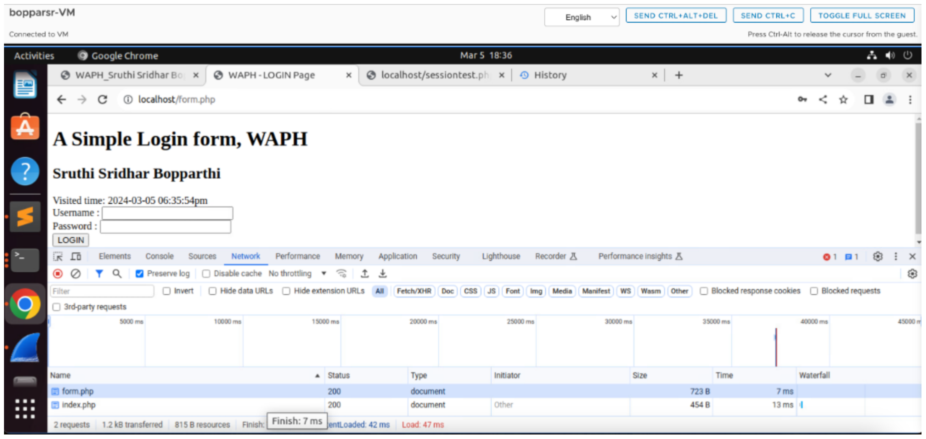
To implement session management for a login system, started by copying the content of the index.php file from either the lab3 or lab4 folder. Create a new file named logout.php to handle logout functionality. Revise the copied index.php file to integrate session management features. Deployed both files to web server and tested the login functionality by entering valid credentials. Additionally, tested the logout functionality by accessing logout.php to ensure the session is properly destroyed.



localhost says
Invalid username/password

```
▾Form Data    view source    view URL-encoded
  username: admin
  password: password
```

Figure 12: Logged in with invalid Credentials



A Simple Login form, WAPH

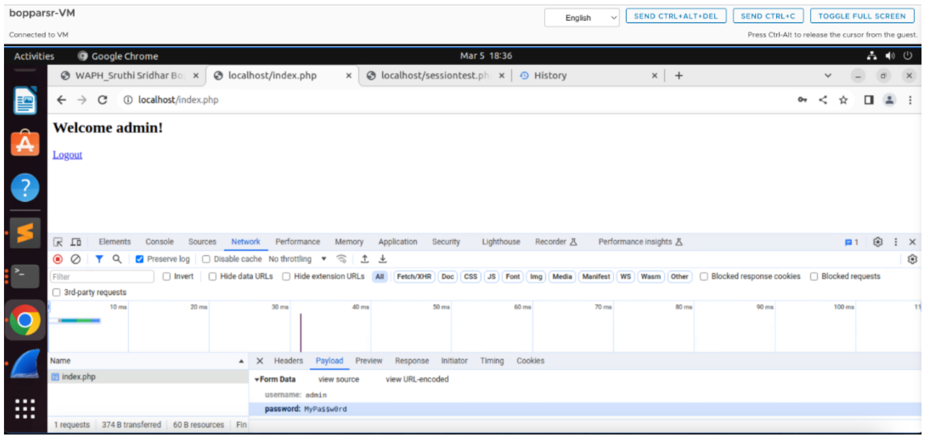
Sruthi Sridhar Boppparthi

Visited time: 2024-03-05 06:35:54pm
Username :
Password :
LOGIN

Name	Status	Type	Initiator	Size	Time	Waterfall
form.php	200	document		723 B	7 ms	
index.php	200	document	Other	454 B	13 ms	

2 requests | 1.2 kB transferred | 815 B resources | Finish: 7 ms | DOMContentLoaded: 42 ms | Load: 47 ms

Figure 13: Redirected to Login Form



Welcome admin!

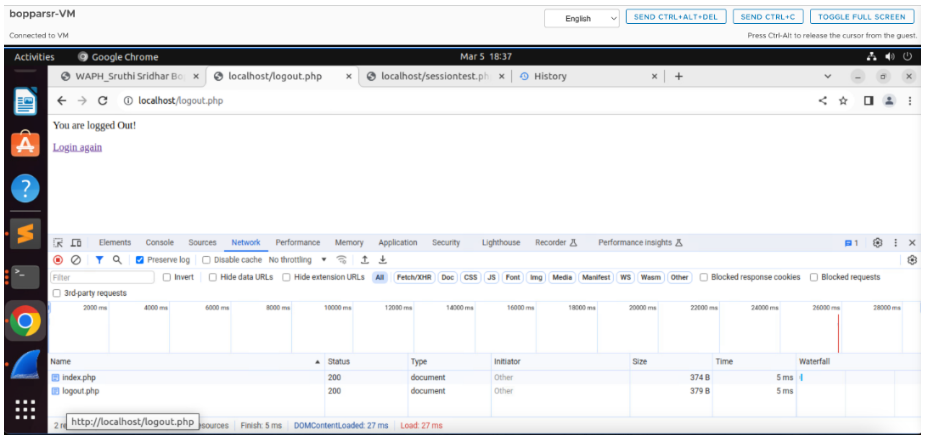
Logout

Name: index.php

▾Form Data   view source   view URL-encoded
username: admin
password: MyPa$$w0rd

1 requests | 374 B transferred | 60 B resources | Fin

Figure 14: Successfull Login with valid Credentials



You are logged Out!

Login again

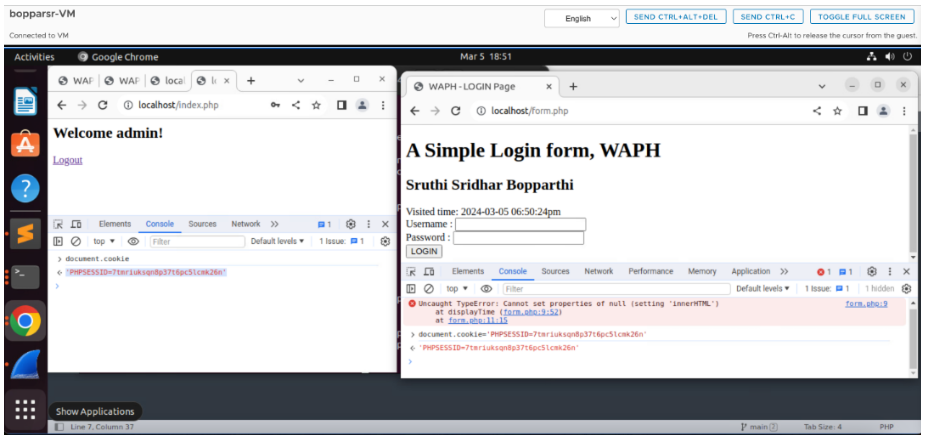
Name	Status	Type	Initiator	Size	Time
index.php	200	document	Other	374 B	5 ms
logout.php	200	document	Other	379 B	5 ms

```
Finish: 5 ms   DOMContentLoaded: 27 ms   Load: 27 ms
```

Figure 15: Logout and Session Destroyed

b) Session Hijacking Attacks

In the conducted session hijacking simulation, I logged in to the web application on one browser, manually copied the session ID, and pasted it into another browser. The process was documented with screenshots, illustrating the steps involved in the session hijacking attack. This hands-on exercise provided a practical understanding of the potential security risks associated with session management and emphasized the importance of implementing protective measures in web applications.



Welcome admin!

Logout

```
> document.cookie
< 'PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n'
```

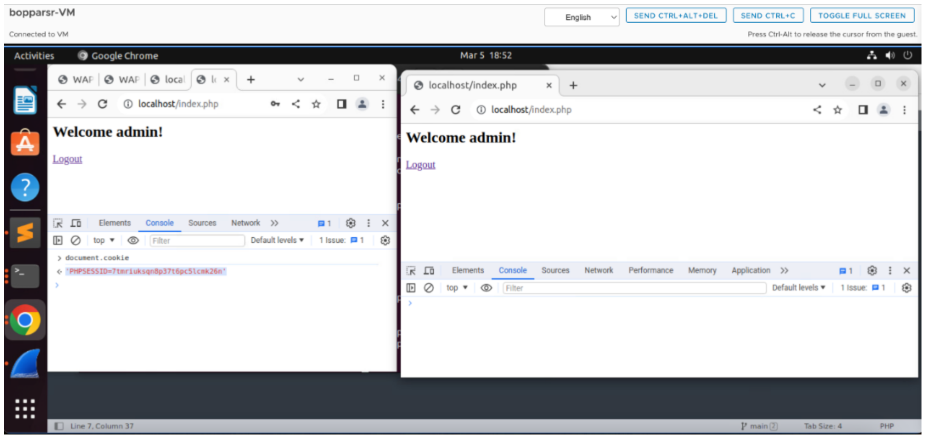
A Simple Login form, WAPH

Sruthi Sridhar Bopparthi

Visited time: 2024-03-05 06:50:24pm
Username :
Password :
LOGIN

```
Uncaught TypeError: Cannot set properties of null (setting 'innerHTML')    form.php:9
    at displayTime (form.php:9:52)
    at form.php:11:15
> document.cookie='PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n'
< 'PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n'
```

Figure 16: Session is copied to another browser



Welcome admin!

Logout

```
> document.cookie
<· 'PHPSESSID=7tmriuksqn8p37t6pc5lcmk26n'
```

Welcome admin!

Logout

Figure 17: Session has been hijacked

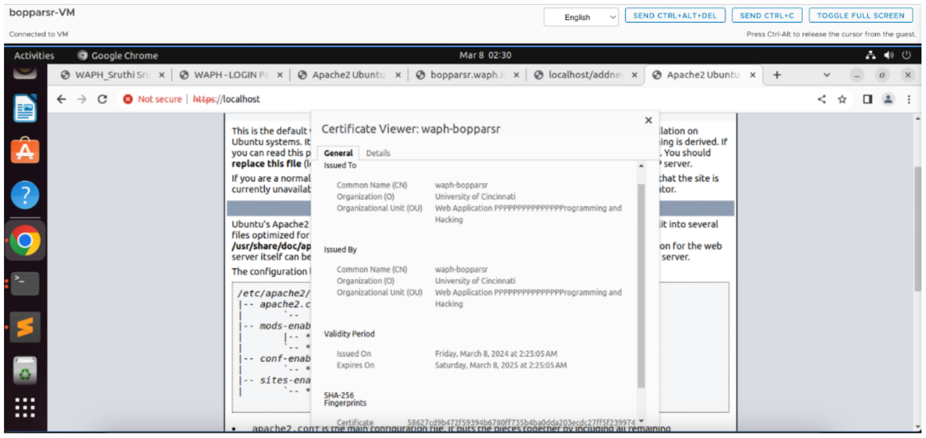
Task 3: Securing Session and Session Authentication

a) Data Protection and HTTPS Setup

I installed HTTPS on my web server, generated an SSL certificate, and viewed a PHP website via HTTPS. The screenshots show the details of the SSL certificate, which permits a secure connection between the client and the server. This guarantees data privacy by encrypting the information sent between the user's browser and the web server. An effective HTTPS configuration will improve the security of the web application.

Code:

```
openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout waph.key -out waph.crt
```



Certificate Viewer: waph-bopparsr

Issued To

Common Name (CN)	waph-bopparsr
Organization (O)	University of Cincinnati
Organizational Unit (OU)	Web Application PPPPPPPPPPPPProgramming and Hacking

Issued By

Common Name (CN)	waph-bopparsr
Organization (O)	University of Cincinnati
Organizational Unit (OU)	Web Application PPPPPPPPPPPPProgramming and Hacking

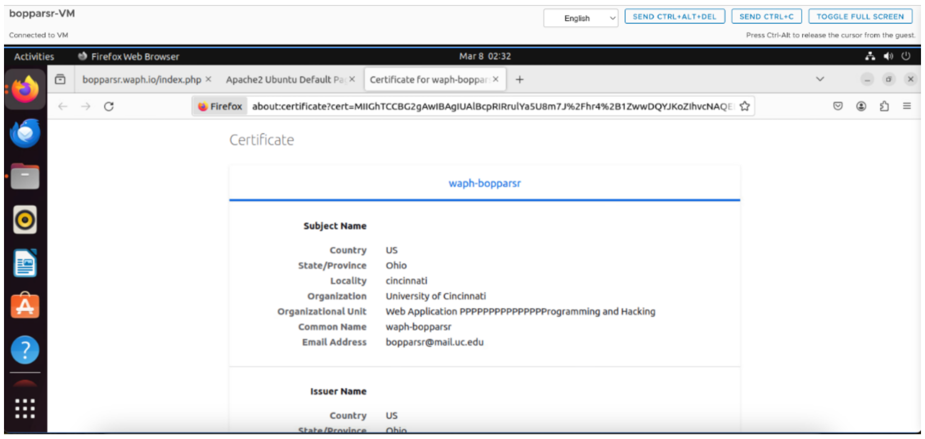
Validity Period

Issued On	Friday, March 8, 2024 at 2:25:05 AM
Expires On	Saturday, March 8, 2025 at 2:25:05 AM

SHA-256 Fingerprints

Certificate	58627cd9b472f59394b6780ff735b4ba0dda203ecdc27ff5f239974

Figure 18: HTTPS Chrome Certificate


bopparsr-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Firefox Web Browser
Mar 8 02:32
bopparsr.waph.io/index.php
Apache2 Ubuntu Default Pa
Certificate for waph-boppar
Firefox

```
about:certificate?cert=MIIGhTCCBG2gAwIBAgIUAlBcpRIRrulYa5U8m7J%2Fhr4%2B1ZwwDQYJKoZIhvcNAQE
```

Certificate

waph-bopparsr

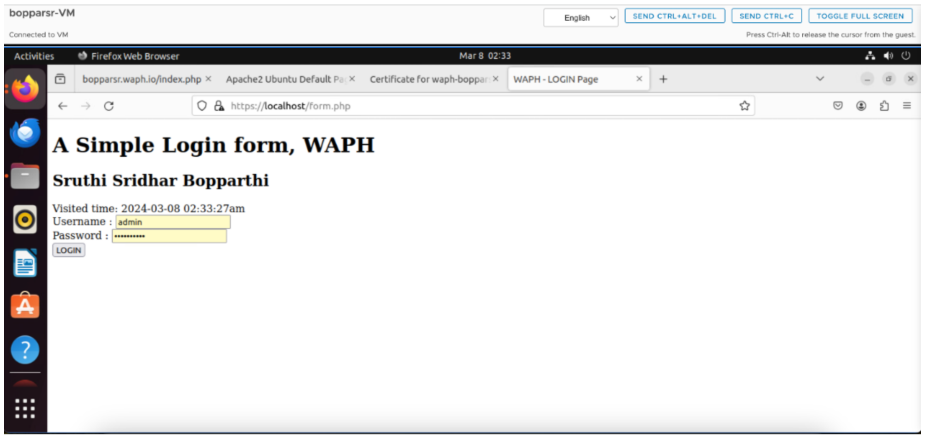
Subject Name

Country	US
State/Province	Ohio
Locality	cincinnati
Organization	University of Cincinnati
Organizational Unit	Web Application PPPPPPPPPPPPPProgramming and Hacking
Common Name	waph-bopparsr
Email Address	bopparsr@mail.uc.edu

Issuer Name

Country	US
State/Province	Ohio

Figure 19: HTTPS Firefox Certificate


bopparsr-VM
Connected to VM
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Press Ctrl-Alt to release the cursor from the guest.
Activities
Firefox Web Browser
Mar 8 02:33
bopparsr.waph.io/index.php
Apache2 Ubuntu Default Pa
Certificate for waph-boppar
WAPH - LOGIN Page
https://localhost/form.php

A Simple Login form, WAPH

Sruthi Sridhar Bopparthi

Visited time: 2024-03-08 02:33:27am
Username : admin
Password : ••••••••••
LOGIN

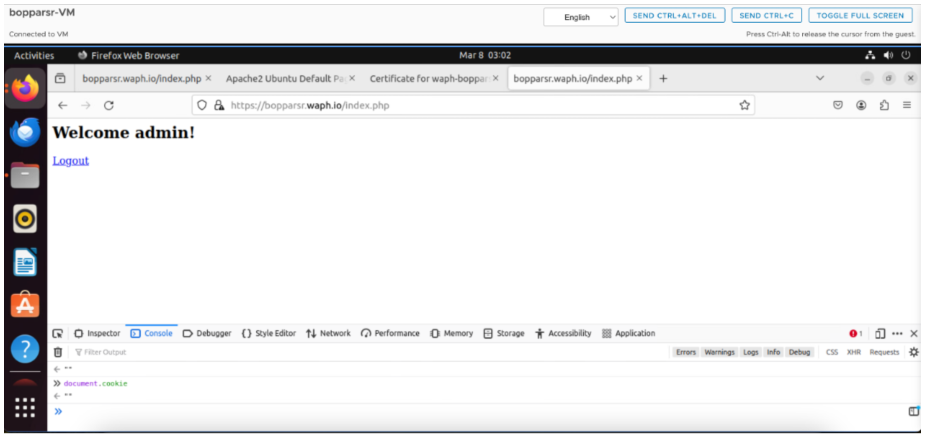
Figure 20: HTTPS Login Form

b) Securing Session Against Session Hijacking Attacks - setting HttpOnly and Secure flags for cookies

To bolster the security of our sessions, we implemented crucial measures by setting the HttpOnly and Secure flags for session cookies. This step ensures that session cookies are marked as HttpOnly, preventing client-side scripts from accessing them, thus mitigating the risk of cross-site scripting (XSS) attacks. Additionally, the Secure flag was applied to indicate that the cookies should only be transmitted over secure, encrypted connections, providing an extra layer of protection against unauthorized interception.

Code:

```
<?php
        $lifetime = 15 * 60;
        $path = "/";
        $domain = "bopparsr.waph.io";
        $secure = TRUE;
        $httponly = TRUE;
        session_set_cookie_params
($lifetime,$path,$domain,$secure,$httponly);
```

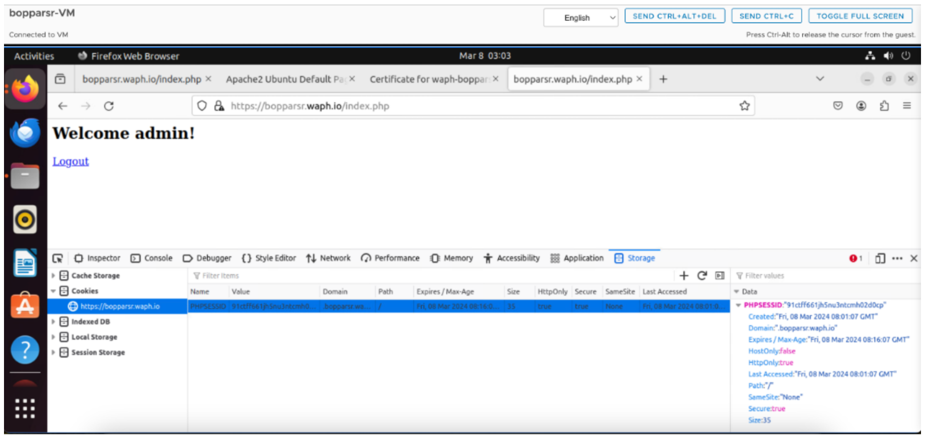


Welcome admin!

Logout

```
← ""
>> document.cookie
← ""
>>
```

Figure 21: Cookies are empty



Welcome admin!

Logout

```
PHPSESSID:"91ctff661jh5nu3ntcmh02d0cp"
    Created:"Fri, 08 Mar 2024 08:01:07 GMT"
    Domain:".bopparsr.waph.io"
    Expires / Max-Age:"Fri, 08 Mar 2024 08:16:07 GMT"
    HostOnly:false
    HttpOnly:true
    Last Accessed:"Fri, 08 Mar 2024 08:01:07 GMT"
    Path:"/"
    SameSite:"None"
    Secure:true
    Size:35
```

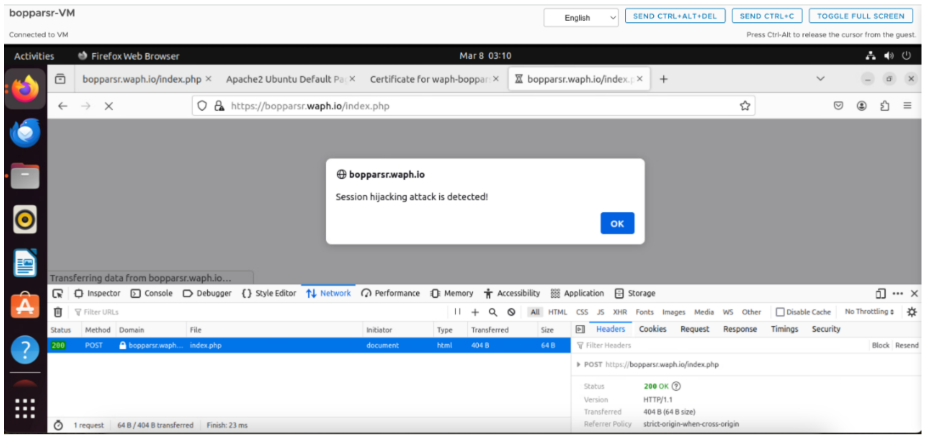
Figure 22: HttpOnly is set to true ennsuring no hijacking happens

c) Securing Session Against Session Hijacking Attacks - Defense In-Depth

In enhancing the security of our web application, I revised the index.php file to incorporate a new session variable that stores browser information following successful authentication. This additional layer of defense involves cross-verifying the information obtained from the user's browser with the stored session data. If discrepancies arise during this validation, indicating a potential session hijack, an alert is triggered, and the user is promptly redirected to the login page. This defense-in-depth strategy adds a proactive element to our session management, allowing us to detect and respond to any suspicious activities promptly.

Code:

```
if($_SESSION["browser"] != $_SERVER["HTTP_USER_AGENT"])
{
    session_destroy();
    echo "<script>alert('Session hijacking attack is detected!');</script>";
    header("Refresh:0; url=from.php");
    die();
}
```



bopparsr.waph.io

Session hijacking attack is detected!

```
200  POST  bopparsr.waph...  index.php  document  html  404 B  64 B
POST https://bopparsr.waph.io/index.php
Status          200 OK
Version         HTTP/1.1
Transferred     404 B (64 B size)
Referrer Policy strict-origin-when-cross-origin
```

Figure 23: Detection of Hijacking