Findings for Capture the Flag (CTF) Exercise 2

Mohammed Sadiq Abuwala: 45921407
Team Number: 16
16 May 2020

Abstract

The capture the Flag (CTF) exercise is a security hacking game organized for COMP 6320 (Offensive Security). For the purpose of this particular exercise, we targeted an application with an open HTTP port and we describe our experience and results here in detail. Every flag was concealed in such a way that it required some sort of operations before they could be revealed. Instead of dividing flags between team members, our team collectively decided to approach the exercise on a best effort basis.

Introduction

Hardware and Software used during the exercise:

Hardware:
Processor: Intel ® Core ™ i7-8550U CPU @ 1.80GHz
Installed memory (RAM): 8.00 GB
Primary Storage: 512GB SSD M.2 2280 PCIe NVMe
Note: Hardware specifications above may not be relevant as we are using a Virtual Machine (VM) as noted below.

Software:
Oracle virtual Machine running Kali Linux with following configuration:
System Base Memory (RAM): 2048MB
Processors: 2

In total, this exercise consisted of capturing six flags over a time duration of two hours. The **connection and target details** are provided below:

1. VPN Access via the Virtual Intranet Access application.
2. Moby dashboard to enter flags and view scoreboard.
3. Target IP address.

Now, let us probe the given target IP address further to find further network properties like open ports by using the network scanning utility **nmap**. This confirms the above data that the HTTP port is open. Hence, we can now try connecting to the address by inputting it in the browser. Flags have been recorded below in the order that they were discovered by the individual.

Flag 1 - Indices

Found by team member - Mohammed Sadiq Abuwala

Now that I have used Nmap to find an open port and the page loads on the browser, I have probed further using the following methods:

1. Gobuster to check for available directories.
2. Check the page source to look for anything of particular interest.
3. Check page header using Burp Suite

While investigating the header using burp suite, I further came across a unique piece of information which is not present in any other page headers. This is shown in the next page:

If-Modified-Since: Mon, 04 May 2020 15:07:25 GMTIf-None-Match: "99b-5a4d3e3798540-gzip"

After attempting to modify and remove above data, the flag was revealed in the page source. A screenshot of the workings has been provided below:

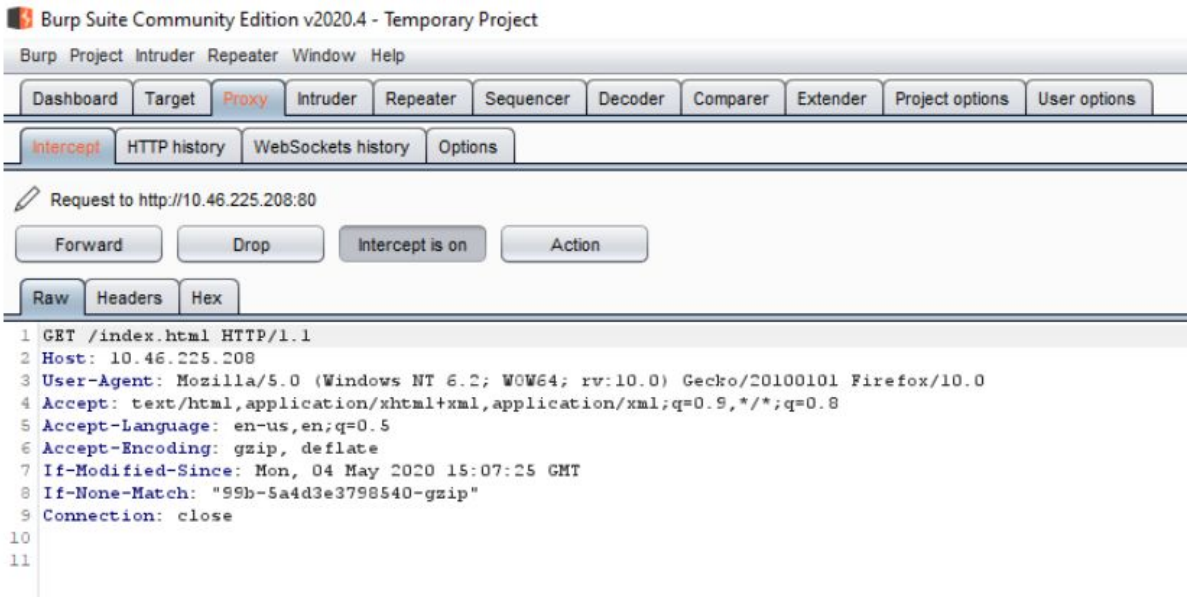

Figure 1: Screenshot for Flag 1

Flag 2 - Busted

Partially solved by team member - Stanley Epuna

White checking for directories using gobuster in flag 1, we had come across the robots.txt file which seems to be of interest here. Going to the directory tells us that /json is disallowed. It could be theorized that the user agent needs to change or permissions need to be gained to access this particular directory but any further attempts were inconclusive.

Flag 3 - Consolation Prize

Found by team member - Divya Kaur

An input box to enter student ID has been provided. The availability of an input box provides us with the opportunity to attempt SQL injection using the following commands:

```
') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
" or ""-"
" or "" "
' or ''-'
' or '' '
```

Attempts were unsuccessful and did not yield any outcomes. Further probing was done by checking the page header using Burp Suite. Ultimately, the solution was much more ingenious as inputting a student ID and checking page source reveals the flag. A screenshot of this findings is shown in the next page:

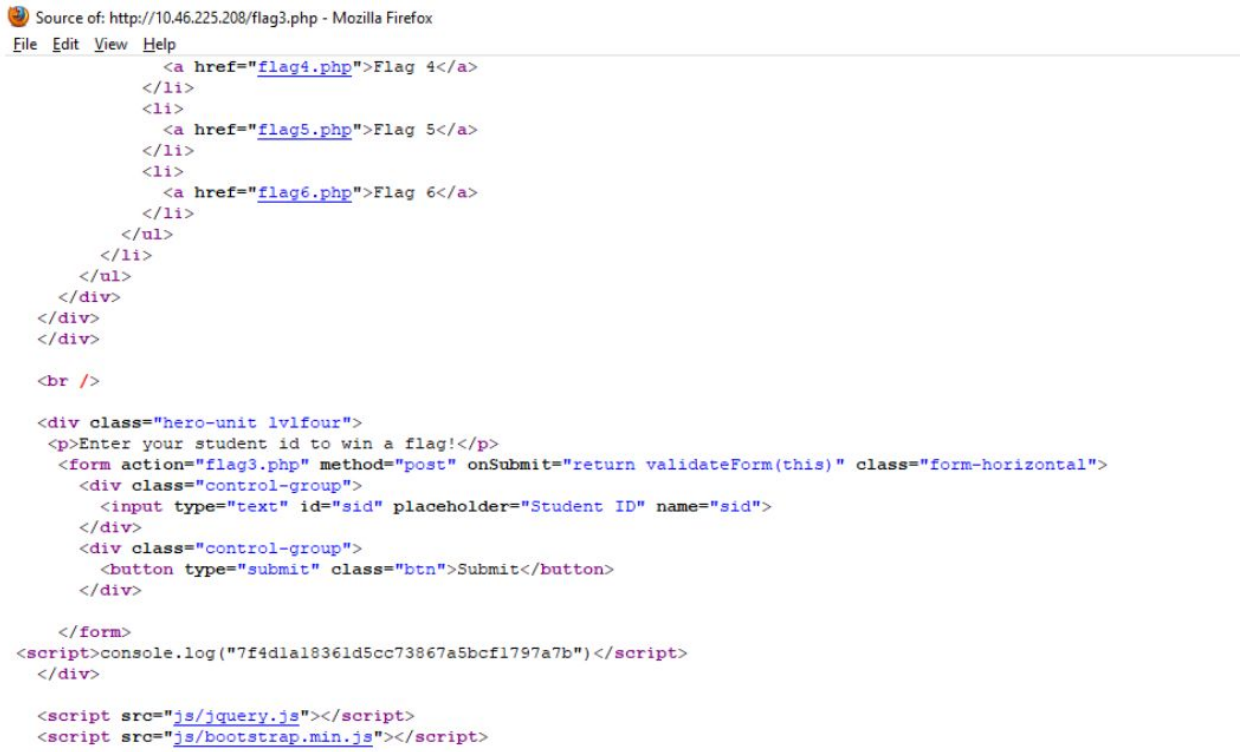

Figure 2: Screenshot for Flag 3

Flag 4 - Secret Agent

Partially solved by team member - Mohammed Sadiq Abuwala

On attempting to open the page, a dialog box pops up informing the user that they must be using the Mosaic browser. No other input or operation on the page is possible. Hence, checking page source is not an option. Other probing methods were attempted - capture header using Burp Suite. We can see the user agent in the page header. We try to change it to the user agent of the Mosaic 3.0 browser :

NCSA **Mosaic/3.0** (Windows 95)

Above operation was successful as changing the user agent and forwarding the header causes the page to load. Now, I came across some input boxes for username and password with the only identifying piece of information being “Textportal 0.8”. Again, I checked the page source for any interesting pieces of information without success. SQL injection methods were then attempted with the following commands:

') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
" or ""-"
" or "" "
' or ''-'
' or '' '

All attempts at SQL injection were unsuccessful. On further research on Textportal 0.8 vulnerabilities, I came across default credentials:

Username: god2
Password: 12345

On inputting the above credentials,a dialog box pops up with the following code:

}1e3f82955284124205089decab36fe66{4galf

On attempting to enter the above code on the Moby dashboard, it was rejected. Any further attempts on trying to solve this flag were unsuccessful.

Flag 6 - Do you like sequels?

Partially solved by team member - Mohammed Sadiq Abuwala

Input boxes for username and password have been provided on the page. We check page source and page header for any interesting pieces of information. Furthermore, I have attempted sql injection techniques as shown below:

```
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
junk OR 1=1 #                // Login successful
```

While login is successful with the above SQL injection technique, no flag can be found on the page or the page source. Any further attempts to solve this flag were unsuccessful.

Summary of CTF Results

FLAG NUMBER	RESULTS	SOLVED BY / ATTEMPTED BY
FLAG 1	SOLVED	MOHAMMED SADIQ ABUWALA
FLAG 2	PARTIALLY SOLVED	STANLEY EPUNA
FLAG 3	SOLVED	DIVYA KAUR
FLAG 4	PARTIALLY SOLVED	MOHAMMED SADIQ ABUWALA
FLAG 5	UNSOLVED	UNSOLVED
FLAG 6	PARTIALLY SOLVED	MOHAMMED SADIQ ABUWALA

Conclusion

In the first CTF, we conducted penetration tests on a linux server. For this exercise, we were able to broaden our horizons further by learning to conduct penetration tests on open HTTP ports. As we have seen from this exercise, The availability of further open ports has provided us with additional techniques that we can use while conducting penetration testing such as SQL injection and page header modification. Furthermore, we could also reflect that completing NATAS and Hack the Box exercises have helped us immensely to prepare for this CTF. These exercises have induced our interest in learning. We would consider this CTF exercise as an interactive and interesting tool in enabling us to further our knowledge in testing systems. Additionally, this exercise helped us understand the concepts at a more practical level.

Disclaimer

It is important to note that this CTF was conducted in a sandboxed environment solely for educational purposes. All team members have signed an Acceptable Usage agreement and understand the responsibilities and consequences while conducting penetration tests.