Blunder walkthrough

Index

List of pictures

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Reconnaissance

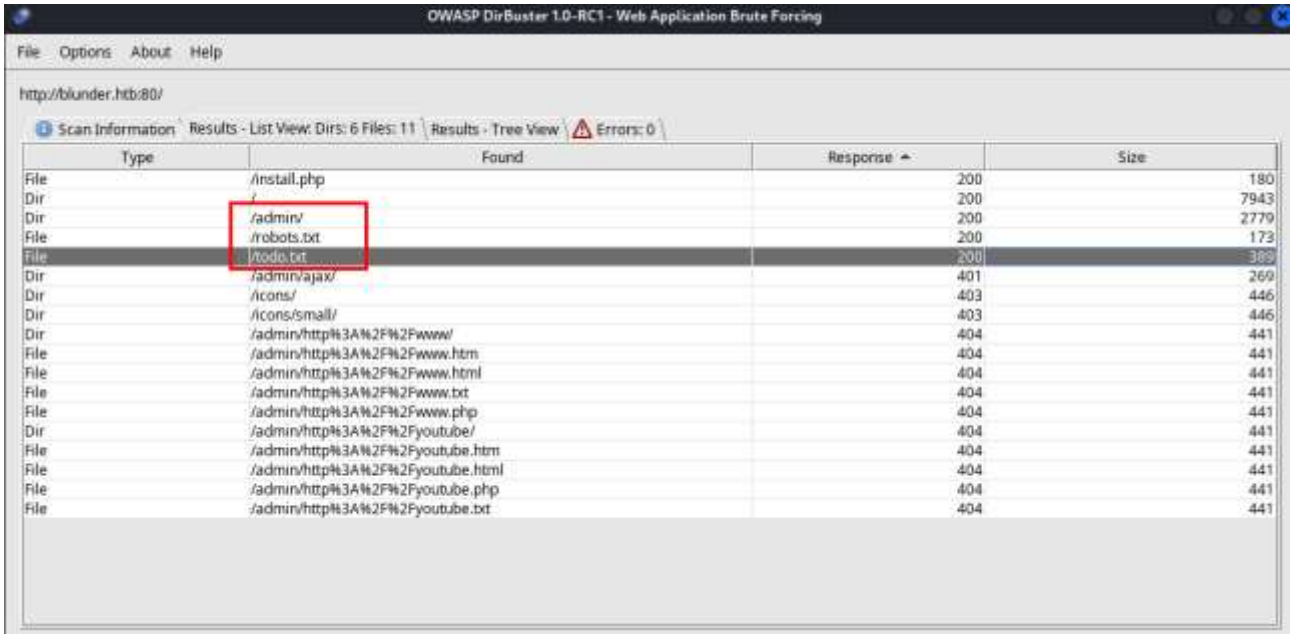
The results of an initial nMap scan are the following:

Figure 1 - nMap scna results

Open ports are 21 and 80. So, this box has FTP service enabled and a web application running on the port 80. Also, nMap guesses the OS as Linux 5.0.

Initial foothold

Since I have a web application, one of the first thing I do is running Dirbuster to find some hidden contents. In this case, I found the following:



Type	Found	Response	Size
File	/install.php	200	180
Dir	/	200	7943
Dir	/admin/	200	2779
File	/robots.txt	200	173
File	/todo.txt	200	389
Dir	/admin/ajax/	401	269
Dir	/icons/	403	446
Dir	/icons/small/	403	446
Dir	/admin/http%3A%2F%2Fwww/	404	441
File	/admin/http%3A%2F%2Fwww.htm	404	441
File	/admin/http%3A%2F%2Fwww.html	404	441
File	/admin/http%3A%2F%2Fwww.txt	404	441
File	/admin/http%3A%2F%2Fwww.php	404	441
Dir	/admin/http%3A%2F%2Fyoutube/	404	441
File	/admin/http%3A%2F%2Fyoutube.htm	404	441
File	/admin/http%3A%2F%2Fyoutube.html	404	441
File	/admin/http%3A%2F%2Fyoutube.php	404	441
File	/admin/http%3A%2F%2Fyoutube.txt	404	441

Figure 2 - Contents found running Dirbuster

In particular, the **todo.txt** file contains an possible username, as shown in the following picture:

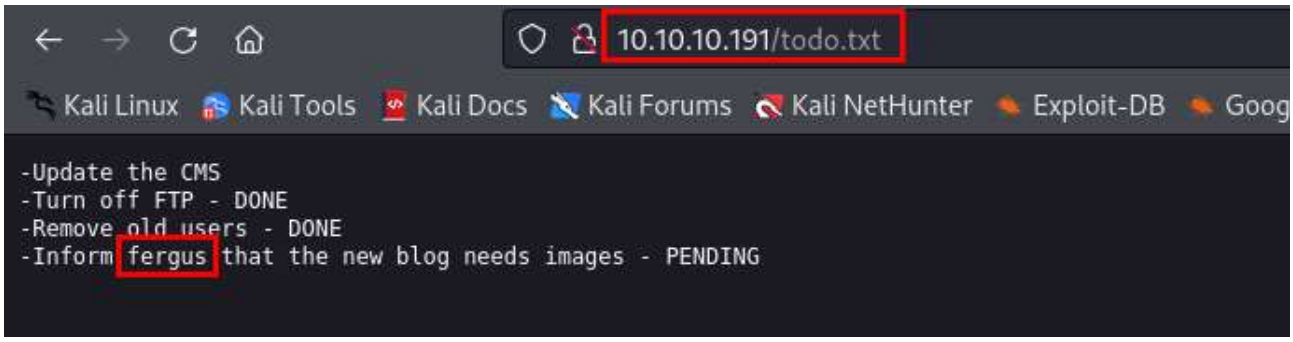


Figure 3 - Possible username found

Also, on the **/admin** page there is a login. Since I have a possible username and a login form, I tried to brute force the password. However, any wordlist I used (from seclist or application like Dirb and Dirbusetr) it din't work. It was very frustrating. At this point I tried to analyze again the application and I though that a password could be something wrote in some page. So, I extract all strings from page using the following command:

$$cewl\ 10.10.10.191\ >\ wordlist.txt$$

At this point I can try to exploit **CVE-2019-17240** e la **CVE-2019-16113** running the file **poc.py**. I set up the exploit as shown in the following figure:

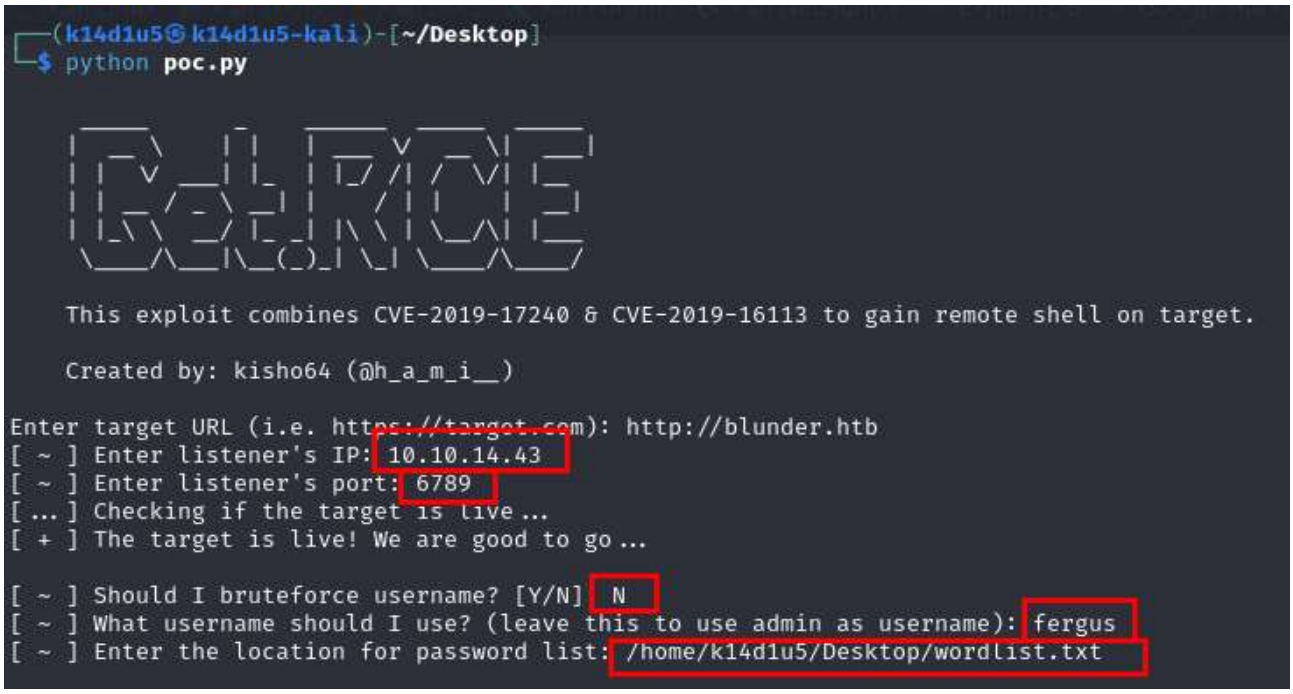


Figure 4 - Exploit configuration

In this way I was able to find credentials and obtain a user shell:

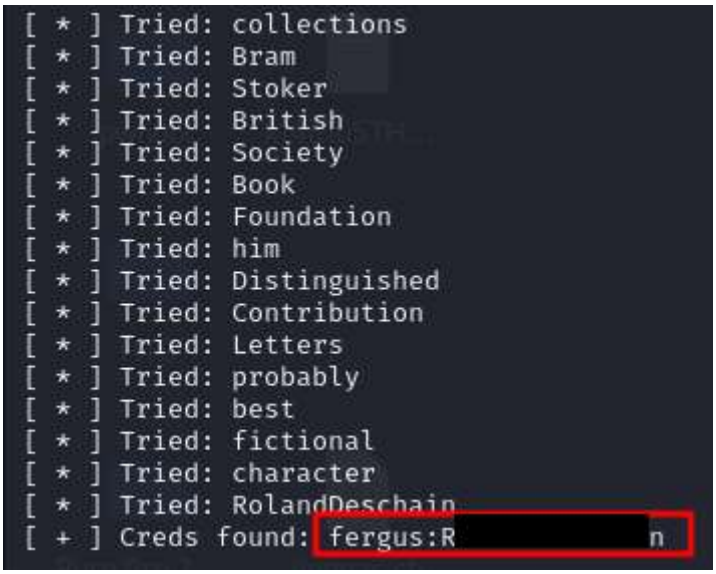


Figure 5 - Credentials found

However, the user I have in this way didn't help me to have the user flag.

User flag

I needed to perform a lateral movement to became **hugo** user. Analyzing the file system, I found two version of the application code. In particular, version 3.10 (the newer, I remembered that in todo.txt file someone wrote that old user was deleted) allowed me to find new credentials:

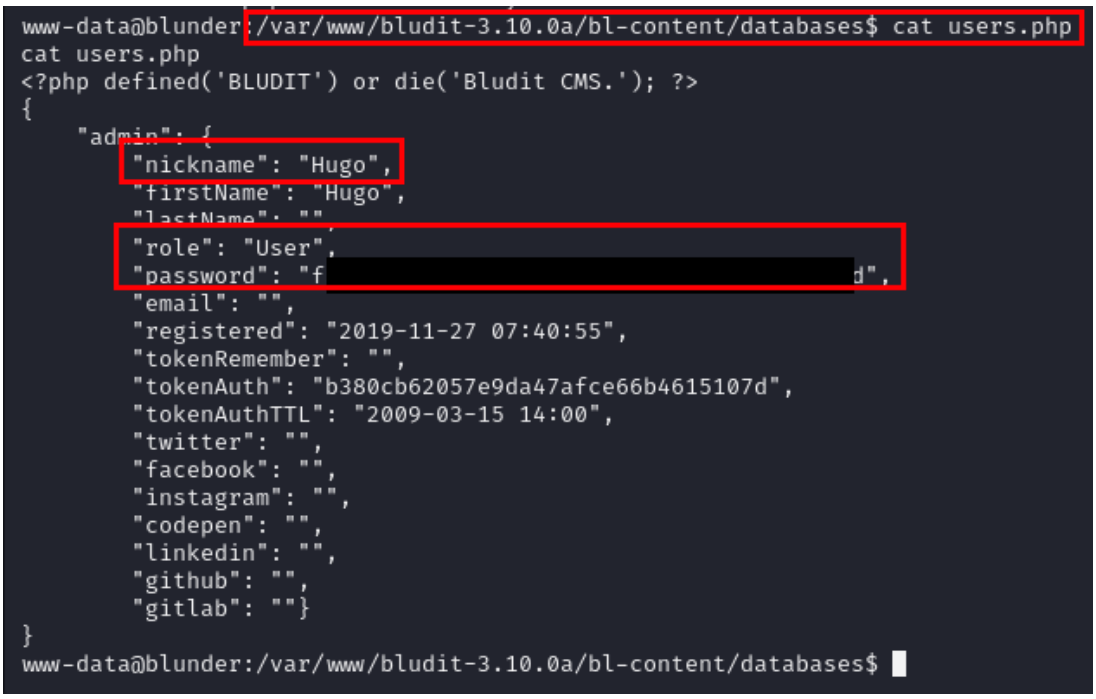


Figure 6 - New credentials found

The password found here is a hashed one. So, I used **CrackStation** to try to decrypt it:

Figure 7 - Password cracked

At this point, I can become hugo simply running the $su\ hugo$ command and I can retrieve the user flag:

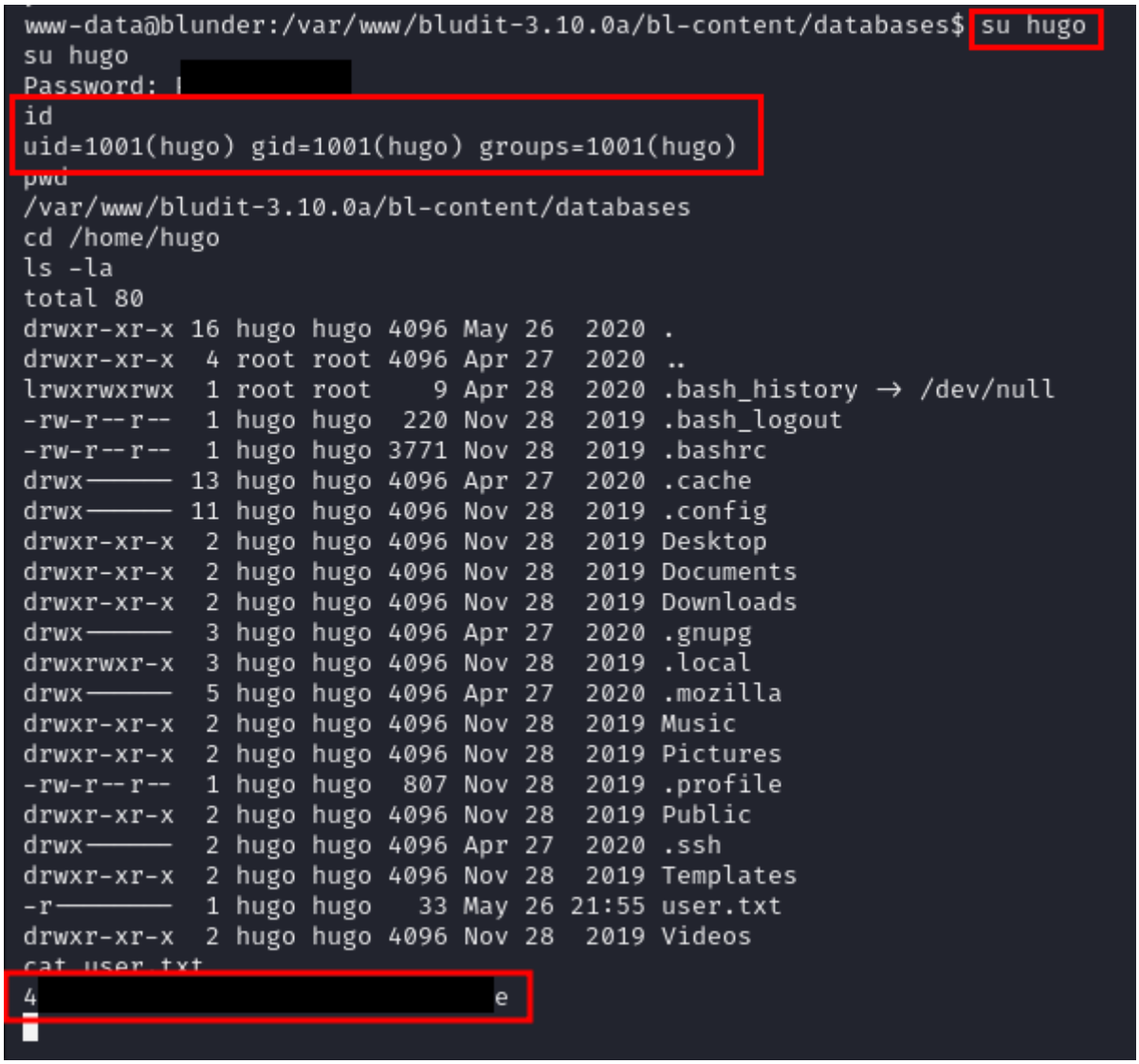


Figure 8 - User flag

Privilege escalation

The privilege escalation is performed noting (using linpeas for example) that the sudo command is vulnerable to **CVE-2019-14287**. So, all I need to became root is running the command

$$sudo\ -u\#-1\ /bin/bash$$

and retrieve the root flag:

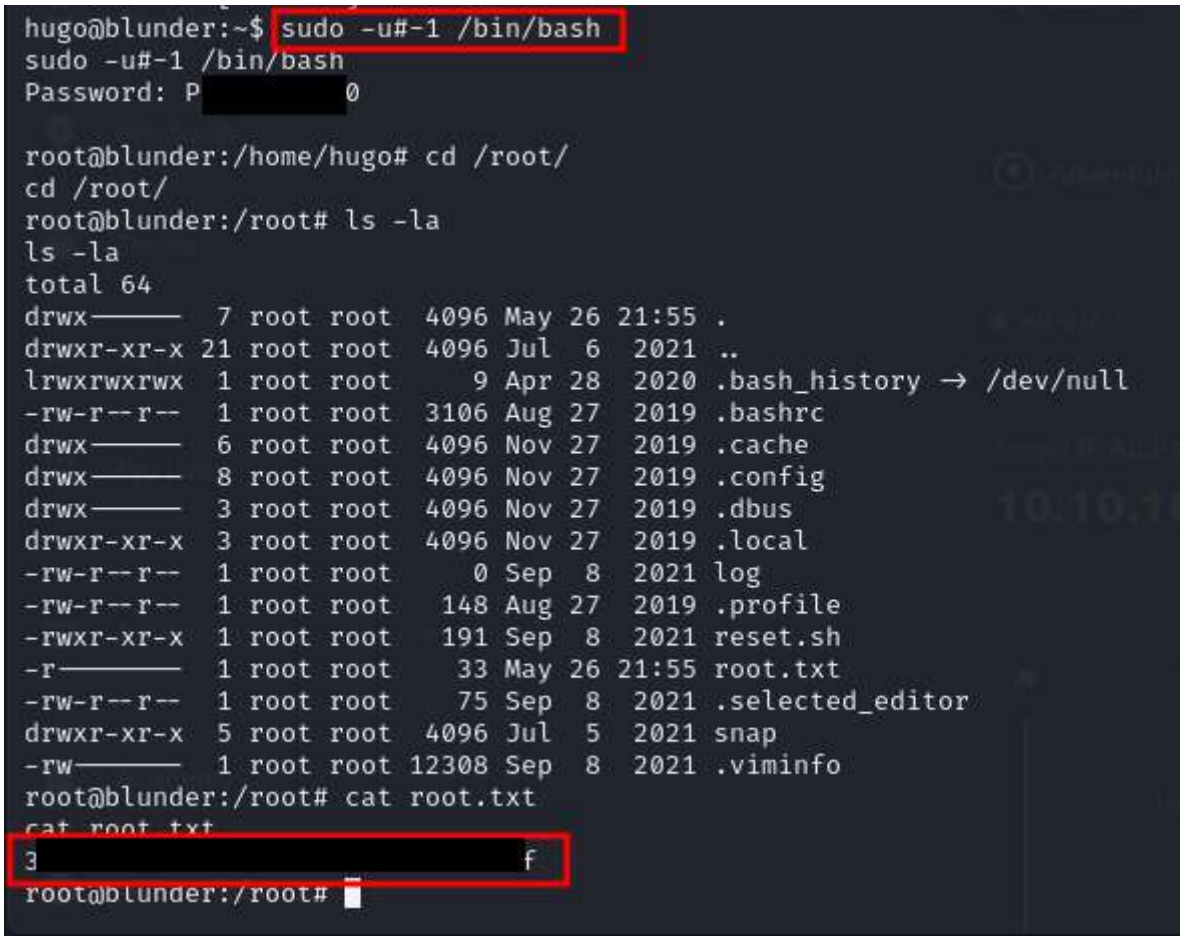


Figure 9 - Privesc and root flag

APPENDIX A - CVEs

CVE-2019-17240

Versions prior to and including 3.9.2 of the Bludit CMS are vulnerable to a bypass of the anti-brute force mechanism that is in place to block users that have attempted to incorrectly login 10 times or more. Within the **bl-kernel/security.class.php** file, there is a function named **getUserIp** which attempts to determine the true IP address of the end user by trusting the X-Forwarded-For and Client-IP HTTP headers:

```
public function getUserIp()
{
  if (getenv('HTTP_X_FORWARDED_FOR')) {
    $ip = getenv('HTTP_X_FORWARDED_FOR');
  } elseif (getenv('HTTP_CLIENT_IP')) {
    $ip = getenv('HTTP_CLIENT_IP');
  } else {
    $ip = getenv('REMOTE_ADDR');
  }
  return $ip;
}
```

Figure 10 - getUserIp function

The reasoning behind the checking of these headers is to determine the IP address of end users who are accessing the website behind a proxy, however, trusting these headers allows an attacker to easily spoof the source address. Additionally, no validation is carried out to ensure they are valid IP addresses, meaning that an attacker can use any arbitrary value and not risk being locked out. As can be seen in the content of the log file below (found in **bl-content/databases/security.php**), submitting a login request with an **X-Forwarded-For** header value of **FakeIp** was processed successfully, and the failed login attempt was logged against the spoofed string:


```
{
    "minutesBlocked": 5,
    "numberFailuresAllowed": 10,
    "blackList": {
        "192.168.194.1": {
            "lastFailure": 1570286876,
            "numberFailures": 1
        },
        "10.10.10.10": {
            "lastFailure": 1570286993,
            "numberFailures": 1
        },
        "FakeIp": {
            "lastFailure": 1570287052,
            "numberFailures": 1
        }
    }
}
```


Figure 11 - FakeIp processed

By automating the generation of unique header values, prolonged brute force attacks can be carried out without risk of being blocked after 10 failed attempts.

CVE-2019-16113

The CVE-2019-16113 affects some unknown processing of the file ***bl-kernel/ajax/upload-images.php***. The manipulation as part of a ***File Name*** leads to a code injection vulnerability. The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment. This is going to have an impact on confidentiality, integrity, and availability.

CVE-2019-14287

The security policy bypass vulnerability that allows users on a Linux system to execute commands as root, while the user permissions in the **sudoers** file explicitly prevents these commands from being run as root. It can be executed by a user that has **ALL** permissions in the **Runas** specification. Which means they can execute commands as any or all users on the system. This consequently allows users to run commands and tools as root by specifying the user id (UID) as **-1** or the unsigned equivalent of **-1**: **4294967295**.