端口扫描技术

介绍

对于主机存活与端口探测，我们的目的是发现更多的存活主机，探测这些主机运行了哪些服务，以扩大我们的攻击面。

主机存活探测

存活探测通常有以下几种技术：

- ARP扫描
- ICMP扫描
- TCP和UDP扫描

ARP扫描：

```
sudo nmap -sn 10.10.210.6/24
```

ICMP扫描：

ICMP其实有几种请求类型，比如ping命令是ICMP 8/Echo 类型，nmap相关的选项有-PE、-PP、-PM

```
sudo nmap -PM -sn 10.10.68.220/24
```

TCP和UDP扫描：

```
sudo nmap -PS80,443,8080 -sn 10.10.68.0/24
sudo nmap -PA -sn 10.10.68.0/24
sudo nmap -PU -sn 10.10.68.220/24
```

端口扫描探测

开放端口探测

Nmap扫描端口有以下几种状态

Open: 表示在指定端口上有一个服务在监听。

Closed: 表示在指定端口上没有服务在监听，但该端口是可访问的。可访问的意思是该端口可以到达，没有被防火墙或其他安全设备/程序阻挡。

Filtered: 表示Nmap无法确定该端口是开放还是关闭，因为该端口不可访问。这种状态通常是由于防火墙阻止Nmap访问该端口。Nmap的包可能被阻止到达该端口，或者响应被阻止到达Nmap的主机。

Unfiltered: 表示Nmap无法确定该端口是开放还是关闭，但该端口是可访问的。使用ACK扫描（-sA）时会遇到这种状态。

Open|Filtered: 表示Nmap无法确定该端口是开放还是已过滤。

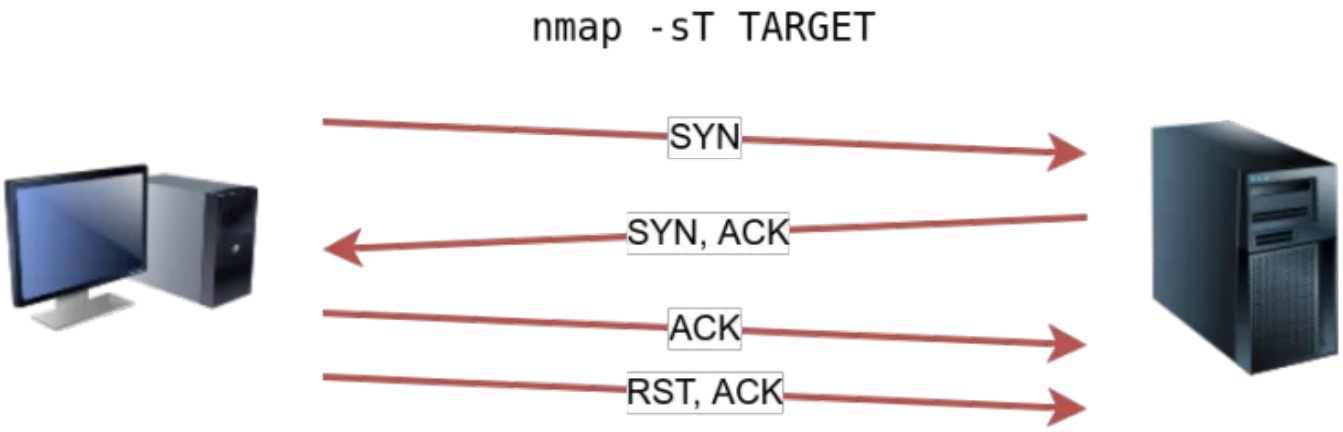
Closed|Filtered: 表示Nmap无法确定该端口是关闭还是已过滤。

TCP扫描

```
Plain Text
1   sudo nmap -sT --min-rate 10000 -p- 172.16.200.131
```

TCP连接扫描

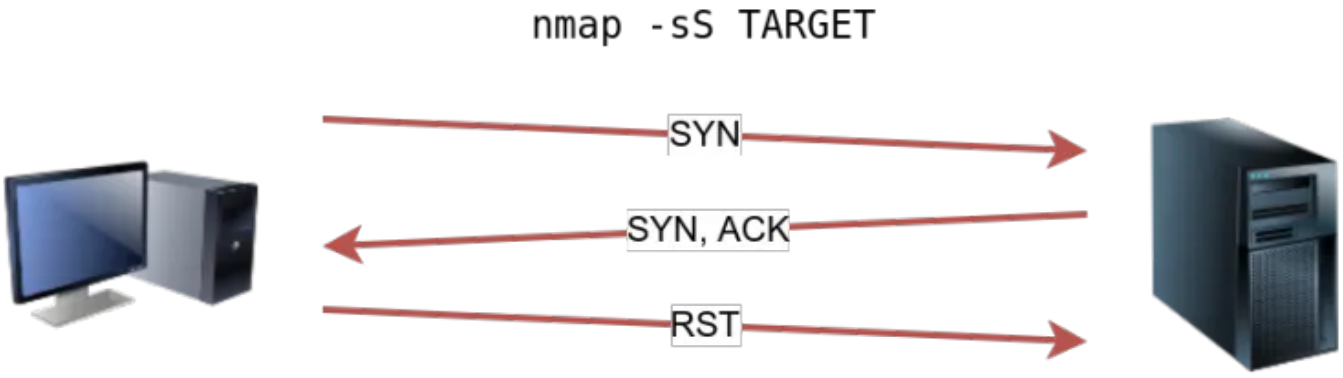
通过-sT选项指定TCP连接扫描，TCP连接扫描会完成TCP三次握手。



Case: TCP port is open.

TCP SYN扫描

通过-sS指定TCP SYN扫描，TCP SYN扫描不需要完成TCP三次握手，一旦接收到服务器的响应，就会断开连接。使用-sS选项需要root权限。



Case: TCP port is open.

UDP

UDP扫描的准确性、速率、攻击面，相对于TCP都要差很多，我们无法保证在 UDP 端口上侦听的服务会响应我们的数据包，扫描速度也要慢很多，我们通常只会扫描最常见的1000或100（-F选项）个端口。

```
Plain Text
1  sudo nmap -sU --min-rate 10000 -p- 172.16.200.131
```

探测响应	Assigned State 指定状态
来自目标端口的任何 UDP 响应（不寻常）	open
未收到响应（即使在重发后）	open|filtered
ICMP 端口不可达错误（类型 3，代码 3）	closed
其他 ICMP 不可达错误（类型 3，代码 1、2、9、10 或 13）	filtered

端口信息获取

获取版本和系统信息。

```
Plain Text
1  sudo nmap -sT -sV -O -p80,111,777,52497,5353,20444 172.16.200.13
```

Nmap默认脚本扫描

使用nmap默认脚本扫描指定端口。

```
Plain Text
1  sudo nmap -sC -p80,111,777 172.16.200.13
```

漏洞脚本扫描。

```
sudo nmap --script=vuln -p80,111,777 172.16.200.13
```

Nmap常用命令

```bash
sudo nmap -sn 172.16.200.0/24
sudo nmap -sS  --min-rate=10000 -p- --defeat-rst-ratelimit -oN ports.nmap -v 10.10.183.12
sudo nmap -sU 10.10.207.202 -F -v
sudo nmap -sU 10.10.207.202 -p- -v
sudo nmap -sT -sV -O -p80,111,777,52497,5353,20444 172.16.200.13 -oN service.nmap
sudo nmap -sT --script=vuln -p80
```

非Nmap下的存活探测

ping

```
for i in {1..254};do ping -c 1 -W 1 172.16.200.$i | grep from ;done
```

> 碰到ctrl c结束不了的情况，需要按ctrl z
>
> 然后 kill -9 %1

```
┌──(kali㉿kali)-[~/vulhub/w1r3s]
└─$ for i in {1..254};do ping -c 1 -W 1 192.168.134.$i | grep from ;done
64 bytes from 192.168.134.2: icmp_seq=1 ttl=128 time=0.893 ms

64 bytes from 192.168.134.156: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 192.168.134.157: icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from 192.168.134.166: icmp_seq=1 ttl=128 time=1.39 ms
```

端口扫描

```
export ip=yourip; for port in $(seq 1 65535); do timeout 0.01 bash -c "</dev/tcp/$ip/$port && echo The port $port is open || echo The Port $port is closed > /dev/null" 2>/dev/null || echo Connection Timeout > /dev/null; done
```

这段Bash脚本用于扫描指定IP地址的所有TCP端口（从1到65535），检查哪些端口是打开的，并过滤掉关闭或连接超时的端口。

```
export ip=yourip;
```

这行代码将目标IP地址赋值给变量**ip**。将**yourip**替换为你要扫描的目标IP地址。

```
for port in $(seq 1 65535);
```

这行代码使用一个**for**循环，遍历从1到65535的所有端口号。**$(seq 1 65535)**生成一个从1到65535的序列，用于端口扫描。

```
do timeout 0.01 bash -c "</dev/tcp/$ip/$port && echo The port $port is open || echo The Port $port is closed > /dev/null" 2>/dev/null || echo Connection Timeout > /dev/null;
```

这个循环体的每一步操作如下：

1. **timeout 0.01 bash -c "..."**：timeout命令限制命令的执行时间为0.01秒。如果命令在规定时间内未完成，将会被终止。
2. **</dev/tcp/$ip/$port**：尝试连接到指定IP地址和端口。如果连接成功，表示端口是打开的；否则表示端口是关闭或超时。
3. **&& echo The port $port is open**：如果连接成功，输出"端口 $port 是打开的"。
4. **|| echo The Port $port is closed > /dev/null**：如果连接失败，输出"端口 $port 是关闭的"到空设备，这实际上丢弃了这个消息。
5. **2>/dev/null**：将标准错误输出重定向到空设备，丢弃所有错误消息。
6. **|| echo Connection Timeout > /dev/null**：如果timeout命令因为超时而失败，输出"连接超时"到空设备，这实际上也丢弃了这个消息。

```
done
```

结束for循环。

拓展

Nmap备忘录：https://www.stationx.net/nmap-cheat-sheet/