链滴

端口扫描

作者：someone38063
原文链接：https://ld246.com/article/1558793238077
来源网站：链滴

端口扫描

- 端口对应网络服务及应用端程序
- 服务端程序的漏洞通过端口攻入
- 发现开放的端口
- 更具体的攻击面

UDP端口扫描

- 假设ICMP port-unreachable响应，代表端口关闭
- 目标系统不响应ICMP port-unreachable时，可能产生误判
- 完整的UDP应用层请求
- 准确性高
- 耗时巨大

Scapy UDP Scan

- 端口关闭：ICMP port-unreachable
- 端口开放：没有回包
- 了解每一种基于UDP的应用层包结构很有帮助
- 与三层相同的技术
- 误判
- Scapy
- sr1(IP(dst="1.1.1.1")/UDP(dport=53),timeout=1,verbose=1)

./udp_scan.py 1.1.1.1 1 100

```
#!/usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import*
import time
```

原文链接：端口扫描

```
import sys
if len(sys.argv) != 4:
print"Usage - ./udp_scan.py [Target-IP] [FIRSR PORT] [LAST PORT]"
print"Example - ./udp_scan.py 192.168.1.1 1 254"
print"Example will UDP port scan ports 1 through 254 on 192.168.1.1"

ip=sys.argv[1]
start=int(sys.argv[2])
end=int(sys.argv[3])

for port in range(start,end):
    a=sr1(IP(dst=ip)/UDP(dport=port),timeout=5,verbose=0)
    time.sleep(1)
    if a ==None:
     print port
    else:
     pass
```

Nmap

- nmap -sU 1.1.1.1 默认扫描常用的1000个端口 （s：scanner U：UDP）
- 默认的1000个参数
- ICMP host-unreachable
- nmap 1.1.1.1 -sU -p 53 （p：port）
- nmap -iL iplist.txt -sU -p 1-200

TCP端口扫描

- 基于连接的协议
- 三次握手

- 隐蔽扫描（只发syn包，目标端口开放回ack，否则回rst ）
- 僵尸扫描
- 全连接扫描（建立完整的三次握手）
- 所有的TCP扫描方式都是基于三次握手的变化，来判断目标端口状态

①• --->(syn)•

②•<---(syn,ack)•

③• --->(ack)•

端口扫描

• 隐蔽扫描——syn
  - 不建立完整链接
  - 应用层日志不记录扫描行为——隐蔽
• 僵尸扫描
  - 极其隐蔽
  - 实施条件苛刻（发起者可以伪造IP地址的网络环境）
  - 可伪造源地址
  - 可选择僵尸机
    - 闲置系统（基本不与服务器通信）
    - 系统使用递增的IPID （ip包头中的id字段：Identification，不同操作系统id产生不同，若为增，每发一个ip包，id加一）
      - 0（×）
      - 随机（×）

①scanner向zombie发送一个ack/syn，zombie回复RST，ip包中id字段=x；

②scanner伪造源ip为zombie的ip，向target发送syn包，target向zombie回复syn/ack，zombie回target RST，ip包中id字段=x+1； （端口没开放则target回复zombie RST，zombie不做回复）

③scanner向zombie发送syn/ack，zombie回复RST，若ip包中id字段=x+2；说明target的目标端开（id=x+1则说明端口关闭）

隐蔽端口扫描

• syn——syn/ack——rst (给目标机器发送syn，目标返回syn/ack，再给目标机器返回rst，断开三握手)

- scapy
  - sr1(IP(dst="192.168.1.1")/TCP(dport=80),timeout=1,verbose=1)
  - ./syn_scan.py

- nmap -sS 192.168.1.1 -p 80,21,25,110,443 (s：scanner S：SYN；默认syn扫描)
- nmap -sS 192.168.1.1 -p 1-65536 --open （只显示open）
- nmap -sS 192.168.1.1 -p- --open (-p-等于-p 65535)
- nmap -sS -iL iplist.txt -p 80,21,22,23

隐蔽端口扫描

- hping3
- hping3 192.168.1.1 --scan 80 -S （S：SYN）
- hping3 192.168.1.1 --scan 80,21,25,443 -S
- hping3 192.168.1.1 --scan 0-65535 -S
- hping3 -c 10 -S --spoof 192.168.1.138 -p ++1 192.168.1.1 （spoof：欺骗，即伪造源ip；++1从端口1开始，每次加1）

全连接端口扫描

- Scapy
  - SYN不需要raw packets
  - 内核认为syn/ack是非法包，直接返回rst终端连接
  - 全连接扫描对scapy比较困难
  - sr1(IP(dst="192.168.20.2")/TCP(dport=22,flags='S'))
  - ./tcp_scan1.py
  - ./tcp_scan2.py
  - iptables -A OUTPUT -p tcp --tcp-flags RST RST -d 192.168.1.1 -j DROP (A：添加规则，p：议，d：目标地址，j：动作)

tcp_scan1.py

```
#!/usr/bin/python
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import*

response=sr1(IP(dst="192.168.1.1")/TCP(dport=80,flags='S'))
reply=sr1(IP(dst="102.158.1.1")/TCP(dport=80,flags='A',ack=(response[TCP].seq+1)))
```

tcp_scan2.py

```
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import*

SYN=IP(dst="192.168.1.1")/TCP(dport=1234,flags='S')

print "--SENT--"
SYN.display()

print "\n\n-- RECEIVED --"
response=sr1(SYN,timeout=1,verbose=0)
response.display()
## 收到SYN-ACK后，此时操作系统内核会自动回复RST，中断连接
if int(response[TCP].flags)==18:
print "\n\n -- SENT --"
A=IP(dst="192.168.1.1")/TCP(dport=1234,flags='A',ack=(response[TCP].seq+1))
A.display()
print "\n\n-- RECEIVED --"
```

response2=sr1(A.,timeout=1,verbose=0)

response2.display()

else:

print "SYN-ACK not returned"

全连接端口扫描

- nmap -sT 1.1.1.1 -p 80 (sT：TCP全连接端口扫描)
- nmap -sT 1.1.1.1 -p 80,21,25
- nmap -sT 1.1.1.1 -p 80-2000
- nmap -sT -iL iplist.txt -p 80
- 默认1000个常用端口

全连接端口扫描

- dmitry
  - 功能简单，但使用简便
  - 默认150个常用端口

- dmitry -p 172.16.36.135
- dmitry -p 172.16.36.135 -o output （o：save output to file）

全连接端口扫描

- nc -nv -w 1 -z 192.168.60.4 1-100 （w：超时时间，v：verbose详细信息，n：不做域名解析）
- for x in $(seq 20 30); do nc -nv -w 1 -z 1.1.1.1 $x; done | grep open
- for x in $(seq 1 254); do nc -nv -w 1 -z 1.1.1.$x 80; done