

TCP协议区分windows和linux实践 – 远程系...

一、前言

最近看到可以通过TCP协议的重试次数识别不同系统（之前都是通过TTL识别，但是不是很靠谱），比较感兴趣，遂进行探索，最后也将两种方式结合武器化。

二、如何识别系统

- 1、网络协议栈：比如nmap通过数据包的字段、字段内容进行判断。
- 2、应用类型：比如SSH一般都是Linux、RDP 一般都是windows
- 3、应用返回：比如banner返回以及一些报错

```
Downloads curl http://185.137.122.24/ -v
* Trying 185.137.122.24:80...
* Connected to 185.137.122.24 (185.137.122.24) port 80
> GET / HTTP/1.1
> Host: 185.137.122.24
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.18.0 (Ubuntu)
< Date: Mon, 16 Sep 2024 09:26:26 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Thu, 26 May 2022 10:15:52 GMT
< Connection: keep-alive
< ETag: "628f5358-264"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
}
```

这里我们武器化一个好用的通过协议识别的工具。

三、实践

这里是根据TCP三次握手的第二次失败重传的次数去判断的，经过测试这种方法是靠谱的。那么其他情况的重传、网络异常处理情况等等都可以作为特征区分。

linux系统

linux一共传6次，没收到rst的情况下

```

22:52:14.379143 IP 10.0.4.12.41207 > 74.119.193.24.22: Flags [S], seq 2889686353, win 8192, length 0
22:52:14.660129 IP 74.119.193.24.22 > 10.0.4.12.41207 ① Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0
22:52:15.701984 IP 74.119.193.24.22 > 10.0.4.12.41207 ② Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0
22:52:17.751943 IP 74.119.193.24.22 > 10.0.4.12.41207 ③ Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0
22:52:21.790153 IP 74.119.193.24.22 > 10.0.4.12.41207 ④ Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0
22:52:30.102488 IP 74.119.193.24.22 > 10.0.4.12.41207 ⑤ Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0
22:52:46.488850 IP 74.119.193.24.22 > 10.0.4.12.41207 ⑥ Flags [S.], seq 2458855162, ack 2889686354, win 29200, options [mss 1330], length 0

```

windows

windows一共传3次，没收到rst的情况下

```

23:08:06.467237 IP 10.0.4.12.13507 > 208.94.245.242.1433: Flags [S], seq 2574941479, win 8192, length 0
23:08:06.670098 IP 208.94.245.242.1433 > 10.0.4.12.13507 ① Flags [S.], seq 36700417, ack 2574941480, win 8192, options [mss 1424], length 0
23:08:09.670987 IP 208.94.245.242.1433 > 10.0.4.12.13507 ② Flags [S.], seq 36700417, ack 2574941480, win 8192, options [mss 1424], length 0
23:08:15.685579 IP 208.94.245.242.1433 > 10.0.4.12.13507 ③ Flags [S.], seq 36700417, ack 2574941480, win 8192, options [mss 1424], length 0

```

为了防止内核自动发送的rst不达到目标机器，使用iptables进行拦截（网上查询scapy可以通过网络延迟不让RST，但是我这里没成功，还是通过iptables拦截）

```

1 # 过滤rst
2 iptables -A OUTPUT -p tcp --tcp-flags RST RST -d 127.0.0.1 -j DROP
3 # 可以人工观察
4 tcpdump -n host 127.0.0.1 -vv

```

四、武器化

这里给出脚本，在10秒搜集syn+ack返回包的次数，并且打印TTL用于辅助识别。

```

1 from scapy.all import *
2 from scapy.layers.inet import IP, TCP
3
4 count = 0
5
6
7 def send_tcp_syn(ip_str, port_int):
8     ans = sr1(IP(dst=ip_str) / TCP(dport=port_int, flags="S", sport=RandShort()))
9
10
11 def prn(pkt):
12     global count
13     count = count + 1
14
15     if pkt.haslayer(IP) and pkt.haslayer(TCP):
16         ip_layer = pkt.getlayer(IP)
17         tcp_layer = pkt.getlayer(TCP)
18         # 获取窗口大小
19         window_size = tcp_layer.window
20

```

```

21     # 获取最大段大小 (MSS) 选项
22     mss_option = ""
23     for option in tcp_layer.options:
24         if option[0] == 'MSS':
25             mss_option = option[1]
26             break
27     print(f"Source IP: {ip_layer.src}, Destination IP: {ip_layer.dst}, T
28     print(f"Source Port: {tcp_layer.sport}, Destination Port: {tcp_layer
29
30
31 def get_result():
32     global count
33     time.sleep(10)
34     if count > 3:
35         print("linux")
36
37     if count == 3:
38         print("windows")
39     os._exit(0)
40
41
42 def listen_port(interface_str, ip_str):
43     sniff(iface=interface_str, filter='tcp and src host %s and tcp[13:1] = 1
44
45
46 if __name__ == '__main__':
47     interface = ""
48     if platform == "darwin":
49         interface = "en0"
50     elif platform == "linux":
51         interface = "eth0"
52
53     if not interface:
54         print("No interface specified")
55         exit()
56
57     if len(sys.argv) > 1:
58         target_ip = sys.argv[1]
59         port = sys.argv[2]
60     else:
61         target_ip = "127.0.0.1"
62         port = 1433
63
64     print(f"iptables -A OUTPUT -p tcp --tcp-flags RST RST -d {target_ip} -j
65     listen_thread = threading.Thread(target=listen_port, args=(interface, ta
66     listen_thread.start()
67
68     result_thread = threading.Thread(target=get_result, args=())
69     result_thread.start()

```

70

```
send_tcp_syn(target_ip, int(port))
```

```
root@VM-4-12-ubuntu:/home/ubuntu/poc# vim CheckSystemByPort.py
root@VM-4-12-ubuntu:/home/ubuntu/poc# python3 CheckSystemByPort.py 180.102.211.237 443
linux
root@VM-4-12-ubuntu:/home/ubuntu/poc# python3 CheckSystemByPort.py 208.94.245.242 1433
windows
root@VM-4-12-ubuntu:/home/ubuntu/poc#
```

1

```
python3 CheckSystemByPort.py 127.0.0.1 1433
```

五、总结

这里总结了三种远程识别系统的方法，并且武器化了一个通过TCP协议栈识别的工具（相比于nmap靠谱点）