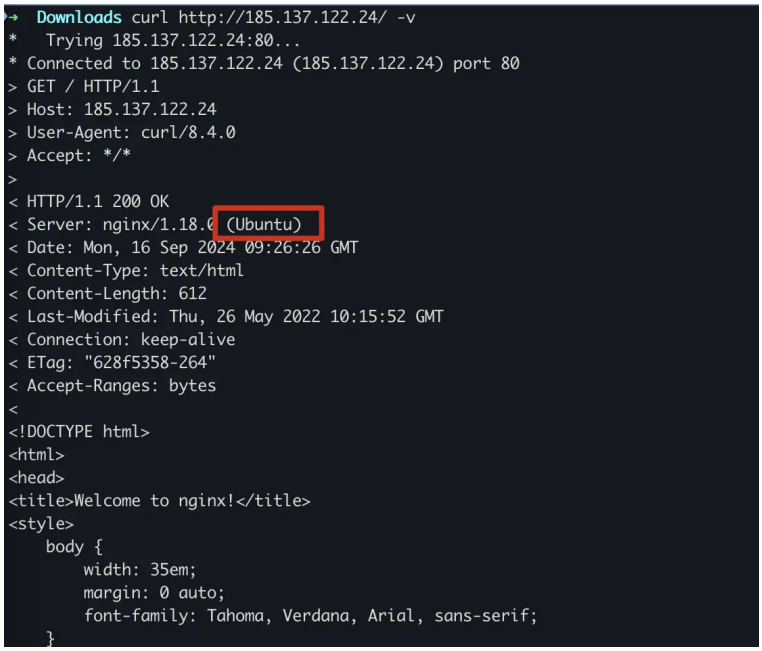
TCP协议区分windows和linux实践 – 远程系...

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



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 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 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 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 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 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 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, leng
th 0
23:08:06.670098 IP 208.94.245.242.1433 > 10.0.4.12.13507 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 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 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）

```
# 过滤rst
iptables -A OUTPUT -p tcp --tcp-flags RST RST -d 127.0.0.1 -j DROP
# 可以人工观察
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。

```
from scapy.all import *
from scapy.layers.inet import IP, TCP

count = 0


def send_tcp_syn(ip_str, port_int):
    ans = sr1(IP(dst=ip_str) / TCP(dport=port_int, flags="S", sport=RandShor


def prn(pkt):
    global count
    count = count + 1

    if pkt.haslayer(IP) and pkt.haslayer(TCP):
        ip_layer = pkt.getlayer(IP)
        tcp_layer = pkt.getlayer(TCP)
        # 获取窗口大小
        window_size = tcp_layer.window

```

```
        # 获取最大段大小（MSS）选项
        mss_option = ""
        for option in tcp_layer.options:
            if option[0] == 'MSS':
                mss_option = option[1]
                break
        print(f"Source IP: {ip_layer.src}, Destination IP: {ip_layer.dst}, T
        print(f"Source Port: {tcp_layer.sport}, Destination Port: {tcp_layer


def get_result():
    global count
    time.sleep(10)
    if count > 3:
        print("linux")

    if count == 3:
        print("windows")
    os._exit(0)


def listen_port(interface_str, ip_str):
    sniff(iface=interface_str, filter='tcp and src host %s and tcp[13:1] = 1


if __name__ == '__main__':
    interface = ""
    if platform == "darwin":
        interface = "en0"
    elif platform == "linux":
        interface = "eth0"

    if not interface:
        print("No interface specified")
        exit()

    if len(sys.argv) > 1:
        target_ip = sys.argv[1]
        port = sys.argv[2]
    else:
        target_ip = "127.0.0.1"
        port = 1433

    print(f"iptables -A OUTPUT -p tcp --tcp-flags RST RST -d {target_ip} -j
    listen_thread = threading.Thread(target=listen_port, args=(interface, ta
    listen_thread.start()

    result_thread = threading.Thread(target=get_result, args=())
    result_thread.start()
```

```
    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#
```

```
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）