APEC 2005 GIKI

Host Fingerprinting and Firewalking With hping

Naveed Afzal

National University Of Computer and Emerging Sciences, Lahore, Pakistan
Email: 1608@nu.edu.pk
Naveedafzal <AT> gmail.com

Abstract: The purpose of this paper is to discuss some techniques that can be effectively used in remote host fingerprinting. The paper will specially cover the cases where network hosts are behind firewalls. We will explain the techniques with various tools but the majority of the work is based on a simple and powerful utility named hping. This paper assumes that reader has a basic understanding of remote host fingerprinting and Transmission Control Protocol/Internet Protocol (TCP/IP). We will review both; the service port fingerprinting and OS fingerprinting in certain fire walled environments and will try to analyze the methods in detail that brings us the advantages and disadvantages of some techniques. Familiarity with hping and nmap will be useful for understanding the methods.

Key words*:* hping, nmap, iptables, tcpdump, sniffing, NAT, VMWare

1. Introduction

Remote host fingerprinting is the process of identifying the opened service ports and operating system of a machine over the network. This is usually achieved by various kinds of active and passive scanning techniques, by sending several packets to the remote machine and reviewing the responses. The generally available tools including nmap do a fairly good job in scanning and guessing the remote operating system. But in the cases where a host is fire walled these tools do not help much, either producing ambiguous or incorrect results. This is especially true for the machines, which are heavily fire walled and only allow very small number of packets to be forwarded and replied. In those cases we require some other methods to correctly determine the state of a remote machine. We will examine some of these methods including RING scan and ICMP scans. The first section describes various port scanning techniques while the next section throws some light on OS fingerprinting.

2. Port Knocking

We start with general port scanning techniques with certain tools including nmap and hping. We will discuss the common SYN, SYNACK scanning first and the behavior of various hosts upon reception of these TCP packets. Then we will see how the results may vary with the machines that are fire walled with those ones, which are not. Afterwards some advanced techniques will be discussed including the FIN scans and UDP scans on firewalled hosts.

2.1 Hping

Hping is described as one of the tools that can be effectively used for scanning, fingerprinting and firewall testing. Some of its powerful features include the ability to send custom crafted packets with several protocols and performing remote scanning. This is very handy for examining the response of various custom created packets.

2.2 Nmap

Network Mapper (nmap) is a famous network-auditing tool that can be used for advanced port scanning and OS detection. It has a powerful set of features available including passive scanning and idle scanning, though it does not have the ability to send custom packets like hping.

2.3 Testing with half open scan (SYN)

The idea of half open scanning (also referred as SYN scanning) is simple. Without completing the TCP three way handshake, send an initial SYN packet and wait for the response, if the SYN ACK is received it means the remote port is opened, otherwise you will receive a packet with RST flag set that is an indication of closed port.

2.3.1 Filtered and Closed Ports

However in the case of some firewalls, the firewall can simply block access to certain ports, which are opened, those are said to be filtered ports. In these situations we do not get any response of our initial SYN packet. Also many firewalls block RST packets in response to closed ports. Thus in those situations its hard to differentiate which ports are closed and which are filtered. Here are the results of scanning a live host without any firewall with normal nmap scans

```
root@life#nmap   –P0   –p   1,2,21,80
202.83.174.99
Interesting          ports          on
(202.83.174.99):
PORT        STATE  SERVICE
1/tcp       closed      tcpmux
2/tcp       closed      compressnet
21/tcp      open        ftp
80/tcp      open        http
Nmap finished: 1 IP address (1 host
up) scanned in 1.140 seconds
```

As we can see from the output, the host doesn’t seems to be fire walled, we have scanned for ports 1,2,21,80 and it has indicated that ports 1,2 are closed and other two are open.
Lets take an example of another host.

```
root@life#nmap   –P0   –p   1,2,21,80
209.41.165.180
Interesting          ports          on
(209.41.165.180):
PORT   STATE    SERVICE
1/tcp       filtered    tcpmux
2/tcp       filtered    compressnet
21/tcp      open        ftp
80/tcp      open        http
Nmap finished: 1 IP address (1 host
up) scanned in 4.047 seconds
```

If you see this one you can instantly find the change the STATE of first two ports 1,2 is marked as filtered. Just by looking at this information you cannot tell exactly whether port 1 and 2 is closed or opened. The only information available is that this port is being filtered. However as we know that all closed ports should send out an RST packet in normal circumstances if they are not filtered. Lets try sending some custom packets to generally used ports with hping and see the behavior

```
root@life#hping   -S   -p   80   -c   2
209.41.165.180
HPING 209.41.165.180 (WAN (PPP/SLIP)
Interface 209.41.165.180): S set, 40
headers + 0 data bytes
len=44  ip=209.41.165.180  ttl=63  DF
id=62648   sport=80   flags=SA   seq=0
win=65535 rtt=2359.0 ms

len=44  ip=209.41.165.180  ttl=63  DF
id=63296   sport=80   flags=SA   seq=1
win=65535 rtt=1359.0 ms
      ---     209.41.165.180     hping
statistics ---
      2    packets    transmitted,    2
packets received, 0% packet loss
round-trip       min/avg/max        =
1359.0/1859.0/2359.0 ms
```

Here I have issued the command
hping -S -p 80 -c 2 209.41.165.180
to send 2 packets with SYN flag set at port 80 and as we can see the response, we got two packets with flag=SA which is an indication to our SYN acknowledgement. The DF indicates that do not fragment bit was set. Now return to our previous problem , we sent the same two packets to the port 1

```
root@life#hping   -S   -p   1   -c   2
209.41.165.180
HPING 209.41.165.180 (WAN (PPP/SLIP)
Interface 209.41.165.180): S set, 40
headers + 0 data bytes

--- 209.41.165.180 hping statistics -
--
2  packets  transmitted,  0  packets
received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0
ms
```

Here we got nothing back, the two packets were lost which is verifying that this port is being filtered by the firewall and it is blocking any kind of response at this port. Now lets send same packet to some more ports

```
root@life#hping     -S     -p     ++20
209.41.165.180
HPING 209.41.165.180 (WAN (PPP/SLIP)
Interface 209.41.165.180): S set, 40
headers + 0 data bytes

len=44  ip=209.41.165.180  ttl=108  DF
id=9352   sport=21   flags=SA    seq=1
win=16616 rtt=1062.0 ms

len=40    ip=209.41.165.180    ttl=108
id=10442   sport=22   flags=RA   seq=2
win=0 rtt=562.0 ms

len=40    ip=209.41.165.180    ttl=108
id=11643   sport=23   flags=RA   seq=3
win=0 rtt=562.0 ms

len=44  ip=209.41.165.180  ttl=108  DF
id=13778   sport=25   flags=SA   seq=5
win=16616 rtt=562.0 ms

len=40    ip=209.41.165.180    ttl=108
id=40085  sport=49   flags=RA   seq=29
win=0 rtt=562.0 ms
```

```
len=40    ip=209.41.165.180    ttl=108
id=40941   sport=50   flags=RA   seq=30
win=0 rtt=562.0 ms
^C
root@life#
```

Here I have asked the hping to send SYN packets to the ports starting from 20 and increment port number by one each time. We can clearly see the difference that the values at certain ports include some flag=RA packets (Reset Acknowledged) which is indicating that those ports are closed and not being fire walled. Since we did not get any response from ports 20,24,26-48, which are being blocked by firewall. Thus it is an indication that those ports may also be closed. Because the firewall policy is set in such a way that all generally used ports are not being filtered. As we can see the port 443 (which is used for https and is generally opened) is responding with RST packet, which tells us that https service is not running and also it is not being blocked.

```
root@life#hping     -S     -p     443
209.41.165.180
HPING 209.41.165.180 (WAN (PPP/SLIP)
Interface 209.41.165.180): S set, 40
headers + 0 data bytes
len=40    ip=209.41.165.180    ttl=108
id=40924   sport=443   flags=RA   seq=0
win=0 rtt=23
8.0 ms
^C
root@life#
```

2.4 Testing with FIN packets

This is based on the fact that when a closed port receives a packet with FIN flag set , the normal behavior is to respond with RST packet. The open ports do not respond to this packet either. This is very useful for the cases where SYN packets are being blocked by the firewall. However this is not applicable when scanning the Windows machines as they do not respond to the individual FIN packets either.

Let us simulate a network with two hosts in Vmware. We install a Linux host in a vmworkstation and sent the FIN packets from another host. On the Linux host we first disabled all the normal SYN traffic . To block all the incoming packets with SYN flag set we can use the iptables.

2.4.1 Iptables

Iptables is basically a Linux based packet filtering tool that can be used for filtering the network packets. There are three built-in tables, each of which contains some predefine chains. The filter table is responsible for filtering (block or permit) and it consists of three chains namely INPUT, OUTPUT and FORWARD.

So we just add an entry in the INPUT chain for dropping the TCP packets with SYN flag set

```
life1# iptables –A INPUT –p tcp –tcp-
flags SYN –j DROP
```

This will allow all other traffic except SYN packets, now test our machine by sending FIN packets from a windows machine, (here I have used hping windows built on a win2k machine). By default the Linux host is listening on ports 21,22,80

Here is the output of sending SYN packets targetting port 80

```
E:\hping>hping –S –p 80 –c 10 LIFE1
HPING  LIFE1  (LAN  eth1)  Interface
192.168.10.2): S set, 40 headers + 0
data bytes

--- LIFE1 hping statistics ---
10  packets  transmitted,  0  packets
received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0
ms
```

Here we can see that all the 10 packets that are being sent to the machine are lost. If we send the packets to the port, which is known to be closed on the Linux machine, we get the same response.

```
E:\hping>hping –S –p 50 –c 10 LIFE1
HPING  LIFE1  (LAN  eth1)  Interface
192.168.10.2): S set, 40 headers + 0
data bytes
--- LIFE1 hping statistics ---
10  packets  transmitted,  0  packets
received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0
ms
```

Now lets send the packets with FIN flag set

```
E:\hping>hping –F –p 50 LIFE1
HPING  LIFE1  (LAN  eth1)  Interface
192.168.10.2): F set, 40 headers + 0
data bytes
len=40    ip=202.83.174.99    ttl=56
id=30859   sport=50   flags=RA   seq=0
win=0 rtt=24.0 ms
len=40    ip=202.83.174.99    ttl=56
id=30863   sport=50   flags=RA   seq=1
win=0 rtt=14.0 ms
^C
E:\hping>
```

Here we see that we got RA packets on a closed port, which was not responding previously, which is a clear indication that port is closed. Reader can verify that same packet at ports 21,22 and 80 is not responded which can be identified as open ports while all other responds with RA.

2.5 UDP ports

Scanning the UDP ports is relatively a tough job mostly because of its inhibit unreliability. Most common technique is to send packets to the UDP ports if you get back nothing, it is assumed that port is open because on closed port you get a Port Unreachable ICMP message from the target operating system under the normal circumstances. This is not always necessary that you get same type of response. Here is a UDP scan of a typical host with nmap

```
root@life#nmap -sU -p 21,53,80
yns1.yahoo.com

Interesting ports on 66.218.71.205:
PORT      STATE          SERVICE
21/udp    open|filtered    ftp
53/udp    open|filtered    domain
80/udp    open|filtered    http

Nmap finished: 1 IP address (1 host
up) scanned in 3.141 seconds
```

The results above seem not very interesting , nmap has failed to determine which ports are open and which are filtered or closed. It is ending up saying all the three ports either open or filtered, which is not the case. Since the host is a Name Server there is a huge probability that its UDP port 53 is opened for DNS type queries. So we do a little trick and scan it with hping. With the normal hping UDP type scan we also didn't got any response.

```
root@life#hping -2 -p 50++
yns1.yahoo.com
HPING yns1.yahoo.com (WAN (PPP/SLIP)
Interface 66.218.71.205): udp mode
set, 28 headers + 0 data bytes
6 packets transmitted, 0 packets
received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0
ms
```

Lets change our strategy, we first created a simple text file with junk data of around 120 bytes and then scanned the host again with the data as payload to each packet by the following command and at the same time I started tcpdump in a separate window and puts it into promiscuous mode and sniff all the network traffic

```
root@life/tmp#tcpdump
root@life#hping -2 -p ++50 -d 120 -E
file.txt yns1.yahoo.com
HPING yns1.yahoo.com (WAN (PPP/SLIP)
Interface 66.218.71.205): udp mode
set, 28 headers + 120 data bytes

len=46 ip=66.218.71.205 ttl=49
id=37187 seq=3 rtt=531.0 ms
^C
root@life#
```

As apparent here , we got a response lets view the tcpdump output for further analysis

```
root@life#tcpdump
tcpdump: listening on
\Device\NPF_GenericDialupAdapter
00:42:50.484375 IP life.2950 >
yns1.yahoo.com.50: UDP, length 120
00:42:51.484375 IP life.2951 >
yns1.yahoo.com.51: UDP, length 120
00:42:52.484375 IP life.2952 >
yns1.yahoo.com.52: UDP, length 120
00:42:53.484375 IP life.2953 >
yns1.yahoo.com.53: 24930 updateM+
[b2&3=0x6364][24930a] [25958q]
[25444n] [25958au][|domain]
00:42:53.953125 IP yns1.yahoo.com.53
> life.2953: 24930 updateM FormErr-
[0q] 0 /0/0 (12)
00:42:53.953125 IP life >
yns1.yahoo.com: ICMP life udp port
2953 unreachable, length 36
00:42:54.484375 IP life.2954 >
yns1.yahoo.com.54: UDP, length 120
```

The interested thing to note is that upon reception of our junk data the UDP port 53 is responded with an error message, which indicates it is opened. All the other packets were not responded at all. Another interesting way through which we can scan the udp ports is by checking the returned ICMP messages. In normal circumstances if we send a packet without any payload to a UDP port, which is closed the system, responds with an ICMP port Unreachable message. The opened ports do not respond to zero payload packets. This can be seen in the following example.

```
root@life#hping -2 -p 11 –c 3
202.179.137.59
HPING 202.179.137.59 (WAN (PPP/SLIP)
Interface 202.179.137.59): udp mode
set, 28
headers + 0 data bytes
ICMP Port Unreachable from
ip=202.179.137.59
ICMP Port Unreachable from
ip=202.179.137.59
ICMP Port Unreachable from
ip=202.179.137.59

--- 202.179.137.59 hping statistics -
--
3 packets transmitted, 3 packets
received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0
ms
```

As you have noticed, I sent 3 packets to the UDP port 11 which are responded with Port Unreachable ICMP type message. However the firewalls usually block such outgoing packets and we have presented a way to bypass the firewalls rule set in the previous example.

3. OS Fingerprinting

OS fingerprinting is usually harder in the firewalled environments as in those cases the firewalls may alter the contents of TCP/IP packets thus making the guess work wrong. The OS fingerprinting is categorized as either Active fingerprinting or Passive.

3.1 Passive OS Fingerprinting

In case of Passive fingerprinting the person who wants to fingerprint the target does not send any packet to the target , instead it uses some intermediate host (known as zombie) and tries to guess the target OS by calculating the difference between IPID sequence numbers. This is known as idle scan method. Or in some other way one can get the traffic going to and from the target and then can judge the target OS without any direct interaction with the target. Without discussing the passive fingerprinting lets talk about Active fingerprintg.

3.2 Active OS Fingerprinting

In Active fingerprinting a host normally sends some packets to the target and try to determine the OS from the responses by calculating some values in the options field of TCP/IP packets, this includes the timestamp values or the IPID sampling, type of service TOS,TCP ISN sampling, and fragmentation handling etc. Another old technique is to use the TTL value of an ICMP echo packet to determine the target OS and this is an easy way to differentiate between various OS, however this cannot differentiate the variants of same OS like win98 with XP or win2k. Usually the TTL values are set to a fixed one in each OS. Microsoft family sets it to default of 128 while Linux sets it to 256. Here is an example to determine the OS by the returned TTL value of ICMP echo packets. I simply sends a ping to the target machine and check the TTL value , the returned TTL value in this case is 113 a quick guess tells me that this may be some windows OS since they have a starting TTL of 128 and the remote host is around 16 hopes away from my machine (as it can be verified with traceroute) so 113 +15= 128

```
E:\>ping 209.41.165.180
Pinging 209.41.165.180 with 32 bytes of
data:
Reply from 209.41.165.180: bytes=32 time
38ms TTL=113
Reply from 209.41.165.180: bytes=32 time
51ms TTL=113
Ping statistics for 209.41.165.180:
    Packets: Sent = 2, Received = 2, Lost
= 0 (0% loss),
Approximate round trip times in milli-
seconds:
    Minimum = 38ms, Maximum =  51ms,
Average =  44ms
```

However it is not a very reliable guess , there may be some routing device or if the host is behind some NAT(Network Address Translation) this technique fails. Without going into the other common OS fingerprinting techniques which are being used by the tools like nmap, I cover a technique that is usually hard to implement but it can provide a good guess of remote OS detection ,it is known as the RING scan and there was also a tool available a while ago which can be used for guessing the remote OS. The idea here is to send some SYN packets to an open port and wait for the SYN ACK packets, but when you get back a SYN ACK packet silently drops it, the remote hosts will timeout after a certain delay and will resend the SYN ACK. By carefully calculating the delay between successive SYN ACK packets sent by various different hosts you can differentiate what is the target OS since various OS send back the packets with a certain amount of delays. This can be used very effectively for differentiating between the OSs, which have same kind of TCP stack and are behind a firewall , an example is that of FreeBSD and Windows2000, which share the same type of TCP stack. I am presenting here an example in which the nmap failed to determine the correct OS and simply guessed it both as FreeBSD , the reason being that one of the hosts was behind a firewall.
Scanning the first host `202.83.174.99`

```
Interesting    ports    on    ntc.net.pk
(202.83.174.99):
(The 97 ports scanned but not shown below
are in state: closed)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
Device type: general purpose
Running: FreeBSD 4.X
OS details: FreeBSD 4.6.2-RELEASE - 4.8-
RELEASE
```

The other one yielded

```
Interesting ports on 202.83.162.27:
(The 99 ports scanned but not shown
below are in state: filtered)
PORT   STATE SERVICE
80/tcp open  http
Device type: general purpose
Running: FreeBSD 4.X|5.X
OS    details:    FreeBSD    4.3    -
4.4PRERELEASE,  FreeBSD  4.9  -  5.1,
FreeBSD 5.0-RELEASE,
```

Now lets scan it with the SYN ACK technique discussed earlier, for that first you have to set up a local firewall rule to drop all packets SYNACK packets from the remote host.

```
life1# iptables –A INPUT –p tcp –j
DROP –s 202.83.162.27
```

Now send the SYN packets to its port 80 which is open and watch the tcpdump output, here is the log of tcpdump output which we received after issuing

```
root@life#   hping   -S   -p   80   -c   1
202.83.162.27
```

```
17:22:51.079596 202.134.134.230.1816 >
202.83.162.27.http: S win 512

17:22:51.208938 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840

17:22:53.218939 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840

17:23:57.218939 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840

17:23:03.218939 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840

17:23:11.468939 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840

17:24:21.618938 202.83.162.27.http >
202.134.134.230.1816: S ack win 5840
```

If we calculate the time difference between each received SYN ACK packet it is around 2,4,6,7,10 seconds successively.

Now lets do the same scan with first host, which we are sure running a FreeBSD operating system. The tcpdump output is given below after the command

```
root@life#   hping   -S   -p   80   -c   1
202.83.174.99
```

```
17:45:50.019746 202.134.134.230.2644 >
202.83.174.99.http: S win 512

17:45:50.148940 202.83.174.99.http >
202.134.134.230.2644: S ack win 5840

17:45:54.108939 202.83.174.99.http >
202.134.134.230.2644: S ack win 5840

17:46:00.108939 202.83.174.99.http >
202.134.134.230.2644: S ack win 5840

17:46:12.308939 202.83.174.99.http >
202.134.134.230.2644: S ack win 5840

17:46:36.378938 202.83.174.99.http >
202.134.134.230.2644: S ack win 5840
```

The calculation here tells us that the difference now is around 4,6,12,24 and then no SYN ACK is received. Experiments with some other hosts we can determine that the retransmission time of SYN ACK packets in FreeBSD system is usually 3,6,12,24 while the windows hosts goes in this way 2,4,6,8,10. This can provide a useful hint for determine the correct Operating system when the other tools fails and unable to provide the correct results. Note however that these values presented above are not much accurate and have been determined by judging a few hosts, I have provided two host values one with Windows2000 and other with FreeBSD 4-6. One may get the more accurate values by examining several dozen hosts. This technique has several other extensions as well , for example instead of checking for the initial SYN ACK response lets continue and complete the standard TCP three-way handshake and then close the connection by sending FIN packet but now don't send any Acknowledgements to the FIN packets , the situations becomes this

Host1 --→	SYN ----→	Host 2
Host2 --→	SYN ACK ---→	Host1
Host1 --→	ACK ---→	Host2
Host1 --→	FIN ----→	Host2
Host2 --→	FIN ACK -----→	Host1
Host2 --→	FIN ACK -----→	Host1
Host2 --→	FIN ACK -----→	Host1

…………………………………………………………

And so on, the first host didn't send any response to the Host 2 for the FIN ACK packet. Still another option is to use the RST packets.

4. Conclusions

The conventional automated methods for fingerprinting can although achieve good results, they are not well suited for the changing environments several different techniques may be combined to get most accurate results. The techniques described in this paper are the manual techniques which may be used intelligently to know more about the network. However the list provided is not complete and there are several other ways available. For example in this paper passive fingerprinting has not been discussed because of its wide scope.

We have seen that most firewalls do not allow certain normal traffic and in most of the cases they hide their identity like the systems which do not send any kind of response against various request packets. But a careful examination of this behaviour can lead to finding closed, filtered and opened ports of both TCP and UDP and the variation of such behaviour can make one guess the remote operating system.

Acknowledgements

Sherjeel A. Qureshi, for his feedback and support.
A. Salman , for his appreciation and suggestions.

REFERENCES

[1] Hping, a command-line oriented TCP/IP packet assembler/analyzer. Http://www.hping.org
[2] Network Mapper (nmap) . A utility for network exploration and auditing. http://insecure.org/nmap
[3] *Ring out the old, RING in the New*, Franck Veysset, Olivier Courtay, and Olivier Heen April 2002.
[4] *Advanced Fingerprinting* , Erwan Arzur March 2005.
[5] *Chatter On The Wire ,OS Fingerprinting* Erric Kollmann August 2005.