PARATRACE - SNEAKY TRACEROUTING

Traceroute is one of those old and useful commands. I am sure we have all used it to see exactly where a packet is going and how long it takes to get there. After all that is what it is designed for. You can find it on linux-style boxes as *traceroute*, even windows machines have a version, called *tracert*. But these days most firewalls block these packets. You see, an attacker can use the information gathered from the *traceroute* output to create a map of your network layout, to try to identify where your firewalls and routers are situated, and even more. It is for these reasons that firewall administrators began blocking *traceroute*. And there it would have ended if not for Dan Kaminsky of DoxPara Research and his brilliant tool *paratrace* - or *parasitic traceroute*. *Paratrace* is a tool which looks for a suitable and established connection to the target host, it then manipulates the packets in that connection in order to perform some *traceroute* functionality.

The environment
The test network I am using is quite simple..

Host A	Firewall	Host B
192.168.10.80	192.168.10.13 / 10.0.0.11	10.0.0.50

Lets start with a fully open firewall..

```
traceroute 10.0.0.50
traceroute to 10.0.0.50 (10.0.0.50), 30 hops max, 38 byte packets
 1  192.168.10.13 (192.168.10.13)  0.339 ms  0.270 ms  0.277 ms
 2  10.0.0.50 (10.0.0.50)  0.696 ms  0.640 ms  0.617 ms
```

```
traceroute 192.168.10.80
traceroute to 192.168.10.80 (192.168.10.80), 30 hops max, 40 byte packets
 1  10.0.0.11 (10.0.0.11)  3.880 ms  0.666 ms  0.621 ms
 2  192.168.10.80 (192.168.10.80)  0.850 ms  0.758 ms  0.731 ms
```

Thus you can see both hosts can perform a *traceroute* to the other.

Not so trusting
Now lets be a lot more paranoid, we will change the firewall rules so that only TCP traffic destined for port 22 (*ssh*) is allowed through, and then lets try again..

Host A

```
traceroute 10.0.0.50 -m 5
traceroute to 10.0.0.50 (10.0.0.50), 5 hops max, 38 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
```

```
ping 10.0.0.50 -c 10
PING 10.0.0.50 (10.0.0.50) 56(84) bytes of data.

--- 10.0.0.50 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9012ms
```

Host B

```
traceroute 192.168.10.80 -m 5
traceroute to 192.168.10.80 (192.168.10.80), 5 hops max, 40 byte
packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
```

```
 ping 192.168.10.80 -c 10
PING 192.168.10.80 (192.168.10.80): 56 data bytes

--- 192.168.10.80 ping statistics ---
10 packets transmitted, 0 packets received, 100% packet loss
```

Here we can see that we are not going to get anything through. But what if the sneaky person at *Host A* still didn't

want to call it quits?

Getting sneaky
The first thing the attacker will want to try to find is an open TCP port (any port will do) and in our setup he can use *ssh*. So he tries..

```
ssh bob@10.0.0.50
bob@10.0.0.50's password:
```

Straight away he sees that he can access the *ssh* service. Now lets pause and take a look at what a normal *ssh* TCP connection setup looks like..

```
The first section here is the connection setup all TCP connections use, the SYN, SYN-ACK, ACK
```

```
19:21:44.377382 192.168.10.80.33562 > 10.0.0.50.ssh: S [tcp sum ok] 2808026547:2808026547(0) win 5840 <mss 1460,sackOK,timestamp 33701443 0,nop,wscale 0> (DF) (ttl 63, id 11248, len 60)
19:21:44.377382 10.0.0.50.ssh > 192.168.10.80.33562: S [tcp sum ok] 3451264356:3451264356(0) ack
19:21:44.377382 192.168.10.80.33562 > 10.0.0.50.ssh: . [tcp sum ok] ack 1 win 5840 <nop,nop,timestamp 33701443 35624172> (DF) (ttl 63, id 11249, len 52)
2808026548 win 32120 <mss 1460,sackOK,timestamp 35624172 33701443,nop,wscale 0> (DF) (ttl 64, id 41656, len 60)
```

```
Then the normal connection carries on. As you can see, the timestamps, ttl's, id's, etc are all normal and expected.
```

```
19:21:44.387382 10.0.0.50.ssh > 192.168.10.80.33562: P [tcp sum ok] 1:24(23) ack 1 win 32120 <nop,nop,timestamp 35624173 33701443> (DF) (ttl 64, id 41657, len 75)
19:21:44.387382 192.168.10.80.33562 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 5840 <nop,nop,timestamp 33701444 35624173> (DF) (ttl 63, id 11250, len 52)
19:21:44.387382 192.168.10.80.33562 > 10.0.0.50.ssh: P [tcp sum ok] 1:25(24) ack 24 win 5840 <nop,nop,timestamp 33701444 35624173> (DF) (ttl 63, id 11251, len 76)
19:21:44.387382 10.0.0.50.ssh > 192.168.10.80.33562: . [tcp sum ok] ack 25 win 32120 <nop,nop,timestamp 35624173 33701444> (DF) (ttl 64, id 41658, len 52)
19:21:44.387382 192.168.10.80.33562 > 10.0.0.50.ssh: P 25:569(544) ack 24 win 5840 <nop,nop,timestamp 33701444 35624173> (DF) (ttl 63, id 11252, len 596)
19:21:44.397382 10.0.0.50.ssh > 192.168.10.80.33562: . [tcp sum ok] ack 569 win 32120 <nop,nop,timestamp 35624174 33701444> (DF) (ttl 64, id 41659, len 52)
19:21:44.407382 10.0.0.50.ssh > 192.168.10.80.33562: P 24:632(608) ack 569 win 32120 <nop,nop,timestamp 35624175 33701444> (DF) (ttl 64, id 41660, len 660)
19:21:44.407382 192.168.10.80.33562 > 10.0.0.50.ssh: P [tcp sum ok] 569:593(24) ack 632 win 6688 <nop,nop,timestamp 33701446 35624175> (DF) (ttl 63, id 11253, len 76)
19:21:44.427382 10.0.0.50.ssh > 192.168.10.80.33562: . [tcp sum ok] ack 593 win 32120 <nop,nop,timestamp 35624177 33701446> (DF) (ttl 64, id 41661, len 52)
19:21:44.497382 10.0.0.50.ssh > 192.168.10.80.33562: P 632:1056(424) ack 593 win 32120 <nop,nop,timestamp 35624184 33701446> (DF) (ttl 64, id 41662, len 476)
19:21:44.527382 192.168.10.80.33562 > 10.0.0.50.ssh: P 593:1009(416) ack 1056 win 7904 <nop,nop,timestamp 33701457 35624184> (DF) (ttl 63, id 11254, len 468)
19:21:44.537382 10.0.0.50.ssh > 192.168.10.80.33562: . [tcp sum ok] ack 1009 win 32120 <nop,nop,timestamp 35624188 33701457> (DF) (ttl 64, id 41663, len 52)
19:21:45.947382 10.0.0.50.ssh > 192.168.10.80.33562: P 1056:1792(736) ack 1009 win 32120 <nop,nop,timestamp 35624328 33701457> (DF) (ttl 64, id 41664, len 788)
19:21:45.967382 192.168.10.80.33562 > 10.0.0.50.ssh: P [tcp sum ok] 1009:1025(16) ack 1792 win 9568 <nop,nop,timestamp 33701602 35624328> (DF) (ttl 63, id 11255, len 68)
19:21:45.987382 10.0.0.50.ssh > 192.168.10.80.33562: . [tcp sum ok] ack 1025 win 32120 <nop,nop,timestamp 35624333 33701602> (DF) (ttl 64, id 41665, len 52)
19:21:45.987382 192.168.10.80.33562 > 10.0.0.50.ssh: P 1025:1073(48) ack 1792 win 9568 <nop,nop,timestamp 33701604 35624333> (DF) (ttl 63, id 11256, len 100)
19:21:45.987382 10.0.0.50.ssh > 192.168.10.80.33562: P 1792:1840(48) ack 1073 win 32120 <nop,nop,timestamp 35624333 33701604> (DF) (ttl 64, id 41666, len 100)
```

So our attacker now knows that he (or she) has a way in. First thing they do is setup the *paratrace* listener..

```
paratrace -v -b1k 10.0.0.50
```

This tells *paratrace* to be verbose (*-v*), not to use to much bandwidth (*-b1k*) and to target 10.0.0.50. The attacker

then tries to connect to the *ssh* service again. Now before we look at what *paratrace* tells us, lets take look at the traffic..

```
As per usual the first three packets are for the normal 3-way handshake. Paratrace
looks for this and waits for this to finish before starting work.
```

```
19:19:48.877382 192.168.10.80.33561 > 10.0.0.50.ssh: S [tcp sum ok]
2692203339:2692203339(0) win 5840 <mss 1460,sackOK,timestamp 33689895 0,nop,wscale 0>
(DF) (ttl 63, id 53868, len 60)
19:19:48.877382 10.0.0.50.ssh > 192.168.10.80.33561: S [tcp sum ok]
3331681981:3331681981(0) ack 2692203340 win 32120 <mss 1460,sackOK,timestamp 35612623
33689895,nop,wscale 0> (DF) (ttl 64, id 41634, len 60)
19:19:48.877382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 1 win 5840
<nop,nop,timestamp 33689896 35612623> (DF) (ttl 63, id 53869, len 52)
```

```
Now lets take a look. Everything seems normal, except for a couple of packets where the
ttl and the id flags have been changed. Paratrace uses these probes to perform it's
traceroute functionality. The incrementing counters are used the same way a normal
traceroute uses the ttl flags.
```

```
19:19:48.877382 10.0.0.50.ssh > 192.168.10.80.33561: P [tcp sum ok] 1:24(23) ack 1 win
32120 <nop,nop,timestamp 35612624 33689896> (DF) (ttl 64, id 41635, len 75)
19:19:48.877382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 5840
<nop,nop,timestamp 33689896 35612624> (DF) (ttl 63, id 53870, len 52)
19:19:48.877382 192.168.10.80.33561 > 10.0.0.50.ssh: P [tcp sum ok] 1:25(24) ack 24 win
5840 <nop,nop,timestamp 33689896 35612624> (DF) (ttl 63, id 53871, len 76)
19:19:48.887382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 25 win 32120
<nop,nop,timestamp 35612624 33689896> (DF) (ttl 64, id 41636, len 52)
19:19:48.887382 192.168.10.80.33561 > 10.0.0.50.ssh: P 25:569(544) ack 24 win 5840
<nop,nop,timestamp 33689896 35612624> (DF) (ttl 63, id 53872, len 596)
19:19:48.887382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 569 win 32120
<nop,nop,timestamp 35612625 33689896> (DF) (ttl 64, id 41637, len 52)
19:19:48.907382 10.0.0.50.ssh > 192.168.10.80.33561: P 24:632(608) ack 569 win 32120
<nop,nop,timestamp 35612626 33689896> (DF) (ttl 64, id 41638, len 660)
19:19:48.907382 192.168.10.80.33561 > 10.0.0.50.ssh: P [tcp sum ok] 569:593(24) ack 632
win 6688 <nop,nop,timestamp 33689899 35612626> (DF) (ttl 63, id 53873, len 76)
19:19:48.917382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 593 win 32120
<nop,nop,timestamp 35612628 33689899> (DF) (ttl 64, id 41639, len 52)
19:19:48.977382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 32120
<nop,nop,timestamp 35612624 33689896> (DF) [ttl 1] (id 2, len 52)
19:19:48.977382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 593 win 32120
<nop,nop,timestamp 35612633 35612624> (DF) (ttl 64, id 41640, len 52)
19:19:48.997382 10.0.0.50.ssh > 192.168.10.80.33561: P 632:1056(424) ack 593 win 32120
<nop,nop,timestamp 35612636 35612624> (DF) (ttl 64, id 41641, len 476)
19:19:49.027382 192.168.10.80.33561 > 10.0.0.50.ssh: P 593:1009(416) ack 1056 win 7904
<nop,nop,timestamp 33689910 35612636> (DF) (ttl 63, id 53874, len 468)
19:19:49.027382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 593 win 32120
<nop,nop,timestamp 35612638 35612624> (DF) (ttl 64, id 41642, len 52)
19:19:49.057382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 32120
<nop,nop,timestamp 35612624 33689896> (DF) (ttl 2, id 3, len 52)
19:19:49.057382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 593 win 32120
<nop,nop,timestamp 35612641 35612624> (DF) (ttl 64, id 41643, len 52)
19:19:49.137382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 32120
<nop,nop,timestamp 35612624 33689896> (DF) (ttl 3, id 4, len 52)
19:19:49.137382 10.0.0.50.ssh > 192.168.10.80.33561: . [tcp sum ok] ack 593 win 32120
<nop,nop,timestamp 35612649 35612624> (DF) (ttl 64, id 41644, len 52)
19:19:49.197382 10.0.0.50.ssh > 192.168.10.80.33561: P 632:1056(424) ack 593 win 32120
<nop,nop,timestamp 35612656 35612624> (DF) (ttl 64, id 41645, len 476)
19:19:49.197382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 1056 win 7904
<nop,nop,timestamp 33689928 35612656,nop,nop,sack sack 1 {632:1056} > (DF) (ttl 63, id
53875, len 64)
19:19:49.217382 192.168.10.80.33561 > 10.0.0.50.ssh: . [tcp sum ok] ack 24 win 32120
<nop,nop,timestamp 35612624 33689896> (DF) (ttl 4, id 5, len 52)
```

So what you ask, it's just four packets, so what. Well lets see what information *paratrace* gave our attacker..

```
paratrace -v -b1k 10.0.0.50
Stat|=====IP_Address==|Port=|Hops|==Time==|=============Details============|
Waiting to detect attachable TCP connection to host/net: 10.0.0.50
10.0.0.50:22/32 1-5
  UP:        10.0.0.50:22    [01]   1.375s
SENT:        10.0.0.50:22    [00]   0.000s
Got 94 on eth0:
```

```
  IP: i=192.168.10.13->192.168.10.80 v=4 hl=5 s=192 id=8849 o=0 ttl=255 pay=60
ICMP:  IP: i=192.168.10.80->10.0.0.50 v=4 hl=5 s=0 id=1 o=64 ttl=1 pay=32
ICMP: TCP: p=33561->22, s/a=2692203364
001 =    192.168.10.13|22    [01]   1.385s(   192.168.10.80 -> 10.0.0.50       )
SENT:        10.0.0.50:22    [00]   0.079s
SENT:        10.0.0.50:22    [00]   0.159s
SENT:        10.0.0.50:22    [00]   0.239s
SENT:        10.0.0.50:22    [00]   0.319s
```

There you go, our attacker has the same information he would normally get from traceroute, and with this they can once more begin to start mapping out your network.

Last Words
Paratrace is one of those tools that teaches us security professionals two very important lessons..

1. There are always new methods being found to bypass our defenses. Never be complacent.
2. Just a firewall is not enough. Defense in depth is a must.

Take a look at the tool, play around with it and get to know it. Remember, have fun and learn.