Hping2 Basics

The classic ping command has served the IT community well. But with the never ending escalation of security and the blocking of most ICMP traffic at both the border as well as the host, the plain old ping command is no longer enough to accomplish even the simplest of network administrative tasks. This is exactly where a handy tool named hping2 comes into the fold to lift the capabilities of ping to heights it never imagined.

For a specific definition of what exactly hping2 is, here is an excerpt from http://www.hping.org:

Hping2 is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired by the ping(8) Unix command, but hping isn't only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features. All header fields can be modified and controlled using the command line. A good understanding of IP and TCP/UDP is mandatory to use and understand the utility.

While hping2 was mainly used as a security tool in the past, it can be used in many ways. Below is a subset of the stuff you can do using hping2:

- Firewall testing
- Advanced port scanning
- Network testing, using different protocols, TOS, fragmentation
- Manual path MTU discovery
- Advanced traceroute, under all the supported protocols
- Remote OS fingerprinting
- Remote uptime guessing
- TCP/IP stacks auditing
- hping can also be useful to students that are learning TCP/IP.

For a more detailed description and to download the binaries, visit http://www.hping.org. You can obtain a full working version of hping2 on a bootable CD (among other tools) at http://www.knoppix-std.org or on BackTrack.

While hping2 can do all of that, we will start by learning how hping2 can manipulate and craft packets for the testing of remote systems. We are going to start out easy and send different types of TCP packets with different flags set.

Hping2 is relatively easy to install on any *nix system. Go to the website and download it or use wget. Once it's downloaded you can issue the configure, make & make install commands to compile and install the program. Once it's installed you will see that hping2 has a ton of options. You can see them by issuing the man hping2 or hping2 –help command. I won't promise we'll go through them all but we are going to try.

Using Hping2 to Craft TCP Packets

Crafting TCP packets is the default behavior of Hping. By specifying the TCP flags, a destination port and a target IP address, one can easily construct TCP packets.

```
-F --fin set FIN flag
-S --syn set SYN flag
-R --rst set RST flag
-P --push set PUSH flag
-A --ack set ACK flag
-U --urg set URG flag
-X --xmas set X unused flag (0x40)
-Y --ymas set Y unused flag (0x80)
```

Before we start throwing packets all over your lab network, you should be aware that when you do not specify a destination port on the targeted computer it will default to 0. Also if you do not specify a source port it will use a random ephemeral port and go up numerically from there. P.S. I am going to use TCPDUMP to view the output of the hping2 packets/scans. If it's a bunch of nonsense to you, I recommend you learn TCPDUMP basics (use Google).

-S (SYN) Packet

The first packet we are going to send is the –S Syn packet. The attacker computer is 192.168.0.105 and the computer we are attacking is 192.168.0.100.

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -S 192.168.0.100
HPING 192.168.0.100 (eth0 192.168.0.100): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=128 id=18414 sport=0 flags=RA seq=0 win=0 rtt=149.9
ms
len=46 ip=192.168.0.100 ttl=128 id=18416 sport=0 flags=RA seq=1 win=0 rtt=0.5
ms
len=46 ip=192.168.0.100 ttl=128 id=18417 sport=0 flags=RA seq=2 win=0 rtt=0.4
ms
len=46 ip=192.168.0.100 ttl=128 id=18418 sport=0 flags=RA seq=3 win=0 rtt=0.5
ms
len=46 ip=192.168.0.100 ttl=128 id=18420 sport=0 flags=RA seq=4 win=0 rtt=1.6
ms
--- 192.168.0.100 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.4/30.6/149.9 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump tcp -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
14:19:22.506194 IP 192.168.0.105.2690 > 192.168.0.100.0: S
729051484:729051484(0) win 512
        0x0000:  4500 0028 f5e2 0000 4006 02d0 c0a8 0069  E..(....@......i
        0x0010:  c0a8 0064 0a82 0000 2b74 715c 00ee aed9  ...d....+tq\....
        0x0020:  5002 0200 d4aa 0000                      P.......
```

```
14:19:23.649879 IP 192.168.0.105.2691 > 192.168.0.100.0: S
1045497134:1045497134(0) win 512
        0x0000:  4500 0028 09bb 0000 4006 eef7 c0a8 0069  E..(....@......i
        0x0010:  c0a8 0064 0a83 0000 3e51 052e 34a4 7513  ...d....>Q..4.u.
        0x0020:  5002 0200 340b 0000                      P...4...
14:19:24.649886 IP 192.168.0.105.2692 > 192.168.0.100.0: S
734408221:734408221(0) win 512
        0x0000:  4500 0028 79cb 0000 4006 7ee7 c0a8 0069  E..(y...@.~....i
        0x0010:  c0a8 0064 0a84 0000 2bc6 2e1d 1432 0224  ...d....+....2.$
        0x0020:  5002 0200 b107 0000                      P.......

-----SNIP---------
10 packets captured
10 packets received by filter
0 packets dropped by kernel
[root@localhost root]#
```

As you can see in blue, hping2 picked an arbitrary port, in this case 2690, and incremented by one each time. In orange is the target port of 0 on the remote system which stays 0 since we did not specify a destination port. We can tell that is a SYN packet by seeing the S in red. Additionally, I received ACKs back from the 192.168.0.100 machine but edited those out here. That explains why in the hping2 output I sent 5 packets and received 5 packets. They were ACKs to my SYN packets.

Sending a SYN packet by the initiating system is the first step in the TCP/IP 3 way handshake. The next step is for the replying computer to send back a SYN/ACK packet, and finally an ACK packet to complete the handshake process.

The SYN (Steath) Scan is one of the most common scans used by port scanners. When the scan was initially being used it was considered stealthy because connections were not logged if they did not complete the 3 way handshake process. This has sense been long remedied and most common Intrusion Detection Systems will alert on SYN Scans.

-R (RST) Packet

The next packet we are going to send is the –R Reset (RST) packet. The reset packet is used to reset a connection. As you can see the command syntax is very similar. The only change is in the actual switch itself. Instead of -S it is -R.

"The RST packet is often used to perform what is known as inverse mapping. What this means is that RST packets are sent out and the response received is what will tell you if the host exists or not. If you send out a RST scan you would get one of two things. You will either get no response which indicates to you that the host is probably alive or you'll receive an ICMP host unreachable message. This would indicate that the host does not exist. This is what is known as inverse mapping." [1]

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -R 192.168.0.100
HPING 192.168.0.100 (eth0 192.168.0.100): R set, 40 headers + 0 data bytes
```

[1] http://www.windowsecurity.com/articles/Packet-analysis-tools-methodology-Part1.html


```
--- 192.168.0.100 hping statistic ---
6 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump tcp -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
13:52:02.992694 IP 192.168.0.105.2894 > 192.168.0.100.0: R
843167096:843167096(0) win 512
        0x0000:  4500 0028 8689 0000 4006 7229 c0a8 0069  E..(....@.r)...i
        0x0010:  c0a8 0064 0b4e 0000 3241 b578 14bc b5a8  ...d.N..2A.x....
        0x0020:  5004 0200 6e56 0000                      P...nV..
13:52:04.009817 IP 192.168.0.105.2895 > 192.168.0.100.0: R
378615428:378615428(0) win 512
        0x0000:  4500 0028 d259 0000 4006 2659 c0a8 0069  E..(.Y..@.&Y...i
        0x0010:  c0a8 0064 0b4f 0000 1691 3684 60ba 0a6b  ...d.O....6.`..k
        0x0020:  5004 0200 6839 0000                      P...h9..
13:52:05.010133 IP 192.168.0.105.2896 > 192.168.0.100.0: R
1069416179:1069416179(0) win 512
        0x0000:  4500 0028 5cd5 0000 4006 9bdd c0a8 0069  E..(\...@......i
        0x0010:  c0a8 0064 0b50 0000 3fbd fef3 51ed 7a0f  ...d.P..?...Q.z.
        0x0020:  5004 0200 15c5 0000                      P.......
13:52:06.009702 IP 192.168.0.105.2897 > 192.168.0.100.0: R
1038765926:1038765926(0) win 512
        0x0000:  4500 0028 f915 0000 4006 ff9c c0a8 0069  E..(....@......i
        0x0010:  c0a8 0064 0b51 0000 3dea 4f66 641a 6926  ...d.Q..=.Ofd.i&
        0x0020:  5004 0200 c5e0 0000                      P.......
---SNIP---
6 packets captured
6 packets received by filter
0 packets dropped by kernel
[root@localhost root]#
```

-F (FIN) Packet

The FIN packet is used to close an established connection. It is also used to conduct a FIN Scan. When a closed port receives a FIN packet, it should respond with a RST packet while an open port should do nothing (ignore the packet).

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -F 192.168.0.100
HPING 192.168.0.100 (eth0 192.168.0.100): F set, 40 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=128 id=20173 sport=0 flags=RA seq=0 win=0 rtt=34.2
ms
len=46 ip=192.168.0.100 ttl=128 id=20174 sport=0 flags=RA seq=1 win=0 rtt=0.8
ms
len=46 ip=192.168.0.100 ttl=128 id=20175 sport=0 flags=RA seq=2 win=0 rtt=0.5
ms
len=46 ip=192.168.0.100 ttl=128 id=20176 sport=0 flags=RA seq=3 win=0 rtt=0.5
ms
--- 192.168.0.100 hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.5/9.0/34.2 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump tcp -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
14:47:52.920308 IP 192.168.0.105.1416 > 192.168.0.100.0: F
1501065776:1501065776(0) win 512
        0x0000:  4500 0028 a604 0000 4006 52ae c0a8 0069  E..(....@.R....i
        0x0010:  c0a8 0064 0588 0000 5978 7230 4472 964e  ...d....Yxr0Dr.N
        0x0020:  5001 0200 7fd4 0000                      P.......
14:47:52.922503 IP 192.168.0.100.0 > 192.168.0.105.1416: R 0:0(0) ack
1501065777 win 0
        0x0000:  4500 0028 4ecd 0000 8006 69e5 c0a8 0064  E..(N.....i....d
        0x0010:  c0a8 0069 0000 0588 0000 0000 5978 7231  ...i........Yxr1
        0x0020:  5014 0000 5c81 0000 0000 0000 0000       P...\.........
14:47:53.950386 IP 192.168.0.105.1417 > 192.168.0.100.0: F
378133699:378133699(0) win 512
        0x0000:  4500 0028 e2c8 0000 4006 15ea c0a8 0069  E..(....@......i
        0x0010:  c0a8 0064 0589 0000 1689 dcc3 42ae d092  ...d........B...
        0x0020:  5001 0200 1faf 0000                      P.......
14:47:53.950837 IP 192.168.0.100.0 > 192.168.0.105.1417: R 0:0(0) ack 378133700
win 0
        0x0000:  4500 0028 4ece 0000 8006 69e4 c0a8 0064  E..(N.....i....d
        0x0010:  c0a8 0069 0000 0589 0000 0000 1689 dcc4  ...i............
        0x0020:  5014 0000 34dc 0000 0000 0000 0000       P...4.........
14:47:54.950227 IP 192.168.0.105.1418 > 192.168.0.100.0: F
716278911:716278911(0) win 512
        0x0000:  4500 0028 3a33 0000 4006 be7f c0a8 0069  E..(:3..@......i
        0x0010:  c0a8 0064 058a 0000 2ab1 8c7f 072d 4ef8  ...d....*....-N.
        0x0020:  5001 0200 18e6 0000                      P.......
14:47:54.950539 IP 192.168.0.100.0 > 192.168.0.105.1418: R 0:0(0) ack 716278912
win 0
        0x0000:  4500 0028 4ecf 0000 8006 69e3 c0a8 0064  E..(N.....i....d
        0x0010:  c0a8 0069 0000 058a 0000 0000 2ab1 8c80  ...i........*...
        0x0020:  5014 0000 70f7 0000 0000 0000 0000       P...p.........
14:47:55.950485 IP 192.168.0.105.1419 > 192.168.0.100.0: F
453633263:453633263(0) win 512
        0x0000:  4500 0028 a536 0000 4006 537c c0a8 0069  E..(.6..@.S|...i
        0x0010:  c0a8 0064 058b 0000 1b09 e4ef 16f3 9998  ...d............
        0x0020:  5001 0200 75b6 0000                      P...u...
14:47:55.950800 IP 192.168.0.100.0 > 192.168.0.105.1419: R 0:0(0) ack 453633264
win 0
        0x0000:  4500 0028 4ed0 0000 8006 69e2 c0a8 0064  E..(N.....i....d
        0x0010:  c0a8 0069 0000 058b 0000 0000 1b09 e4f0  ...i............
        0x0020:  5014 0000 282e 0000 0000 0000 0000       P...(.........

8 packets captured
8 packets received by filter
0 packets dropped by kernel
[root@localhost root]#
```

You can see from the TCPDUMP output that the attacker computer 192.168.0.105 sends a FIN packet to 192.168.0.100, and in turn, because we are sending the packet to a most likely closed port, port 0, it returns a RST packet back. Most documentation will tell you that this scan usually doesn't work anymore due to patching and whatnot, but the 192.168.0.100 computer is an XP Professional SP2 fully patched machine only. The firewall is completely turned off.

For giggles I tried to send FIN packets to ports I knew were open on the box, ports 135 & 445, and received RST back as well. So I guess that tells us something about the reliability of the scan against a Windows XP Box.

I am going to show the output of the scan here, but we will cover destination ports later so don't worry too much about it now.

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -F 192.168.0.100 -p 135
HPING 192.168.0.100 (eth0 192.168.0.100): F set, 40 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=128 id=22178 sport=135 flags=RA seq=0 win=0
rtt=131.7 ms
len=46 ip=192.168.0.100 ttl=128 id=22181 sport=135 flags=RA seq=1 win=0 rtt=0.6
ms
len=46 ip=192.168.0.100 ttl=128 id=22182 sport=135 flags=RA seq=2 win=0 rtt=2.8
ms
len=46 ip=192.168.0.100 ttl=128 id=22183 sport=135 flags=RA seq=3 win=0 rtt=0.9
ms
--- 192.168.0.100 hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.6/34.0/131.7 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump tcp -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
15:17:04.795615 IP 192.168.0.105.2805 > 192.168.0.100.135: F
879656640:879656640(0) win 512
        0x0000:  4500 0028 9418 0000 4006 649a c0a8 0069  E..(....@.d....i
        0x0010:  c0a8 0064 0af5 0087 346e 7ec0 63a0 baf6  ...d....4n~.c...
        0x0020:  5001 0200 4e84 0000                      P...N...
15:17:04.797291 IP 192.168.0.100.135 > 192.168.0.105.2805: R 0:0(0) ack
879656641 win 0
        0x0000:  4500 0028 56a2 0000 8006 6210 c0a8 0064  E..(V.....b....d
        0x0010:  c0a8 0069 0087 0af5 0000 0000 346e 7ec1  ...i........4n~.
        0x0020:  5014 0000 6f07 0000 0000 0000 0000       P...o.........
15:17:05.922394 IP 192.168.0.105.2806 > 192.168.0.100.135: F
1281421513:1281421513(0) win 512
        0x0000:  4500 0028 fb96 0000 4006 fd1b c0a8 0069  E..(....@......i
        0x0010:  c0a8 0064 0af6 0087 4c60 f0c9 349c 85a8  ...d....L`..4...
        0x0020:  5001 0200 28da 0000                      P...(...
15:17:05.922708 IP 192.168.0.100.135 > 192.168.0.105.2806: R 0:0(0) ack
1281421514 win 0
        0x0000:  4500 0028 56a5 0000 8006 620d c0a8 0064  E..(V.....b....d
        0x0010:  c0a8 0069 0087 0af6 0000 0000 4c60 f0ca  ...i........L`..
        0x0020:  5014 0000 e50a 0000 0000 0000 0000       P.............
---SNIP---
8 packets captured
8 packets received by filter
0 packets dropped by kernel
[root@localhost root]#
```

ICMP Packets

Most ping programs use ICMP echo requests and wait for echo replies to come back to test connectivity. Hping2 allows us to do the same testing using any IP packet, including ICMP, UDP, and TCP. This can be helpful since nowadays most firewalls or routers block ICMP. Hping2, by default, will use TCP, but, if you still want to send an ICMP scan, you can. We send ICMP scans using the -1 (one) mode. Basically the syntax will be `hping2 -1 IPADDRESS`

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -1 192.168.0.100
HPING 192.168.0.100 (eth0 192.168.0.100): icmp mode set, 28 headers + 0 data
bytes
len=46 ip=192.168.0.100 ttl=128 id=27118 icmp_seq=0 rtt=14.9 ms
len=46 ip=192.168.0.100 ttl=128 id=27119 icmp_seq=1 rtt=0.5 ms
len=46 ip=192.168.0.100 ttl=128 id=27120 icmp_seq=2 rtt=0.5 ms
len=46 ip=192.168.0.100 ttl=128 id=27121 icmp_seq=3 rtt=1.5 ms
len=46 ip=192.168.0.100 ttl=128 id=27122 icmp_seq=4 rtt=0.9 ms
--- 192.168.0.100 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.5/3.7/14.9 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump -X
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 68 bytes
15:44:16.052016 IP 192.168.0.105 > 192.168.0.100: icmp 8: echo request seq 0
        0x0000:  4500 001c 5161 0000 4001 a762 c0a8 0069  E...Qa..@..b...i
        0x0010:  c0a8 0064 0800 8bb6 6c49 0000            ...d....lI..
15:44:16.052673 IP 192.168.0.100 > 192.168.0.105: icmp 8: echo reply seq 0
        0x0000:  4500 001c 69ee 0000 8001 4ed5 c0a8 0064  E...i.....N....d
        0x0010:  c0a8 0069 0000 93b6 6c49 0000 0000 0000  ...i....lI......
        0x0020:  0000 0000 0000 0000 0000 0000 0000       ..............
```

UDP Packets

Like I already mentioned, the default protocol for hping2 is the TCP. But just like with ICMP, if you want to send a UDP packet you can with hping2. We send UDP scans using the -2 (two) mode. Basically the syntax will be `hping2 -2 IPADDRESS`. UDP Scans can be useful when probing UDP services like NETBIOS, NFS, DNS, & NIS.

Hping2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -2 192.168.0.100
HPING 192.168.0.100 (eth0 192.168.0.100): udp mode set, 28 headers + 0 data
bytes
ICMP Port Unreachable from ip=192.168.0.100 name=UNKNOWN
ICMP Port Unreachable from ip=192.168.0.100 name=UNKNOWN
ICMP Port Unreachable from ip=192.168.0.100 name=UNKNOWN
ICMP Port Unreachable from ip=192.168.0.100 name=UNKNOWN
ICMP Port Unreachable from ip=192.168.0.100 name=UNKNOWN
--- 192.168.0.100 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump udp -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
15:55:32.164563 IP 192.168.0.105.2356 > 192.168.0.100.0: UDP, length: 0
        0x0000:  4500 001c 0b98 0000 4011 ed1b c0a8 0069  E.......@......i
        0x0010:  c0a8 0064 0934 0000 0008 748c            ...d.4....t.
15:55:33.190960 IP 192.168.0.105.2357 > 192.168.0.100.0: UDP, length: 0
        0x0000:  4500 001c bc4e 0000 4011 3c65 c0a8 0069  E....N..@.<e...i
        0x0010:  c0a8 0064 0935 0000 0008 748b            ...d.5....t.
15:55:34.192154 IP 192.168.0.105.2358 > 192.168.0.100.0: UDP, length: 0
        0x0000:  4500 001c 7f81 0000 4011 7932 c0a8 0069  E.......@.y2...i
        0x0010:  c0a8 0064 0936 0000 0008 748a            ...d.6....t.
15:55:35.190593 IP 192.168.0.105.2359 > 192.168.0.100.0: UDP, length: 0
        0x0000:  4500 001c 3a9c 0000 4011 be17 c0a8 0069  E...:...@......i
        0x0010:  c0a8 0064 0937 0000 0008 7489            ...d.7....t.
15:55:36.190661 IP 192.168.0.105.2360 > 192.168.0.100.0: UDP, length: 0
        0x0000:  4500 001c 6faa 0000 4011 8909 c0a8 0069  E...o...@......i
        0x0010:  c0a8 0064 0938 0000 0008 7488            ...d.8....t.
[root@localhost root]#
```

-S SYN Scan and Specifying Ports

Now we are going to start seeing the power of hping2 a little more. We are going to direct a SYN packet at a specified port, in this case port 135. To send a SYN packet at a specific port requires a few more switches. We are going to send a SYN (-S) packet to 192.168.0.100 specifically on port 135 by putting in the (-p) switch. The –p switch allows you to specify the destination port. To specify the source port on your machine you want the packet to go out on, you would use the -s switch followed by a port number just as the destination port example below.

HPING2 INPUT:

```
[root@localhost hping2-rc3]# hping2 -S 192.168.0.100 -p 135
HPING 192.168.0.100 (eth0 192.168.0.100): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.100 ttl=128 DF id=28733 sport=135 flags=SA seq=0 win=16616
rtt=122.8 ms
len=46 ip=192.168.0.100 ttl=128 DF id=28734 sport=135 flags=SA seq=1 win=16616
rtt=11.7 ms
len=46 ip=192.168.0.100 ttl=128 DF id=28737 sport=135 flags=SA seq=2 win=16616
rtt=1.4 ms
len=46 ip=192.168.0.100 ttl=128 DF id=28738 sport=135 flags=SA seq=3 win=16616
rtt=1.5 ms
--- 192.168.0.100 hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.4/34.3/122.8 ms
[root@localhost hping2-rc3]#
```

TCPDUMP OUTPUT:

```
[root@localhost root]# tcpdump tcp  -X -s 1514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
16:09:12.059187 IP 192.168.0.105.1839 > 192.168.0.100.135: S
15960697:15960697(0) win 512
```

```
        0x0000:  4500 0028 596b 0000 4006 9f47 c0a8 0069  E..(Yk..@..G...i
        0x0010:  c0a8 0064 072f 0087 00f3 8a79 3a64 1ef1  ...d./.....y:d..
        0x0020:  5002 0200 3f4d 0000                      P...?M..
16:09:12.061047 IP 192.168.0.100.135 > 192.168.0.105.1839: S
1298117721:1298117721(0) ack 15960698 win 16616 <mss 1460>
        0x0000:  4500 002c 703d 4000 8006 0871 c0a8 0064  E..,p=@....q...d
        0x0010:  c0a8 0069 0087 072f 4d5f b459 00f3 8a7a  ...i.../M_.Y...z
        0x0020:  6012 40e8 4034 0000 0204 05b4 0000       `.@.@4........
16:09:12.069235 IP 192.168.0.105.1839 > 192.168.0.100.135: R
15960698:15960698(0) win 0
        0x0000:  4500 0028 0000 4000 4006 b8b2 c0a8 0069  E..(..@.@......i
        0x0010:  c0a8 0064 072f 0087 00f3 8a7a 0000 0000  ...d./.....z....
        0x0020:  5004 0000 9a9f 0000                      P.......
[root@localhost root]#
```

An open port is indicated by a SA return packet (see the hping2 input), closed ports by a RA packet (see the other hping2 input where we sent the packet to port 0). Remember the TCP 3-way handshake! In this case the 192.168.0.100 computer responded with a SYN-ACK and the attacker computer responded with a RST to end the connection.

Now that you are starting to see the possibilities of crafting your own custom packets with hping2, it's time for you to expand on the knowledge you have just acquired. Test out hping2 on your own and start to think creatively about ways in which this versatile tool can be used. Then you will be ready for the more advanced tutorial that will be arriving within the next few months.

References:

Network Intrusion Detection by Stephen Northcutt

Packet Analysis Tools and methodology part 1 by Don Parker: http://www.windowsecurity.com/articles/Packet-analysis-tools-methodology-Part1.html

Hping_conv.pdf by Don Parker. Available at: www.hping.org/hping_conv.pdf.

This tutorial is very much based on Don Parker's paper. I didn't like that he didn't show the hping output or explain very much on the tcpdump output so I attempted to improve upon his work by showing the output of the hping2 commands and going into more detail with the tcpdump output.

About the Author

Chris Gates, CISSP, CPTS, CEH is the operations manager for www.LearnSecurityOnline.com. His computer security interests are in Windows and Web Application security. He also serves a student mentor and course developer for LSO. Feel free to email comments and suggestions on the paper to chris [at] learnsecurityonline [dot] com.