

# HPING2 HOWTO

## Salvatore Sanfilippo

N.B.: this HOWTO is not completed and in some points very silly. I leave this here only because maybe it's better than nothing.

### Changes Log

Aug 7 1999 vi HPING2-HOWTO.txt  
Aug 8 1999 \_\_0000, \_\_0001, \_\_0002, \_\_0003  
Aug 10 1999 \_\_0004

### Index

[search \_\_XXXX in order to jump to point you want]

- \_\_0000: Copyright notice
- \_\_0001: What is hping?
- \_\_0002: What i need to know about TCP/IP in order to use hping?
- \_\_0003: First step with hping
- \_\_0004: IP id and how to scan TCP ports using spoofing.
- \_\_0005: How to test firewall rules. (TODO)
- \_\_0006: How to trasfer files accross firewall. (TODO)
- \_\_000A: hping usage example (TODO)

### **0000: Copyright notice, License, and all that stuff**

Copyright (C) Salvatore Sanfilippo, 1999.

Permission is granted to make and distribute copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the derived work is distributed under the terms of a permission notice identical to this one. Translations fall under the category of modified versions."

Warranty: None.

Recommendations: Commercial redistribution is allowed and encouraged; however, it is strongly recommended that the redistributor contact the author before the redistribution, in the interest of keeping things up-to-date (you could send me a copy of the thing you're making while you're at it). Translators are also advised to contact the author before translating. The printed version looks nicer. Recycle.

### **0001: What is hping?**

Hping is a software to do TCP/IP stack auditing, to uncover firewall policy, to scan TCP port in a lot of different modes, to transfer files accross a firewall and many other stuff. Using hping you are able to do even a lot of not security-regarding stuff. For example you can test networks performance, check if a host is up, check if TOS is handled et cetera.

### **0002: What i need to know about TCP/IP in order to use hping?**

If you know TCP/IP you will find hping very usefull, otherwise you can use hping only to do well known tests. See \_\_000A for some example.

### **0003: First step with hping**

The simplest usage of hping is the following:

```
#hping host
```

## HPING2 HOWTO

Salvatore Sanfilippo

This command sends a TCP null-flags packet to port 0 of target host every second and show the host replies. For example:

```
# hping www.debian.org
ppp0 default routing interface selected (according to /proc)
HPING www.debian.org (ppp0 209.81.8.242): NO FLAGS are set, 40 headers + 0 data bytes
40 bytes from 209.81.8.242: flags=RA seq=0 ttl=243 id=63667 win=0 time=369.4 ms
40 bytes from 209.81.8.242: flags=RA seq=1 ttl=243 id=63719 win=0 time=420.0 ms
40 bytes from 209.81.8.242: flags=RA seq=2 ttl=243 id=63763 win=0 time=350.0 ms
[Ctrl+C]

--- www.debian.org hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
```

As you can see host replies with a TCP packet with RST and ACK flags set. So you are able to perform a 'TCP ping', usefull when ICMPs are filtered. By default port 0 are used because it's very strange that is in LISTEN state. If we send a TCP null-flags to a port in LISTEN state a lot of TCP/IP stack will not send any reply. So we are able to know if a port is in LISTEN state. For example:

```
# hping www.debian.org -p 80
ppp0 default routing interface selected (according to /proc)
HPING www.debian.org (ppp0 209.81.8.242): NO FLAGS are set, 40 headers + 0 data bytes
[Ctrl+C]

--- www.debian.org hping statistic ---
5 packets trasmitted, 0 packets received, 100% packet loss
```

Since port 80 of www.debian.org is in LISTEN mode we got no response.

But What's happen if we try to hping a firewalled port? This depends on firewall policy/implementation. Usually we get an ICMP or nothing. For example:

```
# hping www.yahoo.com -p 79
ppp0 default routing interface selected (according to /proc)
HPING www.yahoo.com (ppp0 204.71.200.67): NO FLAGS are set, 40 headers + 0 data bytes
ICMP Packet filtered from 206.132.254.41 (pos1-0-2488M.hr8.SNV.globalcenter.net)

--- www.yahoo.com hping statistic ---
14 packets tramitted, 0 packets received, 100% packet loss
```

yahoo firewall doesn't allow connection to port 79, so reply with an ICMP Packet filtered (ICMP unreachable code 13). However there are a lot of firewall that simply drop the packet. For example:

```
# hping www.microsoft.com -p 79
ppp0 default routing interface selected (according to /proc)
HPING www.microsoft.com (ppp0 207.46.130.150): NO FLAGS are set, 40 headers + 0 data bytes

--- www.microsoft.com hping statistic ---
4 packets tramitted, 0 packets received, 100% packet loss
```

No reply from microsoft. Is the port firewalled or in LISTEN mode? To uncover this is very simply. Just we try to set ACK flag instead to send a TCP null-flag packet. If the host respond maybe this port is in LISTEN mode (but it's possible that there is a rules that deny null-flag TCP packet but allow ACK).

# HPING2 HOWTO

## Salvatore Sanfilippo

```
# hping www.microsoft.com -A -p 79
ppp0 default routing interface selected (according to /proc)
HPING www.microsoft.com (ppp0 207.46.130.149): A set, 40 headers + 0 data bytes

--- www.microsoft.com hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
```

No response again, So this port seems to be filtered. Anyway it's possible that microsoft is using an 'intelligent' firewall that know that in order to connect first I must send a SYN.

```
# hping www.microsoft.com -S -p 79
ppp0 default routing interface selected (according to /proc)
HPING www.microsoft.com (ppp0 207.46.130.149): S set, 40 headers + 0 data bytes

--- www.microsoft.com hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
```

Ok.. seems that port 79 of microsoft is really filtered. Just for clearness we send some ACK to port 80 of www.debian.org:

```
# hping www.debian.org -p 80 -A
ppp0 default routing interface selected (according to /proc)
HPING www.debian.org (ppp0 209.81.8.242): A set, 40 headers + 0 data bytes
40 bytes from 209.81.8.242: flags=R seq=0 ttl=243 id=5590 win=0 time=379.5 ms
40 bytes from 209.81.8.242: flags=R seq=1 ttl=243 id=5638 win=0 time=370.0 ms
40 bytes from 209.81.8.242: flags=R seq=2 ttl=243 id=5667 win=0 time=360.0 ms

--- www.debian.org hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
```

We can see replies even if port 80 is in LISTEN mode because a port in LISTEN mode may not replay only to NULL, FIN, Xmas, Ymas flags TCP packet. ACK and RST are two important TCP flags that allow to do ACL tests and to guess ip->id without to produce any log (usually).

### 0004: IP id and how to scan TCP ports using spoofing.

Every IP packet is identified by a 16 bit id. Thanks to this id IP stacks are able to handle fragmentation. A lot of OSs handle ip->id trivially: just increment by 1 this id for each packet sent. Using this id you are able at least to estimate hosts traffic and to scan with spoofed packets. OpenBSD >= 2.5 and many others implement a random not repetitive id so you aren't able to joke with ip->id. Win\* ip->id has different byte ordering, so you must specify --winid or -W option if you are using hping2 against Win\*.

N.B.: You are able to scan spoofed hosts with safe/random ip->id because in order to spoof your packets you need a third part host with incremental id rule but you don't need that target of your scanning has an incremental id.

How to estimate host traffic using ip->id? It's really simple:

```
# hping www.yahoo.com -p 80 -A
ppp0 default routing interface selected (according to /proc)
HPING www.yahoo.com (ppp0 204.71.200.74): A set, 40 headers + 0 data bytes
40 bytes from 204.71.200.74: flags=R seq=0 ttl=53 id=29607 win=0 rtt=329.4 ms
40 bytes from 204.71.200.74: flags=R seq=1 ttl=53 id=31549 win=0 rtt=390.0 ms
```

## HPING2 HOWTO

Salvatore Sanfilippo

```
40 bytes from 204.71.200.74: flags=R seq=2 ttl=53 id=33432 win=0 rtt=390.0 ms
40 bytes from 204.71.200.74: flags=R seq=3 ttl=53 id=35368 win=0 rtt=380.0 ms
40 bytes from 204.71.200.74: flags=R seq=4 ttl=53 id=37335 win=0 rtt=390.0 ms
40 bytes from 204.71.200.74: flags=R seq=5 ttl=53 id=39157 win=0 rtt=380.0 ms
40 bytes from 204.71.200.74: flags=R seq=6 ttl=53 id=41118 win=0 rtt=370.0 ms
40 bytes from 204.71.200.74: flags=R seq=7 ttl=53 id=43330 win=0 rtt=390.0 ms
```

```
--- www.yahoo.com hping statistic ---
8 packets tramitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 329.4/377.4/390.0 ms
```

As you can see id field increase. Packet with sequence 0 has id=29607, sequence 1 has id=31549, so www.yahoo.com host sent 31549-29607 = 1942 packets in circa one second. Using -r|--relid option hping output id field as difference between last and current received packet id.

```
# hping www.yahoo.com -P 80 -A -r
ppp0 default routing interface selected (according to /proc)
HPING www.yahoo.com (ppp0 204.71.200.68): A set, 40 headers + 0 data bytes
40 bytes from 204.71.200.68: flags=R seq=0 ttl=53 id=65179 win=0 rtt=327.1 ms
40 bytes from 204.71.200.68: flags=R seq=1 ttl=53 id=+1936 win=0 rtt=360.0 ms
40 bytes from 204.71.200.68: flags=R seq=2 ttl=53 id=+1880 win=0 rtt=340.0 ms
40 bytes from 204.71.200.68: flags=R seq=3 ttl=53 id=+1993 win=0 rtt=330.0 ms
40 bytes from 204.71.200.68: flags=R seq=4 ttl=53 id=+1871 win=0 rtt=350.0 ms
40 bytes from 204.71.200.68: flags=R seq=5 ttl=53 id=+1932 win=0 rtt=340.0 ms
40 bytes from 204.71.200.68: flags=R seq=6 ttl=53 id=+1776 win=0 rtt=330.0 ms
40 bytes from 204.71.200.68: flags=R seq=7 ttl=53 id=+1749 win=0 rtt=320.0 ms
40 bytes from 204.71.200.68: flags=R seq=8 ttl=53 id=+1888 win=0 rtt=340.0 ms
40 bytes from 204.71.200.68: flags=R seq=9 ttl=53 id=+1907 win=0 rtt=330.0 ms
```

```
--- www.yahoo.com hping statistic ---
10 packets tramitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 320.0/336.7/360.0 ms
```

Obviously checking the id every 1/2 second instead of 1 second, increment will be half.

```
# hping www.yahoo.com -P 80 -A -r -i u 500000
ppp0 default routing interface selected (according to /proc)
HPING www.yahoo.com (ppp0 204.71.200.68): A set, 40 headers + 0 data bytes
40 bytes from 204.71.200.68: flags=R seq=0 ttl=53 id=35713 win=0 rtt=327.0 ms
40 bytes from 204.71.200.68: flags=R seq=1 ttl=53 id=+806 win=0 rtt=310.0 ms
40 bytes from 204.71.200.68: flags=R seq=2 ttl=53 id=+992 win=0 rtt=320.0 ms
40 bytes from 204.71.200.68: flags=R seq=3 ttl=53 id=+936 win=0 rtt=330.0 ms
40 bytes from 204.71.200.68: flags=R seq=4 ttl=53 id=+987 win=0 rtt=310.0 ms
40 bytes from 204.71.200.68: flags=R seq=5 ttl=53 id=+952 win=0 rtt=320.0 ms
40 bytes from 204.71.200.68: flags=R seq=6 ttl=53 id=+918 win=0 rtt=330.0 ms
40 bytes from 204.71.200.68: flags=R seq=7 ttl=53 id=+809 win=0 rtt=320.0 ms
40 bytes from 204.71.200.68: flags=R seq=8 ttl=53 id=+881 win=0 rtt=320.0 ms
```

```
--- www.yahoo.com hping statistic ---
9 packets tramitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 310.0/320.8/330.0 ms
```

N.B. Warning, using ip->id you are able only to guess \*the number of packets sent/time\*. You can't always compare different hosts. ip->id refers to all host interfaces and for example if an host use NAT or ct TCP connections to another host (for example a firewall used to hide a web server) ip->id increment may result fakely increased.

# HPING2 HOWTO

## Salvatore Sanfilippo

hpinging windows box without using --winid option you will see as increments are 256 multiple because different id byteordering. This can be really usefull for OS fingerprinting:

```
#hping win95 -r
HPING win95 (eth0 192.168.4.41): NO FLAGS are set, 40 headers + 0 data bytes
46 bytes from 192.168.4.41: flags=RA seq=0 ttl=128 id=47371 win=0 rtt=0.5 ms
46 bytes from 192.168.4.41: flags=RA seq=1 ttl=128 id=+256 win=0 rtt=0.5 ms
46 bytes from 192.168.4.41: flags=RA seq=2 ttl=128 id=+256 win=0 rtt=0.6 ms
46 bytes from 192.168.4.41: flags=RA seq=3 ttl=128 id=+256 win=0 rtt=0.5 ms

--- win95 hping statistic ---
4 packets tramitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.6 ms
```

Windows systems are "marked", so in order to discovery if an host is a Windows host you need to send just some packet.

How to perform spoofed SYN scan using incremental id? The following is the original message to bugtraq about spoofed/indirect/idle scan method, bottom i'll try to explain details and how this is possible even with UDP with some restriction.

### ---- bugtraq posting about spoofed scanning ----

Hi,

I have uncovered a new tcp port scan method. Instead all others it allows you to scan using spoofed packets, so scanned hosts can't see your real address. In order to perform this i use three well known tcp/ip implementation peculiarities of most OS:

(1) \* hosts reply SYN|ACK to SYN if tcp target port is open, reply RST|ACK if tcp target port is closed.

(2) \* You can know the number of packets that hosts are sending using id ip header field. See my previous posting 'about the ip header' in this ml.

(3) \* hosts reply RST to SYN|ACK, reply nothing to RST.

The Players:

host A - evil host, the attacker.

host B - silent host.

host C - victim host.

A is your host. B is a particular host: It must not send any packets while you are scanning C. There are a lot of 'zero traffic' hosts in internet, especially in the night :) C is the victim, it must be vulnerable to SYN scan. I've called this scan method 'dumb host scan' in honour of host B characteristics.

How it works: Host A monitors number of outgoing packets from B using id iphdr. You can do this simply using hping:

```
#hping B -r
HPING B (eth0 xxx.yyy.zzz.jjj): no flags are set, 40 data bytes
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=0 ttl=64 id=41660 win=0 time=1.2 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=1 ttl=64 id=+1 win=0 time=75 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=2 ttl=64 id=+1 win=0 time=91 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=3 ttl=64 id=+1 win=0 time=90 ms
```

# HPING2 HOWTO

## Salvatore Sanfilippo

```
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=4 ttl=64 id=+1 win=0 time=91 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=5 ttl=64 id=+1 win=0 time=87 ms
-cut-
..
.
```

As you can see, id increases are always 1. So this host have the characteristics that host B should to own.

Now host A sends SYN to port X of C spoofing from B. (using hping => 0.67 is very easy, <http://www.kyuzz.org/antirez>) if port X of C is open, host C will send SYN|ACK to B (yes, host C don't know that the real sender is A). In this case host B replies to SYN|ACK with a RST. If we send to host C a few of SYN it will reply to B with a few of SYN|ACK, so B will reply to C a few of RST... so we'll see that host B is sending packets!

```
.
..
-cut-
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=17 ttl=64 id=+1 win=0 time=96 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=18 ttl=64 id=+1 win=0 time=80 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=19 ttl=64 id=+2 win=0 time=83 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=20 ttl=64 id=+3 win=0 time=94 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=21 ttl=64 id=+1 win=0 time=92 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=22 ttl=64 id=+2 win=0 time=82 ms
-cut-
..
.
```

The port is open!

Instead, if port X of C is closed sending to C a few of SYN spoofed from B, it will reply with RST to B, and B will not reply (see 3). So we'll see that host B is not sending any packet:

```
.
..
-cut-
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=52 ttl=64 id=+1 win=0 time=85 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=53 ttl=64 id=+1 win=0 time=83 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=54 ttl=64 id=+1 win=0 time=93 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=55 ttl=64 id=+1 win=0 time=74 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=56 ttl=64 id=+1 win=0 time=95 ms
60 bytes from xxx.yyy.zzz.jjj: flags=RA seq=57 ttl=64 id=+1 win=0 time=81 ms
-cut-
..
.
```

The port is closed.

All this can appear complicated to perform, but using two sessions of hping on Linux virtual consoles or under X makes it more simple. First session listen host B: hping B -r Second session send spoofed SYN: hping C -a B -S

Sorry if my english is not so clear. However this posting is not adequate to describe exhaustively this scan method, so i'll write a paper on this topic, specially about how to implement this in a port scanner (i.e. nmap), and about players characteristics and OS used.

## HPING2 HOWTO

Salvatore Sanfilippo

happy new year,  
antirez

----- EOF -----

As you can see spoofed scanning is trivial to perform, especially using hping2 you are able to specify micro seconds interval (-i uX) so you don't need that B host is a totally idle host. You may read id increment once every second sending 10 SYN every second. If you send an adequate SYNnumber/second expected id increment is so big that you are able to see if port is open or closed even if B host is sending other packets. Example:

```
# hping awake.host.org -p 80 -A -r
ppp0 default routing interface selected (according to /proc)
HPING server.alicom.com (ppp0 111.222.333.44): A set, 40 headers + 0 data bytes
40 bytes from 111.222.333.44: flags=R seq=0 ttl=249 id=47323 win=0 rtt=239.7 ms
40 bytes from 111.222.333.44: flags=R seq=1 ttl=249 id=+6 win=0 rtt=630.0 ms
40 bytes from 111.222.333.44: flags=R seq=2 ttl=249 id=+6 win=0 rtt=280.0 ms
40 bytes from 111.222.333.44: flags=R seq=3 ttl=249 id=+8 win=0 rtt=340.0 ms
40 bytes from 111.222.333.44: flags=R seq=4 ttl=249 id=+5 win=0 rtt=440.0 ms
40 bytes from 111.222.333.44: flags=R seq=5 ttl=249 id=+5 win=0 rtt=410.0 ms
40 bytes from 111.222.333.44: flags=R seq=6 ttl=249 id=+8 win=0 rtt=1509.9 ms
40 bytes from 111.222.333.44: flags=R seq=7 ttl=249 id=+4 win=0 rtt=1460.0 ms
40 bytes from 111.222.333.44: flags=R seq=8 ttl=249 id=+7 win=0 rtt=770.0 ms
40 bytes from 111.222.333.44: flags=R seq=9 ttl=249 id=+5 win=0 rtt=230.0 ms
...
```

as you can see this host isn't in idle, it sends ~ 6 packets every second. Now scan www.yahoo.com's port 80 to see if it's open:

```
root.1# hping -a server.alicom.com -S -p 80 -i u10000 www.yahoo.com
ppp0 default routing interface selected (according to /proc)
HPING www.yahoo.com (ppp0 204.71.200.74): S set, 40 headers + 0 data bytes
```

[wait some second and press CTRL+C]

```
--- www.yahoo.com hping statistic ---
130 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Looking output of 'hping awake.host.org -p 80 -A -r' it's simple to understand that www.yahoo.com's port 80 is open:

```
40 bytes from 111.222.333.44: flags=R seq=59 ttl=249 id=+16 win=0 rtt=380.0 ms
40 bytes from 111.222.333.44: flags=R seq=60 ttl=249 id=+75 win=0 rtt=850.0 ms
40 bytes from 111.222.333.44: flags=R seq=61 ttl=249 id=+12 win=0 rtt=1050.0 ms
40 bytes from 111.222.333.44: flags=R seq=62 ttl=249 id=+1 win=0 rtt=450.0 ms
40 bytes from 111.222.333.44: flags=R seq=63 ttl=249 id=+27 win=0 rtt=230.0 ms
40 bytes from 111.222.333.44: flags=R seq=64 ttl=249 id=+11 win=0 rtt=850.0 ms
```

note that  $16+75+12+27+11+1-6 = 136$  and that we sent 130 packets. So it's very realistic that increments are produced by our packtes.

Tips: Using an idle host to perform spoofed scanning it's usefull to output only replies that show an increment != 1. Try `hping host -r | grep -v "id=+1"'