

Getting Started with Hping3

unknown

This document is a quick introduction to hping3. hping3 is mostly command line compatible with hping2 so the command line interface is not documented in this document. Instead this is an introduction to the hping Tcl scripting capabilities, and how to use them interactively and in standalone scripts.

Important Note: to get the best of hping3 you should learn some basic Tcl programming. To make the task more simple I'm writing a book about Tcl programming, the first nine chapters (all you need to start with Tcl IMHO) are online for free here: <http://www.invece.org/tclwise/>.

First Steps

First of all you need a working hping3 installation. Go to the download page, and download the latest hping3 tar.gz available. Install it, and log in as the root user (you need this to send and receive raw packets).

To enter the hping3 interactive shell, just type:

```
# hping3
```

without any argument. If hping was compiled with Tcl scripting capabilities you should see a prompt. The prompt will accept any Tcl command, it's actually a Tcl shell, what's special about it is that there is a new command called hping, and support for big numbers using commands like +, -, and so on.

As first try, you can type a simple command and see the result:

```
hping3.0.0-alpha-1> hping resolve www.google.com
66.102.9.104
```

The hping command should be called with a subcommand as a first argument (resolve in the example) and additional arguments according to the particular subcommand. The hping resolve command is used to convert a hostname to an IP address.

Ok, that's the basic usage. Now we can start to try more advanced commands (you can find a complete list of commands in the hping3 API page). For example the hping send command can send TCP/IP packets that you can easily describe as strings:

```
hping3.0.0-alpha-1> hping send {ip(daddr=192.168.1.8)+icmp(type=8,code=0)}
```

This command means "send an ICMP echo request packet to 192.168.1.8". Many details of the packet can be omitted. For example we didn't specify our source address (that will default to the real source address of the sender, the one of the outgoing interface), nor the IP or ICMP checksum. hping will compute them for us.

Let's check what tcpdump running at 192.168.1.8 detected:

```
tcpdump: listening on eth0
19:09:16.556695 192.168.1.6 > 192.168.1.8: icmp: echo request [ttl 0]
19:09:16.556803 192.168.1.8 > 192.168.1.6: icmp: echo reply
```

Getting Started with Hping3

unknown

Our ICMP packet reached the destination, that kindly replied with an ICMP echo reply packet. It's better to recall for a second the previous command, to analyze it better:

```
hping3.0.0-alpha-1> hping send {ip(daddr=192.168.1.8)+icmp(type=8,code=0)}
```

As you can see, there are { and } surrounding the packet description. This is required by Tcl in order to quote the string so that special characters will not be interpreted. Quoting with {} in Tcl is just like to quote with "" in most other languages, with the difference that no escapes are recognized inside {} quoting.

The second thing to note is the format we used to describe the packet. That's called APD, and was introduced with hping3 itself. The APD syntax is trivial, and there is a simple way to figure how to generate a given packet, because hping3 use this format to send packets, but also to receive packets as we will see in a moment.

TCL Inside

Before to show how it's possible to receive packets, I want to stress the fact that we are inside a Tcl interpreter, so we can use any of the Tcl abilities in hping scripts.

The following hping script will send the same ICMP packet we already sent to 192.168.1.8, but using different TTL values, from 5 to 10.

```
foreach i [list 5 6 7 8 9 10] {  
    hping send "ip(daddr=192.168.1.8,ttl=$i)+icmp(type=8,code=0)"  
}
```

With scripts longer then one line it can be a good idea to write the script with a text editor, and then run it using hping:

```
# hping exec foo.htcl
```

Cut and paste it into the hping interactive shell also works well.

Note that because this example uses a variable i to increment the ttl value on every iteration of the foreach, we used "" rather than {} quoting so that \$i would be expanded to the value of i.

I think it's clear now that in order to make a good use of hping3 you need to learn the Tcl language. The good news are that Tcl is a very powerful language, but it's very easy to learn, and if you learn Tcl you will enjoy it in many different tasks related or not to hping. The best site about Tcl is the TcLer's Wiki.

Packet reception

Another very important subcommand of hping is hping recv, that is used to capture packets from the specified interface. The simplest usage is the following:

Getting Started with Hping3

unknown

```
hping3.0.0-alpha-1> hping recv eth0
ip(ihl=0x5,ver=0x4,tos=0x00,totlen=52,id=42833,fragoff=0,mf=0,df=1,rf=0,ttl=5
4,proto=6,cksum=0xd53a,saddr=192.106.224.132,daddr=192.168.1.6)+\
tcp(sport=6667,dport=52466,seq=2163829654,ack=3105171942,x2=0x0,off=8,flags=a
,win=2848,cksum=0x99bd,urp=0)+\
tcp.nop()+tcp.nop()+tcp.timestamp(val=181365875,ecr=104872758)
```

Because the received packet description is too long I added newlines quoted with \, but actually hping will read the packet as a unique string.

hping recv returns a Tcl list, where every element is a packet (but by default it will be just one-element list).

At every call, hping recv eth0 will return the packet(s) in queue. If there is no packet to receive the command will block until one is available.

If you don't want hping recv to block forever, you can specify an additional argument. One more argument will tell hping the max number of packets to return in a single call. To learn the details please check the hping recv page in this wiki.

Note that the command always returns a Tcl list of packets, even when just one packet is returned. So if you want to use the returned packets you need to use Tcl list commands (as we will see in a moment). Another thing to note is that the packets are received in APD format, so it's possible to get a packet, possibly manipulate it, and resend the packet using hping send.

The following is an example script using hping recv.

```
while 1 {
    set p [lindex [hping recv eth0] 0]
    puts "[hping getfield ip saddr $p] -> [hping getfield ip ttl $p]"
}
```

The first line is just a while loop that will repeat the script provided as second argument forever. The second line, set p [lindex [hping recv eth0] 0] gets the next packet, the lindex command is used to extract the packet from the Tcl list (and the 0 argument tells lindex to get the first packet).

The second line of code, puts "...", print on the screen the source IP address and the TTL value of the packet. To extract fields from packets there is the command hping getfield (see the specific page for more information as usually).

If you execute this script, you'll get an output similar to the following:

```
# ./hping3 exec /tmp/test.tcl
192.168.1.6 -> 128
192.168.1.20 -> 128
```

Getting Started with Hping3

unknown

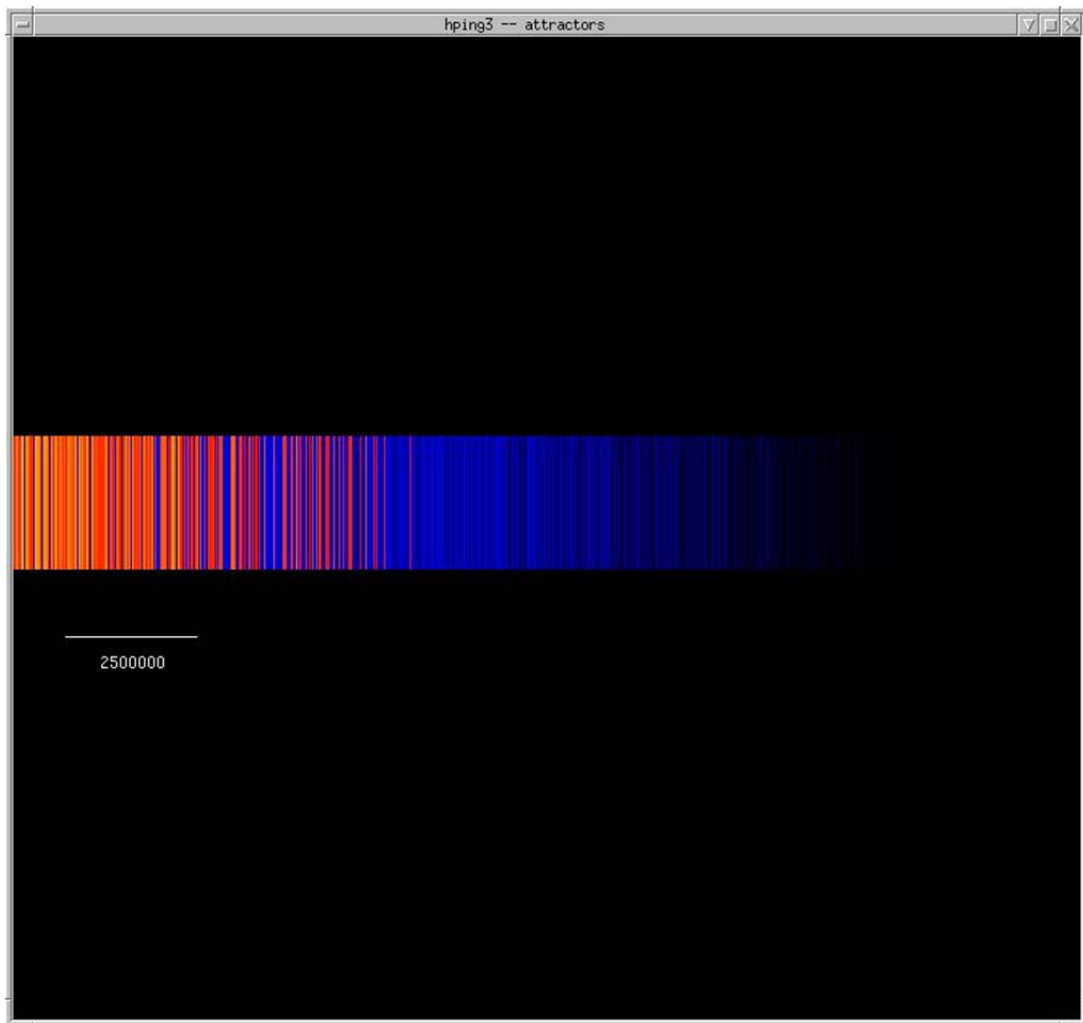
the script will dump the packets until you press ctrl+C.

A More Complex Example

The following is a more real-world example, an hping script used to analyze the Initial Sequence Number of a TCP/IP stack implementation. This example uses Tk in order to be able to display a spectrogram of the ISN increments in graphic. Note that the script is quite "rude" in order to make it as simple as possible (otherwise it will hardly be good in order to learn hping3 by examples).

Before to show the actual code, I want to show an example output for Linux and Windows.

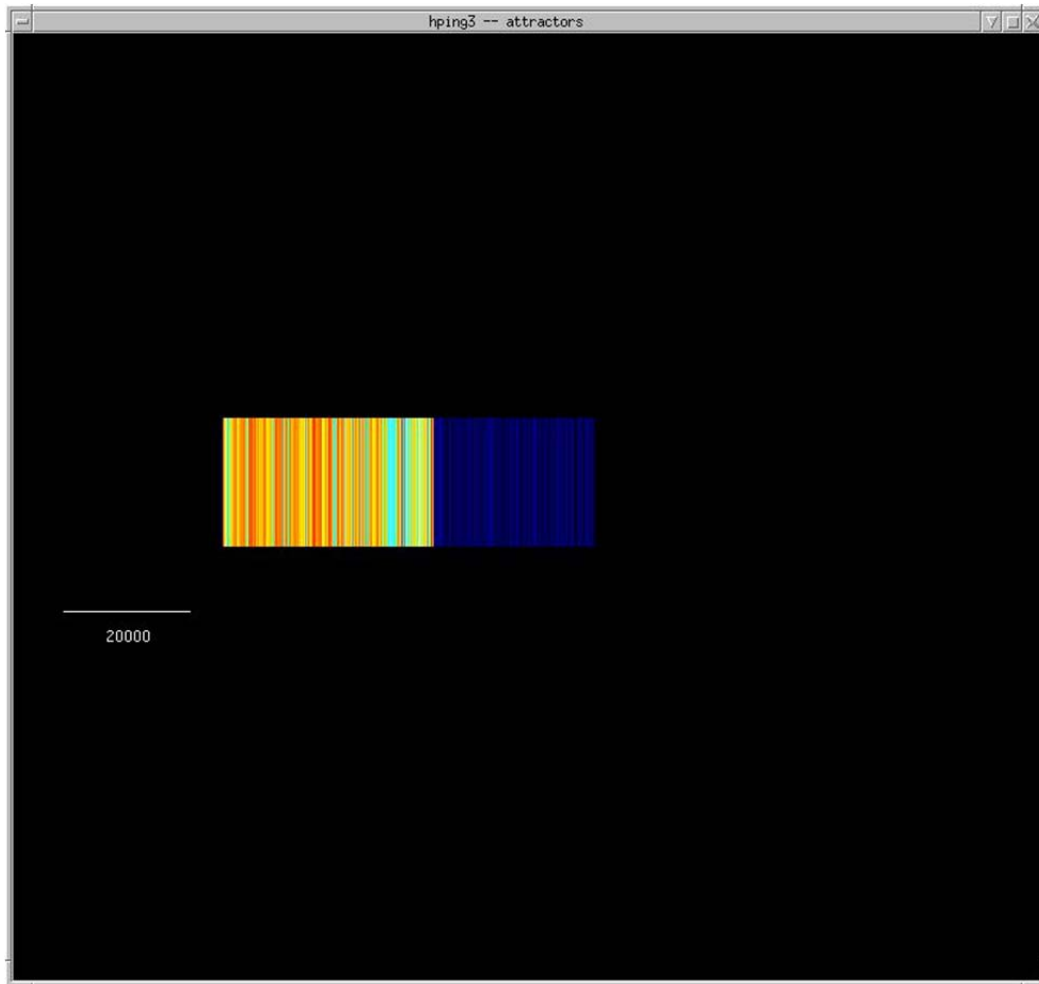
That's Linux:



While that's what I get with Windows 2000:

Getting Started with Hping3

unknown



To appreciate the real difference about the two OSes note the scale indication in the pictures.

Note that the script sends SYN packets to the target host always using the same IP address, so it does only check how random the increment is in a particular situation. But if your TCP/IP stack will show a bad spectrogram in this context, there is already something of bad (think about dialups, or DSL lines with dynamic IP, I can connect, sample a given host, reconnect with a different IP and try to do IP spoofing with the previous one).

Finally that's the hping3 script to do the actual analysis:

```
# isn-spectrogram.htcl -- show the ISN increments "spectrogram".
# Copyright(C) 2003 Salvatore Sanfilippo.
#
# All rights reserved.
#
# Here the idea is very simple, in operating systems implementing
# ISN as random increments, it is useless to analyze the whole
# sequence number, because the random part is just the increment.
```

Getting Started with Hping3

unknown

```
# Moreover, some weaknes isn't about correlation between previous
# and successive increments, but just about increments don't show
# a good distribution. So the idea is to display a spectrogram
# of the increments distribution instead of the more complex to read
# 3D attractors (See [1]). This way is possible to see at least some of
# the common vulnerabilities you can discover with 3D attractors,
# but it is much simpler to guess how hard is to exploit the system
# just from the picture.
#
# [1] http://razor.bindview.com/publish/papers/tcpseq.html
#
# Please if you make this script better write me back the
# changes. (antirez@invece.org).
#
# The script requires Tk to run.
```

```
package require Tk
source hpingstdlib.htcl
```

```
if {$argc != 3} {
    puts stderr "Usage: isn-spectrogram <host> <scale> <open-tcp-port>"
    puts stderr "Example: isn-spectrogram www.example.com 100000 80"
    exit
}
```

```
set bgcolor {#000000}
wm title . {hping3 -- attractors}
set w .main
frame $w
pack $w -side top
. config -background $bgcolor
$w config -background $bgcolor
```

```
# canvas
set xres 800
set yres 800
canvas $w.can -width $xres -height $yres
$w.can config -background $bgcolor
pack $w.can -fill both -expand true
```

```
# globals
foreach {hostname div dport} $argv break
set sport 1
#set dport 80
set target [hping resolve $hostname]
set targetif [outifname $target]
set myip [hping outifa $target]
set isnqueue {}
set relative_attractor 1
set lastisn 0
#set div 10000000
```

Getting Started with Hping3

unknown

```
hping setfilter $targetif "tcp and src host $target"

$w.can create rectangle 40 450 139 450 -fill white -width 0
$w.can create text 90 470 -fill white -text [expr $div*100]

proc sendsyn {} {
    global sport dport myip target
    append syn "ip(saddr=$myip,daddr=$target,ttl=255)+"
    append syn "tcp(sport=$sport,dport=$dport,flags=s)"
    hping send $syn
    incr sport
    after 1 sendsyn
}

proc recvsynack {} {
    global lastisn relative_attractor

    set packets [hping recv eth0 0 0]
    foreach p $packets {
        if {[hping hasfield tcp flags $p]} continue
        set isn [hping getfield tcp seq $p]
        if {$relative_attractor} {
            set tism [expr abs($isn-$lastisn)]
            set lastisn $isn
            set isn $tism
        }
        #puts "ISN: $isn"
        displaypoint $isn
    }
    after 10 recvsynack
}

proc displaypoint isn {
    global w xres yres pastcol div

    set isn [expr $isn/$div]
    set y 300
    set x $isn
    puts "$x $y"
    if {[haskey pastcol $x.$y]} {
        set graylevel [incr pastcol($x.$y) 10]
    } else {
        set pastcol($x.$y) 0
        set graylevel 0
    }
    if {$graylevel >= 256*3} {
        set graylevel [expr (256*3)-1]
    }
    if {$graylevel <= 255} {
        set b $graylevel
    }
}
```

Getting Started with Hping3

unknown

```
    set g 0
    set r 0
} elseif {$graylevel <= 511} {
    set b 0
    set g [expr $graylevel - 256]
    set r 255
} elseif {$graylevel <= 767} {
    set b 255
    set g 255
    set r [expr $graylevel - 512]
}
set color [format "%02X%02X%02X" $r $g $b]
$w.can create rectangle $x $y [expr $x+1] [expr $y+100] -fill $color -width 0
}
```

```
after 1 sendsyn
after 1 recvsynack
```

```
vwait forever
```

```
# vim: filetype=tcl softtabstop=4
```

If you know some basic Tcl/Tk you will find it very simple to read I hope. Note that if you want to run this code you require a little hping standard library, but both this program and the lib itself are under the /lib directory of the hping3 distribution, so don't bother to retype it from this page.

That's how to use the script against a Linux box.

```
cd /your/path/hping3/lib
../hping3 exec isn-spectrogram.htcl <target-host> 100000 25
```

Note that '25' is an open port, you need to specify an open TCP port for the target system. 100000 is instead the scale, if you see that the graph is bigger than the screen use a bigger scale value, if you see it concentrating in the left of the screen and very dense, use a lower one. auto-scaling is trivial but not implemented in that script.

Work in progress... I hope to refine this tutorial at some point