NMAP Cheat Sheet

nmap scan type: -sT	TcpConnect scan. Scans by attempting the TCP three way handshake connection
nmap scan type: -sX	Xmas scan. Scans by setting all flags on TCP packet
nmap scan type: -sS	SYN scan. Just send SYN packet
nmap scan type: -sU	UDP scan.
nmap scan type: -sA	ACK scan. Scans by just sending the ACK packet.
nmap scan type: -sF	FIN scan.
nmap scan type: -sN	NULL scan. Send TCP packet with flags all set to null.
nmap scan type: -sL	List/DNS scan. Simply list targets to scan
nmap scan type: -sP	PING scan
nmap scan type: -sO	Protocol scan.
nmap scan type: -sW	Window scan.
nmap scan type: -sI (i)	Idle scan.
nmap scan type: -sR	RPC scan
nmap ping detection: -P0	Don't ping
nmap ping detection: -PI (i)	ICMP ping
nmap ping detection: -PP	ICMP timestamp
nmap ping detection: -PT	TCP ping
nmap ping detection: -PS	SYN ping
nmap ping detection: -PB	= (PT + PI). TCP ping + ICMP ping
nmap ping detection: -PM	ICMP netmask
nmap output format: -oN	Normal format
nmap output format: -oG	Grepable format
nmap output format: -oX	XML format
nmap output format: -oA	All formats (normal + grepable + xml)
nmap timing: -T0	PARANOID - serial scan + 300 sec wait
nmap timing: -T1	SNEAKY - serial scan + 15 sec wait
nmap timing: -T2	POLITE - serial scan + 0.4 sec wait

nmap timing: -T3 NORMAL - parallel scan

nmap timing: -T4 AGGRESSIVE - parallel scan + 300 sec wait + 1.25 sec probe

nmap timing: -T5 INSANE - parallel scan + 75 sec timeout + 0.3 sec probe

nmap flag: -F Fast scan mode

nmap flag: -n No reverse DNS lookup

nmap flag: -S Source IP address

nmap flag: -g Port number

nmap flag: -f fragmentation

nmap flag: -O OS detection

nmap flag: -p port ranges

nmap flag: -D Use decoys to mask scan

nmap scan type: -sC Script enabled scan

nmap flag: -A Enable OS detection, version detection, script scanning, and traceroute

HPING3 Cheat Sheet

1.	**-A**	Set ACK flag
2.	**-F**	set FIN flag
3.	**hping3 -1 (IP address)**	Ping Sweep
4.	**hping3 -2 (IP address)**	UDP Scan
5.	**hping3 -8 50-60 -s (IP address) -v**	SYN scan on port 50-60
6.	**hping3 -8 (IP address)**	SYN Scan
7.	**hping3 -9 HTTP -I eth0**	Intercept all traffic containing HTTP signature
8.	**hping3 -A (IP address)**	ACK Scan
9.	**hping3 -F -P -U (IP address)**	XMAS Scan
10.	**Mode -1**	ICMP (ping sweep)
11.	**Mode -2**	UDP scan
12.	**Mode -8**	SYN Scan
13.	**Mode -9**	Listen Mode
14.	**-P**	Set PSH flag
16.	**-Q**	Show TCP Sequence Number
17.	**-R**	Set RST flag
20.	**-U**	Set URG flag
18.	**-S**	set SYN flag
19.	**-s**	set base source port
15.	**-p**	set destination port

SYNOPSIS

hping3 [**-hvnqVDzZ012WrfxykQbFSRPAUXYjJBuTG**] [**-c** *count*] [**-i** *wait*] [**--fast**] [**-I** *interface*] [**-9** *signature*] [**-a** *host*] [**-t** *ttl*] [**-N** *ip id*] [**-H** *ip protocol*] [**-g** *fragoff*] [**-m** *mtu*] [**-o** *tos*] [**-C** *icmp type*] [**-K** *icmp code*] [**-s** *source port*] [**-p[+][+]** *dest port*] [**-w** *tcp window*] [**-O** *tcp offset*] [**-M***tcp sequence number*] [**-L** *tcp ack*] [**-d** *data size*] [**-E** *filename*] [**-e** *signature*] [**--icmp-ipver** *version*] [**--icmp-iphlen** *length*] [**--icmp-iplen** *length*] [**--icmp-ipid** *id*] [**--icmp-ipproto** *protocol*] [**--icmp-cksum** *checksum*] [**--icmp-ts**] [**--icmp-addr**] [**--tcpexitcode**] [**--tcp-timestamp**] [**--tr-stop**] [**--tr-keep-ttl**] [**--tr-no-rtt**] [**--rand-dest**] [**--rand-source**] [**--beep**] hostname

DESCRIPTION

hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols. Using hping3 you are able to perform at least the following stuff:

- Test firewall rules - Advanced port scanning - Test net performance using different protocols, packet size, TOS (type of service) and fragmentation. - Path MTU discovery - Transferring files between even really fascist firewall rules. - Traceroute-like under different protocols. - Firewalk-like usage. - Remote OS fingerprinting. - TCP/IP stack auditing. - A lot of others.

It's also a good didactic tool to learn TCP/IP. hping3 is developed and maintained by antirez@invece.org and is licensed under GPL version 2. Development is open so you can send me patches, suggestion and affronts without inhibitions.

BASE OPTIONS

-h --help

Show a help screen on standard output, so you can pipe to less.

-v --version

Show version information and API used to access to data link layer, *linux sock packet* or *libpcap.*

-c --count count

Stop after sending (and receiving) *count* response packets. After last packet was send hping3 wait COUNTREACHED_TIMEOUT seconds target host replies. You are able to tune COUNTREACHED_TIMEOUT editing hping3.h

-i --interval

Wait the specified number of seconds or micro seconds between sending each packet. --interval X set *wait* to X seconds, --interval uX set *wait* to X micro seconds. The default is to wait one second between each packet. Using hping3 to transfer files tune this option is really important in order to increase transfer rate. Even using hping3 to perform idle/spoofing scanning you should tune this option, see **HPING3-HOWTO** for more information.

--fast

Alias for -i u10000. Hping will send 10 packets for second.

--faster

Alias for -i u1. Faster then --fast ;) (but not as fast as your computer can send packets due to the signal-driven design).

--flood

Sent packets as fast as possible, without taking care to show incoming replies. This is ways faster than to specify the -i u0 option.

-n --numeric

Numeric output only, No attempt will be made to lookup symbolic names for host addresses.

-q --quiet

Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

-I --interface interface name

By default on linux and BSD systems hping3 uses default routing interface. In other systems or when there is no default route hping3 uses the first non-loopback interface. However you are able to force hping3 to use the interface you need using this option. Note: you don't need to specify the whole name, for example -I et will match eth0 ethernet0 myet1 et cetera. If no interfaces match hping3 will try to use lo.

-V --verbose

Enable verbose output. TCP replies will be shown as follows:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms tos=0 iplen=40 seq=0 ack=1380893504 sum=2010 urp=0

-D --debug

Enable debug mode, it's useful when you experience some problem with hping3. When debug mode is enabled you will get more information about **interface detection, data link layer access, interface settings, options parsing, fragmentation, HCMP protocol** and other stuff.

-z --bind

Bind CTRL+Z to **time to live (TTL)** so you will able to increment/decrement ttl of outgoing packets pressing CTRL+Z once or twice.

-Z --unbind

Unbind CTRL+Z so you will able to stop hping3.

--beep

Beep for every matching received packet (but not for ICMP errors).

PROTOCOL SELECTION

Default protocol is TCP, by default hping3 will send tcp headers to target host's port 0 with a winsize of 64 without any tcp flag on. Often this is the best way to do a 'hide ping', useful when target is behind a firewall that drop ICMP. Moreover, a tcp null-flag to port 0 has a good probability of not being logged.

-0 --rawip

RAW IP mode, in this mode hping3 will send IP header with data appended with --signature and/or --file, see also --ipproto that allows you to set the ip protocol field.

-1 --icmp

ICMP mode, by default hping3 will send ICMP echo-request, you can set other ICMP type/code using **--icmptype --icmpcode** options.

-2 --udp

UDP mode, by default hping3 will send udp to target host's port 0. UDP header tunable options are the following: **--baseport, --destport, --keep.**

-8 --scan

Scan mode, the option expects an argument that describes groups of ports to scan. port groups are comma separated: a number describes just a single port, so 1,2,3 means port 1, 2 and 3. ranges are specified using a start-end notation, like 1-1000, that tell hping to scan ports between 1 and 1000 (included). the special word **all** is an alias for 0-65535, while the special word **known** includes all the ports listed in /etc/services.
Groups can be combined, so the following command line will scan ports between 1 and 1000 AND port 8888 AND ports listed in /etc/services: **hping --scan 1-1000,8888,known -S target.host.com**
Groups can be negated (subtracted) using a ! character as prefix, so the following command line will scan all the ports NOT listed in /etc/services in the range 1-1024: **hping --scan '1-1024,!known' -S target.host.com**
Keep in mind that while hping seems much more like a port scanner in this mode, most of the hping switches are still honored, so for example to perform a SYN scan you need to specify the **-S** option, you can change the TCP windows size, TTL, control the IP fragmentation as usually, and so on. The only real difference is that the standard hping behaviors are encapsulated into a scanning algorithm.
Tech note: The scan mode uses a two-processes design, with shared memory for synchronization. The scanning algorithm is still not optimal, but already quite fast.
Hint: unlike most scanners, hping shows some interesting info about received packets, the IP ID, TCP win, TTL, and so on, don't forget to look at this additional information when you perform a scan! Sometimes they shows interesting details.

-9 --listen signature

HPING3 listen mode, using this option hping3 waits for packet that contain *signature* and dump from *signature* end to packet's end. For example if hping3 --listen TEST reads a packet that contain **234-09sdflkjs45-TESThello_world** it will display **hello_world**.

IP RELATED OPTIONS

-a --spoof hostname

Use this option in order to set a fake IP source address, this option ensures that target will not gain your real address. However replies will be sent to spoofed address, so you will can't see them. In order to see how it's possible to perform spoofed/idle scanning see the **HPING3-HOWTO**.

--rand-source

This option enables the **random source mode**. hping will send packets with random source address. It is interesting to use this option to stress firewall state tables, and other per-ip basis dynamic tables inside the TCP/IP stacks and firewall software.

--rand-dest

This option enables the **random destination mode**. hping will send the packets to random addresses obtained following the rule you specify as the target host. You need to specify a numerical IP address as

target host like **10.0.0.x**. All the occurrences of **x** will be replaced with a random number in the range 0-255. So to obtain Internet IP addresses in the whole IPv4 space use something like **hping x.x.x.x --rand-dest**. If you are not sure about what kind of addresses your rule is generating try to use the **--debug** switch to display every new destination address generated. When this option is turned on, matching packets will be accept from all the destinations.
Warning: when this option is enabled hping can't detect the right outgoing interface for the packets, so you should use the **--interface** option to select the desired outgoing interface.

-t --ttl time to live

Using this option you can set **TTL (time to live)** of outgoing packets, it's likely that you will use this with **--traceroute** or **--bind** options. If in doubt try '**hping3 some.host.com -t 1 --traceroute**'.

-N --id

Set ip->id field. Default id is random but if fragmentation is turned on and id isn't specified it will be **getpid() & 0xFF**, to implement a better solution is in TODO list.

-H --ipproto

Set the ip protocol in RAW IP mode.

-W --winid

id from Windows* systems before Win2k has different byte ordering, if this option is enable hping3 will properly display id replies from those Windows.

-r --rel

Display id increments instead of id. See the **HPING3-HOWTO** for more information. Increments aren't computed as id[N]-id[N-1] but using packet loss compensation. See relid.c for more information.

-f --frag

Split packets in more fragments, this may be useful in order to test IP stacks fragmentation performance and to test if some packet filter is so weak that can be passed using tiny fragments (anachronistic). Default 'virtual mtu' is 16 bytes. see also *--mtu* option.

-x --morefrag

Set more fragments IP flag, use this option if you want that target host send an **ICMP time-exceeded during reassembly**.

-y --dontfrag

Set don't fragment IP flag, this can be used to perform **MTU path discovery**.

-g --fragoff fragment offset value

Set the fragment offset.

-m --mtu mtu value

Set different 'virtual mtu' than 16 when fragmentation is enabled. If packets size is greater that 'virtual mtu' fragmentation is automatically turned on.

-o --tos hex_tos

Set **Type Of Service (TOS)**, for more information try **--tos help**.

-G --rroute

Record route. Includes the RECORD_ROUTE option in each packet sent and displays the route buffer of returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option. Also note that using hping you are able to use record route even if target host filter ICMP. Record route is an IP option, not an ICMP option, so you can use record route option even in TCP and UDP mode.

ICMP RELATED OPTIONS

-C --icmptype type

Set icmp type, default is **ICMP echo request** (implies --icmp).

-K --icmpcode code

Set icmp code, default is 0 (implies --icmp).

--icmp-ipver

Set IP version of IP header contained into ICMP data, default is 4.

--icmp-iphlen

Set IP header length of IP header contained into ICMP data, default is 5 (5 words of 32 bits).

--icmp-iplen

Set IP packet length of IP header contained into ICMP data, default is the real length.

--icmp-ipid

Set IP id of IP header contained into ICMP data, default is random.

--icmp-ipproto

Set IP protocol of IP header contained into ICMP data, default is TCP.

--icmp-cksum

Set ICMP checksum, for default is the valid checksum.

--icmp-ts

Alias for --icmptype 13 (to send ICMP timestamp requests).

--icmp-addr

Alias for --icmptype 17 (to send ICMP address mask requests).

TCP/UDP RELATED OPTIONS

-s --baseport source port

hping3 uses source port in order to guess replies sequence number. It starts with a base source port number, and increase this number for each packet sent. When packet is received sequence number can be computed as *replies.dest.port - base.source.port*. Default base source port is random, using this option you are able to set different number. If you need that source port not be increased for each sent packet use the *-k --keep* option.

-p --destport [+][+]dest port

Set destination port, default is 0. If '+' character precedes dest port number (i.e. +1024) destination port will be increased for each reply received. If double '+' precedes dest port number (i.e. ++1024), destination port will be increased for each packet sent. By default destination port can be modified interactively using **CTRL+z**.

--keep

keep still source port, see *--baseport* for more information.

-w --win

Set TCP window size. Default is 64.

-O --tcpoff

Set fake tcp data offset. Normal data offset is tcphdrlen / 4.

-M --setseq

Set the TCP sequence number.

-L --setack

Set the TCP ack.

-Q --seqnum

This option can be used in order to collect sequence numbers generated by target host. This can be useful when you need to analyze whether TCP sequence number is predictable. Output example:

#hping3 win98 --seqnum -p 139 -S -i u1 -I eth0

HPING uaz (eth0 192.168.4.41): S set, 40 headers + 0 data bytes

2361294848 +2361294848

2411626496 +50331648

2545844224 +134217728

2713616384 +167772160

2881388544 +167772160

3049160704 +167772160

3216932864 +167772160

3384705024 +167772160

3552477184 +167772160

3720249344 +167772160

3888021504 +167772160

4055793664 +167772160

4223565824 +167772160

The first column reports the sequence number, the second difference between current and last sequence number. As you can see target host's sequence numbers are predictable.

-b --badcksum

Send packets with a bad UDP/TCP checksum.

--tcp-timestamp

Enable the TCP timestamp option, and try to guess the timestamp update frequency and the remote system uptime.

-F --fin

Set FIN tcp flag.

-S --syn

Set SYN tcp flag.

-R --rst

Set RST tcp flag.

-P --push

Set PUSH tcp flag.

-A --ack

Set ACK tcp flag.

-U --urg

Set URG tcp flag.

-X --xmas

Set Xmas tcp flag.

-Y --ymas

Set Ymas tcp flag.

COMMON OPTIONS

-d --data data size

Set packet body size. Warning, using --data 40 hping3 will not generate 0 byte packets but protocol_header+40 bytes. hping3 will display packet size information as first line output, like this: **HPING www.yahoo.com (ppp0 204.71.200.67): NO FLAGS are set, 40 headers + 40 data bytes**

-E --file filename

Use **filename** contents to fill packet's data.

-e --sign signature

Fill first *signature length* bytes of data with *signature*. If the *signature length* is bigger than data size an error message will be displayed. If you don't specify the data size hping will use the signature size as data size. This option can be used safely with *--file filename* option, remainder data space will be filled using *filename*.

-j --dump

Dump received packets in hex.

-J --print

Dump received packets' printable characters.

-B --safe

Enable safe protocol, using this option lost packets in file transfers will be resent. For example in order to send file /etc/passwd from host A to host B you may use the following:

[host_a]

 # hping3 host_b --udp -p 53 -d 100 --sign signature --safe --file /etc/passwd

[host_b]

 # hping3 host_a --listen signature --safe --icmp

-u --end

If you are using *--file filename* option, tell you when EOF has been reached. Moreover prevent that other end accept more packets. Please, for more information see the **HPING3-HOWTO**.

-T --traceroute

Traceroute mode. Using this option hping3 will increase ttl for each **ICMP time to live 0 during transit** received. Try **hping3 host --traceroute**. This option implies --bind and --ttl 1. You can override the ttl of 1 using the --ttl option. Since 2.0.0 stable it prints RTT information.

--tr-keep-ttl

Keep the TTL fixed in traceroute mode, so you can monitor just one hop in the route. For example, to monitor how the 5th hop changes or how its RTT changes you can try **hping3 host --traceroute --ttl 5 --tr-keep-ttl**.

--tr-stop

If this option is specified hping will exit once the first packet that isn't an ICMP time exceeded is received. This better emulates the traceroute behavior.

--tr-no-rtt

Don't show RTT information in traceroute mode. The ICMP time exceeded RTT information aren't even calculated if this option is set.

--tcpexitcode

Exit with last received packet tcp->th_flag as exit code. Useful for scripts that need, for example, to known if the port 999 of some host reply with SYN/ACK or with RST in response to SYN, i.e. the service is up or down.

TCP OUTPUT FORMAT

The standard TCP output format is the following:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms

len is the size, in bytes, of the data captured from the data link layer excluding the data link header size. This may not match the IP datagram size due to low level transport layer padding.

ip is the source ip address.

flags are the TCP flags, R for RESET, S for SYN, A for ACK, F for FIN, P for PUSH, U for URGENT, X for not standard 0x40, Y for not standard 0x80.

If the reply contains **DF** the IP header has the don't fragment bit set.

seq is the sequence number of the packet, obtained using the source port for TCP/UDP packets, the sequence field for ICMP packets.

id is the IP ID field.

win is the TCP window size.

rtt is the round trip time in milliseconds.

If you run hping using the **-V** command line switch it will display additional information about the packet, example:

len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 ms tos=0 iplen=40 seq=0 ack=1223672061 sum=e61d urp=0

tos is the type of service field of the IP header.

iplen is the IP total len field.

seq and ack are the sequence and acknowledge 32bit numbers in the TCP header.

sum is the TCP header checksum value.

urp is the TCP urgent pointer value.

UDP OUTPUT FORMAT

The standard output format is:

len=46 ip=192.168.1.1 seq=0 ttl=64 id=0 rtt=6.0 ms

The field meaning is just the same as the TCP output meaning of the same fields.

ICMP OUTPUT FORMAT

An example of ICMP output is:

ICMP Port Unreachable from ip=192.168.1.1 name=nano.marmoc.net

It is very simple to understand. It starts with the string "ICMP" followed by the description of the ICMP error, Port Unreachable in the example. The ip field is the IP source address of the IP datagram containing the ICMP error, the name field is just the numerical address resolved to a name (a dns PTR request) or UNKNOWN if the resolution failed.

The ICMP Time exceeded during transit or reassembly format is a bit different:

TTL 0 during transit from ip=192.168.1.1 name=nano.marmoc.net

TTL 0 during reassembly from ip=192.70.106.25 name=UNKNOWN

The only difference is the description of the error, it starts with TTL 0.

XOR and use for Nonce

XOR represents the bitwise logical "exclusive-or".
Please find the XOR truth table below:

INPUT		
A	**B**	**OUTPUT**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

0 = False
1 = True

When the **inputs match the output is a 0,** and when the **inputs do not match the output is a 1**.
Use the XOR operation process below:

Example below taken from RFC3174

```
        01101100101110011101001001111011
XOR     01100101110000010110100110110111
    =   00001001011110001011101111001100
```

Types of Bitwise logical word operations:
X AND Y = bitwise logical "and" of X and Y.
X OR Y = bitwise logical "inclusive-or" of X and Y.
X XOR Y = bitwise logical "exclusive-or" of X and Y.
NOT X = bitwise logical "complement" of X.

Using XOR in practice: Below is an example of applying the XOR function to an initialization vector/nonce for a cryptographic cipher.
Let's break down the definition so that we can understand all the pieces:
A nonce is an initialization vector (IV), a random bit string that is the same length as the block size and is XORed with the message.

An initialization vector is a field added to the payload (any attribute likely to be known by the both the sender and receiver but be unpredictable by a third party). Therefore a nonce is an initialization vector; it is added to the encrypted payload as part of the cipher-text. The nonce can be random, can be a counter, a timestamp, or a message number.

A nonce can also be considered as a one-time session key. You would not want to reuse a nonce, it should be unique to each execution of the encryption operation.
If the nonce is XOR'd then the initialization vector goes through a modulus of 2 operation; this can also be considered an exclusive disjunction operation. *The best way and easiest way to define an exclusive disjunction operation is to think of this as 'flipping the bits' or 'bit flipping'.*

Example below taken from RFC3174 (example same as above)

01101100101110011101001001111011
XOR 01100101110000010110100110110111
= 00001001011110001011101111001100

So, why would we need to use a nonce and in what types of technologies would it be used?
A nonce is used to prevent replay attacks as well as man-in-the-middle attacks, and the function assists with session/message synchronization.

Here are some examples of how a nonce can be used:
RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication
Like Basic Access Authentication, the Digest scheme is based on a simple challenge-response paradigm. The Digest scheme challenges using a nonce value. A valid response contains a checksum default, the MD5 checksum) of the username, the password, the given nonce value, the HTTP method, and the requested URI. In this way, the password is never sent in the clear. Just as with the Basic scheme, the username and password must be prearranged in some fashion not addressed by this document.
…….cut for brevity….
An implementation might choose not to accept a previously used nonce or a previously used digest, in order to protect against a replay attack. Or, an implementation might choose to use one-time nonces or digests for POST or PUT requests and a time-stamp for GET requests.

RFC 4418 - UMAC: Message Authentication Code using Universal Hashing - https://tools.ietf.org/html/rfc4418
Nonce is a value that changes with each generated tag. The receiver needs to know which nonce was used by the sender, so some method of synchronizing nonces needs to be used. This can be done by explicitly sending the nonce along with the message and tag, or agreeing upon the use of some other non-repeating value such as a sequence number. The nonce need not be kept secret, but care needs to be taken to ensure that, over the lifetime of a UMAC key, a different nonce is used with each message.