Welcome to

DDoS Testing Your Infrastructure, including IPv6 SYN floods

Hacktivity 2022

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Github

hacktivity-2022.tex in the repo security-courses

Note: My main contribution is about performing structured DDoS testing, many great tools exist already

Contact information

- Henrik Kramselund, he/him internet samurai mostly networks and infosec
- Network and security consultant Zencurity.com, teach at KEA and activist
- Master from the Computer Science Department at the University of Copenhagen, DIKU
- Email: hlk@zencurity.dk Mobile: +45 2026 6000
- Run a small network for fun AS57860

You are welcome to drop me an email

What is this presentation about

When **connecting to the Internet we immediately receive traffic from unknown sources**. We should consider **testing our infrastructure using active pentest methods**, to verify robustness.

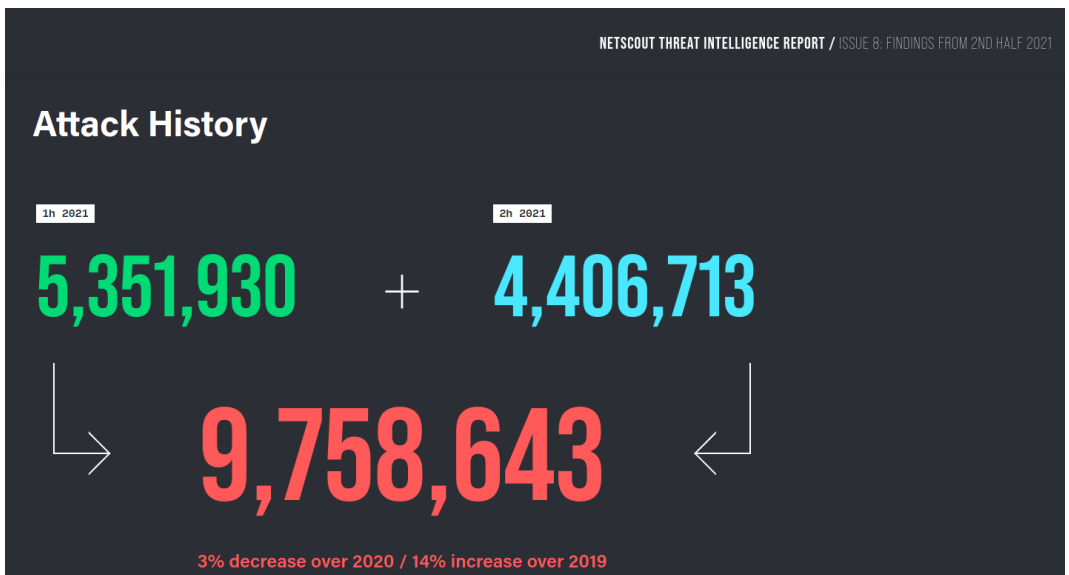
You will learn:

- This talk will be about DDoS testing your infrastructures
- Ultra short firewall description – what is a firewall really, short
- Doing port scans for discovery of infrastructures ...
- Followed by detailed advice how to perform active DDoS simulation
- My advice for protection using your existing devices and networks

Note: The attack tools will be already developed and possibly known tools, but with a lot of focus on the process and experiences. I also have some opinions and experiences to share.

The Internet and DDoS is trouble

Security attacks and DDoS is very much in the media

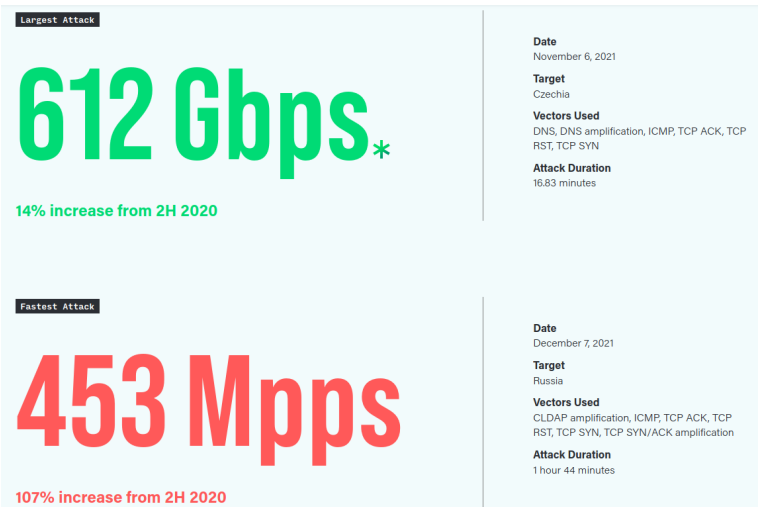


Source: Netscout Global DDoS Threat Intelligence Report 2nd half 2021

https://www.netscout.com/threatreport/global-ddos-attack-trends/

DDoS Attacks are HUGE

Extremely hard to protect against from a small network



Source: Netscout Global DDoS Threat Intelligence Report 2nd half 2021

https://www.netscout.com/threatreport/global-ddos-attack-trends/

We can do a lot to improve our infrastucture – Don't give up!

Definition of firewalls – multiple definitions exist

We define a firewall as **a collection of components** placed between two networks that collectively have the following properties:

- All traffic from inside to outside, and vice-versa, must pass through the firewall.
- Only authorized traffic, as defined by the **local security policy**, will be allowed to pass.
- The firewall itself is immune to penetration.

We should note that these are design goals; a failure in one aspect does not mean that the collection is not a firewall, simply that it is not a very good one.

We will consider this a firewall, but we know today that **both inside and outside at meaningless**, since we have **multiple networks inside, we have partner network connections etc.**

Source: *Firewalls and Internet Security; Repelling the Wily Hacker.* by Cheswick and Bellovin 1994

Definition of firewalls – Wikipedia

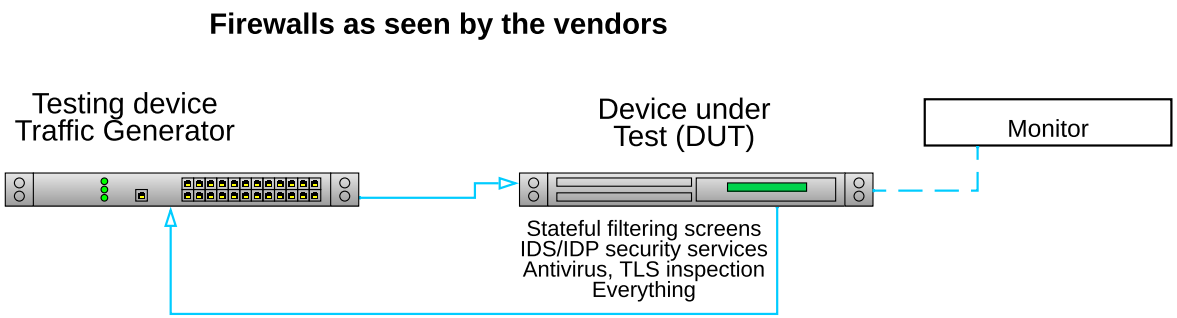
Another short definition that encapsulates this is found on Wikipedia, and may suffice in many situations. Again there will typically be multiple networks, zones or areas of the networks with varying degrees of trust.

> In computing, a firewall is a **network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules**.[1] A firewall typically establishes a barrier between a trusted network and an untrusted network, such as the Internet.[2]

Source: Wikipedia https://en.wikipedia.org/wiki/Firewall_(computing)

TL;DR Not necessarily a single device

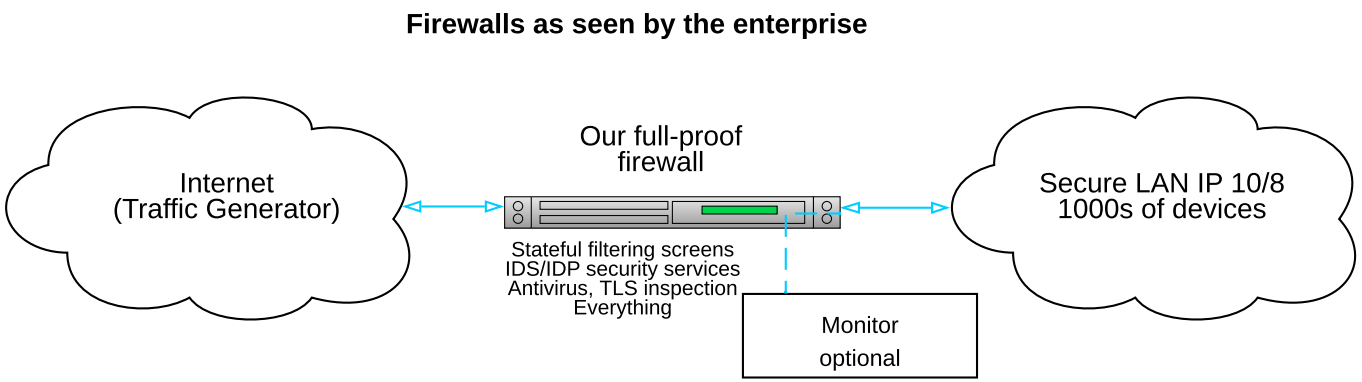
A firewall – in the vendor eyes



"**Can your firewall flex in the face of change?**
Does it harmonize your network, workload, and application security? Does it protect apps and employees in your hybrid or multicloud environment? Make sure you're covered."

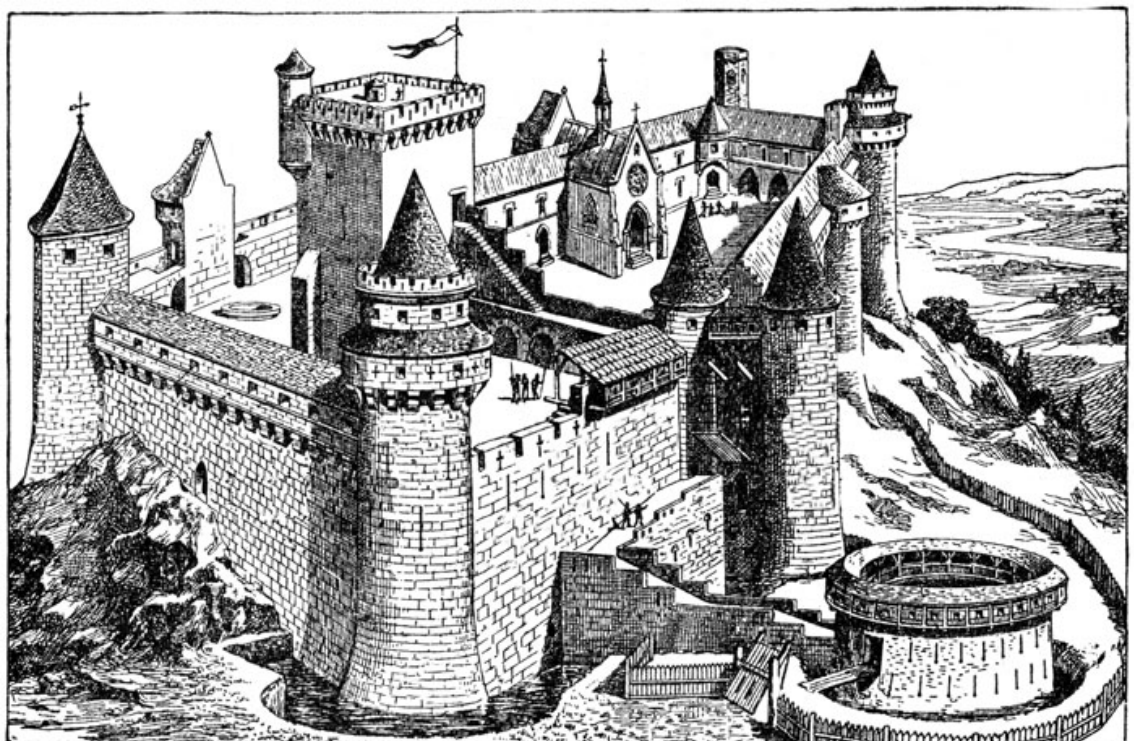
Source: not shown to protect the audience from further marketing speak

A firewall – in the enterprise mindset



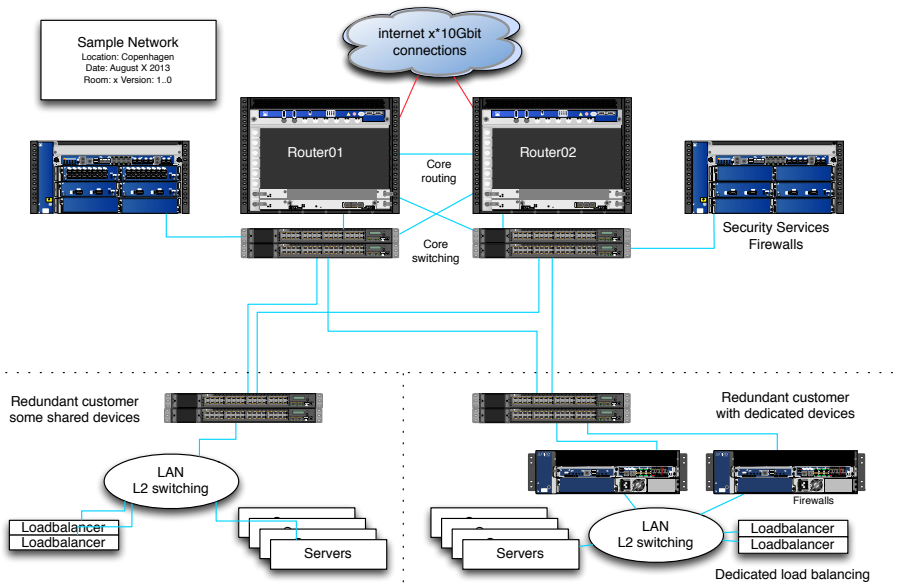
- Even though some vendors suggest they can do everything in a single box, I don't believe them!
- Truth – yes, we can do almost anything in software
- Realization **Your infrastructure is based on multiple components and or devices**

Defense in depth



Picture originally from: http://karenswhimsy.com/public-domain-images

Bottlenecks exist, but where



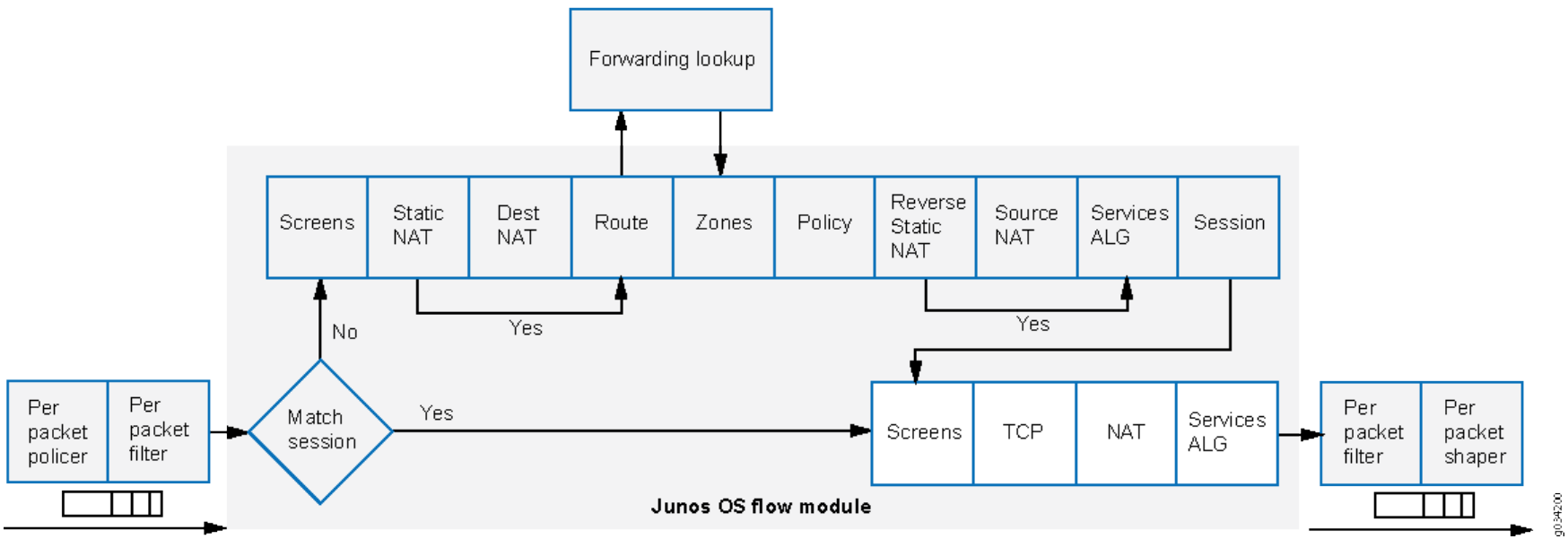
- Lower layer attacks Transport Layer Attacks TCP SYN flood – packet based
- Higher layer attacks like Slowloris and web attacks – keep sessions running
- Protect everything without loosing functionality or creating administrative nightmare

Availability and Network flooding attacks

The attacks we are discussing today are:

- **SYN flood** is the most basic and very common on the internet towards 80/tcp and 443/tcp
- **ICMP and UDP flooding** are the next popular targets – more similar ones exist
- Special packets and protocols – anything that can create *load on systems* work
- All of them try to use up some resources
- **Memory space** in specific sections of the **kernel, TCP state, firewalls state, number of concurrent sessions/connections**
- **Interrupt processing** of packets - packets per second (pps)
- **CPU processing** in firewalls, pps
- CPU processing in server software
- **Bandwidth** - megabits per second (mbps)
- Typically source is spoofed or amplification attacks abusing devices on the Internet

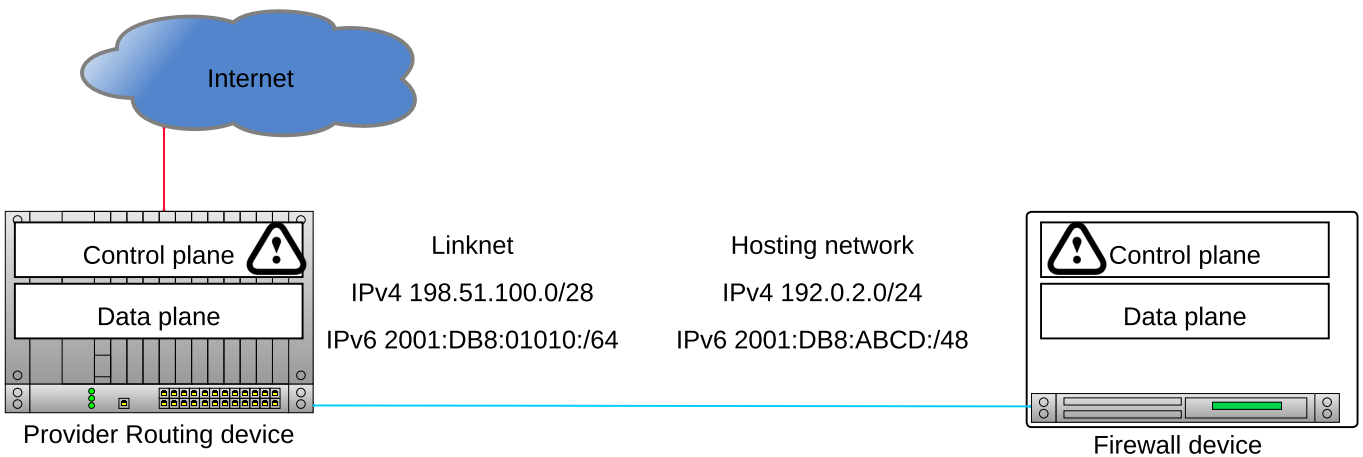
Packet processing in firewalls – detailed view



Traffic Processing on SRX Series Devices Overview

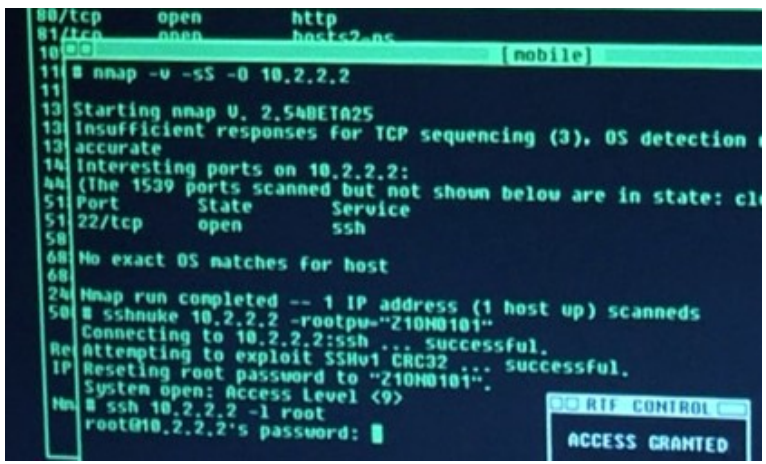
https://www.juniper.net/documentation/us/en/software/junos/flow-packet-processing/topics/topic-map/security-srx-devices-processing-overview.html

Scanning and Attacking – Pressure Points and Scope



- In scope for me is everything that could adversely affect the network
- Typical scope IPv4: Link network /28 or /26 and a hosting network /24 or even /22
- Typical scope IPv6: Link network /64 (bad) or /127 (RFC9099) and a hosting network /48 with subnets

Prepare for the testing



- Portscan the whole linknet and hosting range in IPv4
- Ask about IPv6 ranges in use, specific subnets and IPv6 addresses
  We can guess from traceroute, Nmap test first 100 addresses in each subnet etc. but easier to ask
- Portscan the whole linknet - identify provider devices, hosting network devices, type of device router/firewall
- Also Thank you Fyodor (Gordon Lyon) and contributors for Nmap!

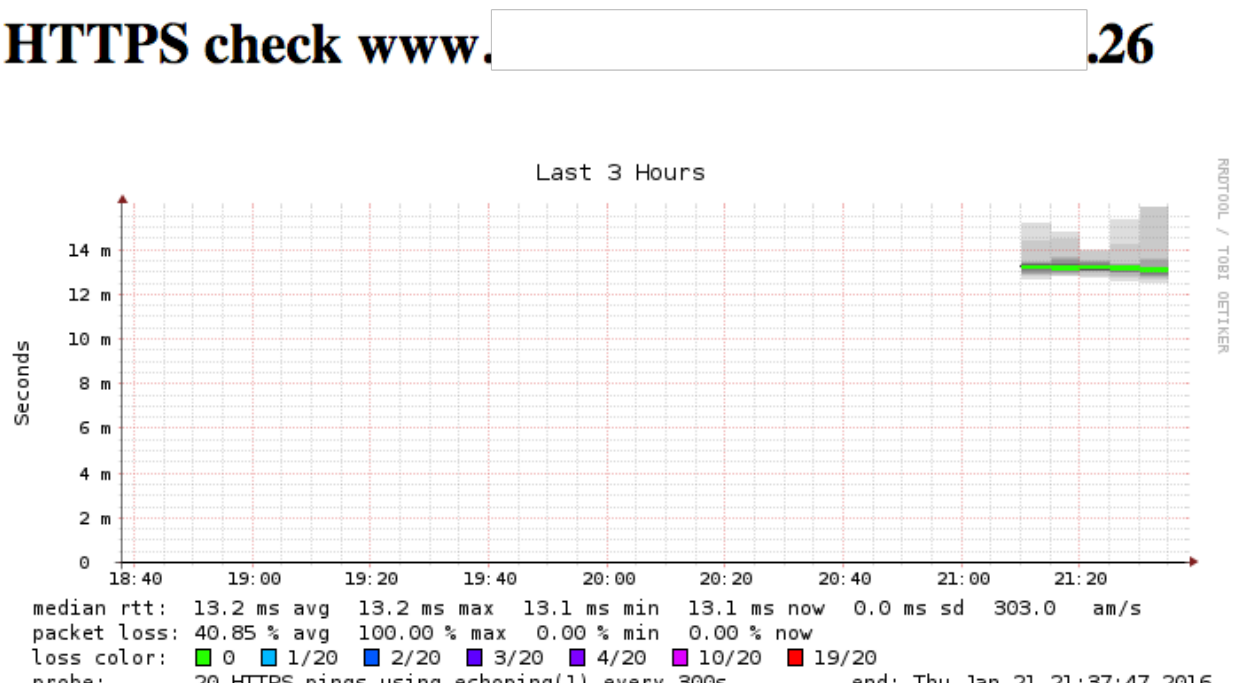
Detailed Scope and Plan

Select a few targets for monitoring and attacks, from the port scans

Best case would be to have:

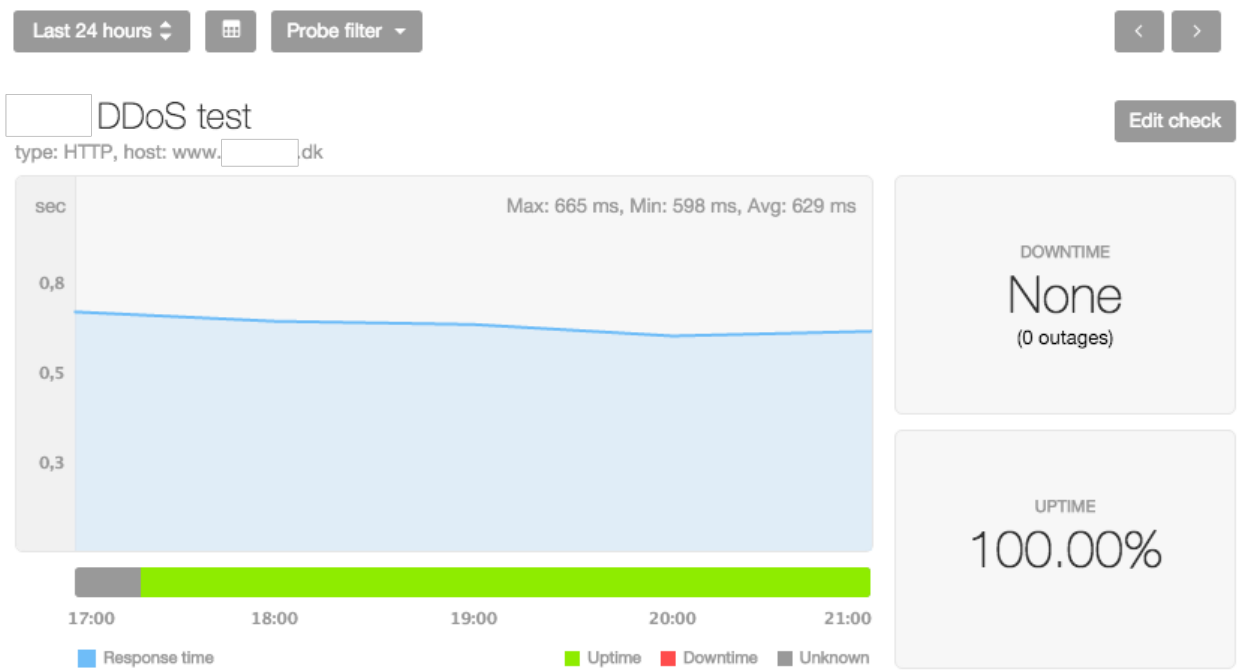
- Ping ICMP allowed to a provider router and hosting firewall – is the connection alive
- TCP port with service checks, HTTP being attacked and one not being attacked
- UDP port with service check, DNS is a favourite – ask for localhost/127.0.0.1
- Put monitoring on these, a week before testing is nice
- Agree on a day or night for testing, inform participants and system owners

Before testing: Smokeping



Before DDoS testing use Smokeping software

Before testing: Pingdom



Another external monitoring from Pingdom.com

Performing the DDoS test

So lets break this task down into:

- Use Nmap to port scan the network
- Run testing tool `hping3`, `t50` or `penguinping`
- Go through all scenarios before making changes to environment
- Expect things to break, investigate, repeat failed scenarios

- BTW we usually schedule this for night time! There WILL be interruptions

- How do you run the tools, since they all have so many options?
  `man nmap | enscript -o test.ps` result in 54 pages, `hping3` has about 80 options and `t50` has even more

Hint: Save the scope in variables

Hint: use a variable to keep the target address, carefully enter it and avoid mystyping it later

```
# export  CUST_NET4="192.0.2.0/24"
# export CUST_NET6="2001:DB8:ABCD:1000::/64"
# nmap -p 1-65535 -Pn -A -oA full-scan  $CUST_NET4

# export CUST_IP=192.0.2.138
# date;time hping3 -q -c 1000000  -i u60 -S -p 80 $CUST_IP
```

Better yet, script it all – but most likely you will want to repeat specific steps.

Nmap port sweep for TCP services, full TCP scan

Goal is to enumerate the ports that are allowed through the network.

```
#  nmap -Pn -A -p 1-65535 -oA full-tcp-customer-ipv4 $CUST_NET4
...
Nmap scan report for 192.0.2.138
Host is up (0.00012s latency).
PORT    STATE  SERVICE
80/tcp  open   http
443/tcp closed https
#  nmap -Pn -A -p 1-65535 -oA full-tcp-customer-ipv6 $CUST_NET6
#  nmap -Pn -A -p 1-65535 -oA full-tcp-linknet-ipv4 $LINK_NET4
#  nmap -Pn -A -p 1-65535 -oA full-tcp-linknet-ipv6 $LINK_NET6
```

Note: Pretty harmless, if something dies, then it is *vulnerable to normal traffic* - and should be fixed!

```
Options:
-Pn -- Scan all IPs, dont use ping or TCP ping to check alive    -A advanced -- perform full TCP connection and grab banner
-p 1-65535 -- full portscan all ports  -oA filename -- Saves output in "all formats" normal, XML, and grepable formats
```

Nmap port sweep for SNMP port 161/UDP

Perform some UDP scanning, cannot do full scan, but often SNMP is there, example:

```
# nmap -A -sU -p 161 --script "snmp-info" -oA snmp-scan $LINK_NET4
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-26 20:20 CEST
Nmap scan report for 193.111.162.0
Host is up (0.00082s latency).

PORT    STATE SERVICE VERSION
161/udp open  snmp    Cisco SNMP service; ciscoSystems SNMPv3 server
| snmp-info:
|   enterprise: ciscoSystems
|   engineIDFormat: mac
|   engineIDData: 00:08:4f:xx:yy:zz
|   snmpEngineBoots: 4
|_  snmpEngineTime:  732d07h09m04s
Too many fingerprints match this host to give specific OS details
Network Distance: 6 hops
```

More reliable to use Nmap script with probes like –script=snmp-info

hping3 packet generator

```
usage: hping3 host [options]
  -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.
UDP/TCP
  -s  --baseport   base source port             (default random)
  -p  --destport   [+][+]<port> destination port(default 0) ctrl+z inc/dec
  -L  --setack     set TCP ack
  -F  --fin        set FIN flag
  -S  --syn        set SYN flag
...
```

- Hping3 packet generator is a very flexible tool to produce simulated DDoS traffic with specific charateristics
- Home page: http://www.hping.org/hping3.html Source repository https://github.com/antirez/hping
- My fork with IPv6 and VXLAN branches added https://github.com/kramse/hping-2018
- Used to be my primary DDoS testing tool, now I am moving to Penguinping / MoonGen

t50 packet generator

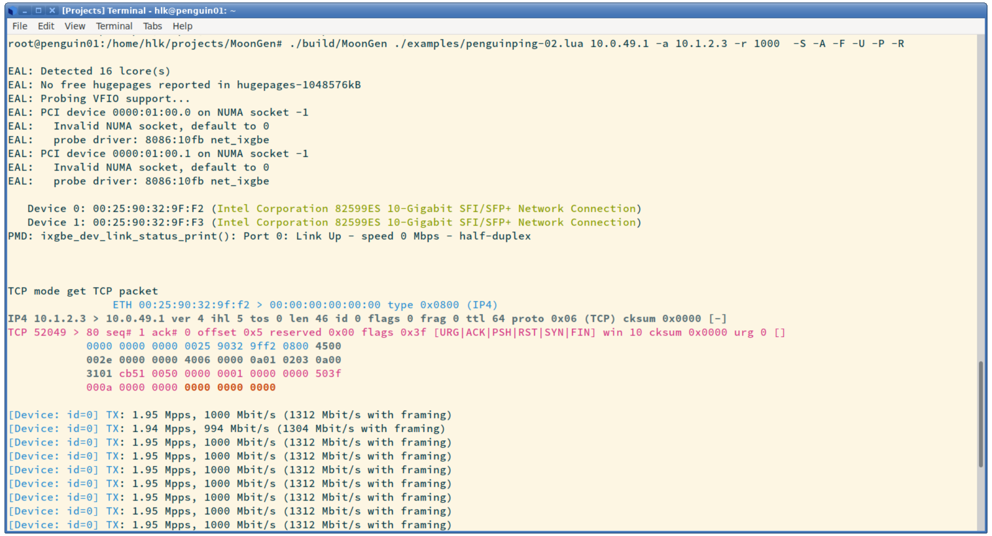
```
#  t50 -?
T50 Experimental Mixed Packet Injector Tool 5.4.1
Originally created by Nelson Brito <nbrito@sekure.org>
Maintained by Fernando Mercês <fernando@mentebinaria.com.br>

Usage: T50 <host> [/CIDR] [options]

Common Options:
    --threshold NUM        Threshold of packets to send     (default 1000)
    --flood                This option supersedes the 'threshold'
...
6. Running T50 with '--protocol T50' option, sends ALL protocols sequentially.
#  t50 -? | wc -l
264
```

- T50 packet generator, another high speed packet generator can easily overload most firewalls by producing randomized traffic with multiple protocols like IPsec, GRE, MIX
  home page: http://t50.sourceforge.net/resources.html
- Very fast and breaks most firewalls when flooding all protocols, easy 800k pps/400Mbps

Penguingping packet generator



- Penguinping packet generator, my high speed packet generator home page: https://penguinping.org
- First versions are only about 230 lines of Lua code and implement basic command line to replace hping3
- Built on top of MoonGen/Libmoon https://github.com/emmericp/MoonGen

Extremely fast and allows easy customization

Process: monitor, attack, break, repeat

- Start small, run with delays between packets
- Turn up until it breaks, decrease delay - until using full bandwidth / max pps
- Monitor speed of attack on your router interface pps/bandwidth
- Give it up to maximum speed
  `hping3 --flood -1` and `hping3 --flood -2` – probably about 1mpps / core/process
  `penguinping -r 10000` if you have it – rate 10Gbit using one CPU core!
- Have a common chat with network operators/customer to talk about symptoms and things observed
- Any information resulting from testing is good information

Running Attacks with hping3

```
# export CUST_IP=192.0.2.138
# date;time hping3 -q -c 1000000  -i u60 -S -p 80 $CUST_IP &
Thu Jan 21 22:37:06 CET 2022
HPING 192.0.2.1 (eth0 192.0.2.1): S set, 40 headers + 0 data bytes

--- 192.0.2.1 hping statistic ---
1000000 packets transmitted, 999996 packets received, 1% packet loss
round-trip min/avg/max = 0.9/7.0/1005.5 ms

real    1m7.438s
user    0m1.200s
sys     0m5.444s
```

Dont forget to do a killall hping3 when done ☺

Recommendations During Test

- Run each test for at least 5 minutes, or even 15 minutes
  Some attacks require some build-up before resource run out
- Take note of any change in response, higher latency, lost probes
- If you see a change, then **re-test using the same parameters**, or a little less first
- We want to know the **approximate level where it breaks**
- **If you want to change environment, then wait until all scenarios are tested**
- Keep a log handy, write notes and start the session with `script ddos-date.log`
- Check once in a while if you have some process running, using `ps auxw | grep hping3`
- Run multiple instances of the tools. One process might generate 800.000 pps, while two may double it. Though 10 processes might not be 10 times exactly

Running the tools

A basic test would be:

- TCP SYN flooding
- TCP other flags, PUSH-ACK, RST, ACK, FIN
- ICMP flooding
- UDP flooding
- Spoofed packets src=dst=target ☺
- Small fragments
- Bad fragment offset
- Bad checksum
- Be creative
- Mixed packets - like `t50 --protocol T50`
- Perhaps esoteric or unused protocols, GRE, IPSec

Test-cases / Scenarios

The minimal run contains at least these:

- SYN flood: `hping3 -q -c 1000000  -i u60 -S -p 80 $CUST_IP &`
- SYN+ACK: `hping3 -q -c 1000000  -i u60 -S -A -p 80 $CUST_IP &`
- ICMP flood: `hping3 -q -c 1000000 --flood -1 $CUST_IP &`
- UDP flood: `hping3 -q -c 1000000 --flood -2 $CUST_IP &`
- Near end of test we also throw in the joker to **kill firewalls**
  `t50 --flood --protocol T50 $CUST_IP`

While testing I use the tool `ifpps` tool from the Netsniff-ng package http://netsniff-ng.org/ to monitor sending speed, or you can use your router/switch – Junos `monitor interface`

Tuning the testing

Further hints:

- Vary the speed using the packet interval `-i u60` up/down
- Add more processes and monitor change in responses
- Use flooding with caution, runs max speeeeeeeeeeeed ☺
- TCP testing use a port which is allowed through the network, often 80/443
- Focus on attacks which are hard to block, example TCP SYN must be allowed in
- Also if you found devices like routers in front of environment
  `hping3 -q -c 1000000  -i u60 -S -p 22 $ROUTER_IP` Secure Shell
  `hping3 -q -c 1000000  -i u60 -S -p 179 $ROUTER_IP` BGP routing protocol
  (Hint: routers should use router protection filters!)

I start using a single test-case at a time, but later run multiple in parallel

Note about IPv6 Testing

My favourite tools have not always supported IPv6, which is a shame

Two options, modify tools or use newer tools

- Fortunately these tools are open source,
- I truly love Hping, this tools is very flexible and powerful, so I modified it to suit my needs, basically search and replace for inet to inet6, so I could have an IPv6 DDoS testing tool
  https://github.com/kramse/hping-2018 Hping-2018, rough support for IPv6 packets
  `sudo ./hping3 --inet6 -I eth0 -D -c 1 -p 80 www.kramse.org`
- Today I would recommend using MoonGen – which support IPv6 already
  https://github.com/emmericp/MoonGen
- I have created a small script Penguinping which using 230 lines of Lua can re-create the same basic attacks as Hping3 – but using less CPU and easier -r rate option
  https://penguinping.org/

Test-cases / Scenarios, continued Spoof Source

Spoofed packets src=dst=target ☺

Flooding with spoofed packet source, within hosting range,
may result in your **single packet** returning answer to another inside network + ARP traffic

```
-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.
```

```
hping3 -q --flood -p 80 -S -a $CUST_IP $CUST_IP
```

Preferably using a test-case you know fails, to see effect

Still amazed how often this works

Test-cases / Scenarios, continued Small Fragments

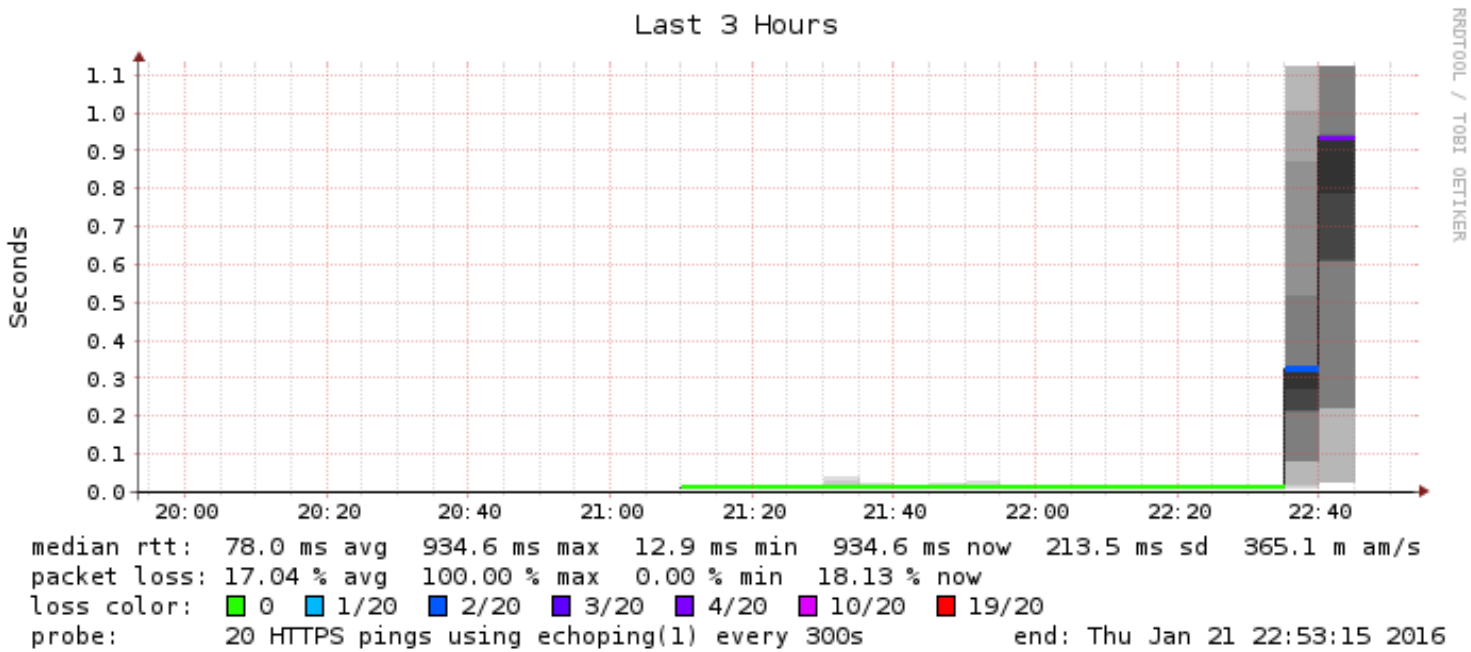
Using the built-in option -f for hping

```
-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.
```

```
hping3 -q --flood -p 80 -S -f $CUST_IP
```

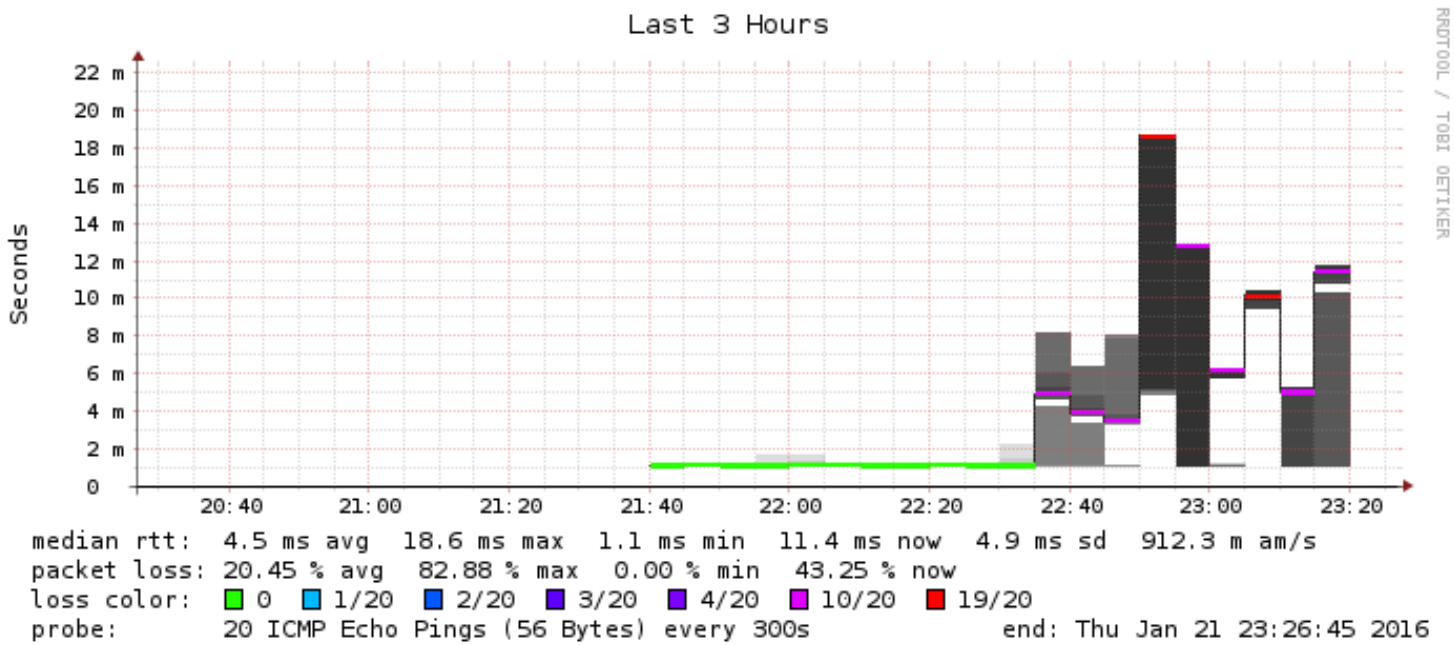
Similar process with bad checksum and Bad fragment offset

Rocky Horror Picture Show - 1



Really does it break from 50.000 pps SYN attack?

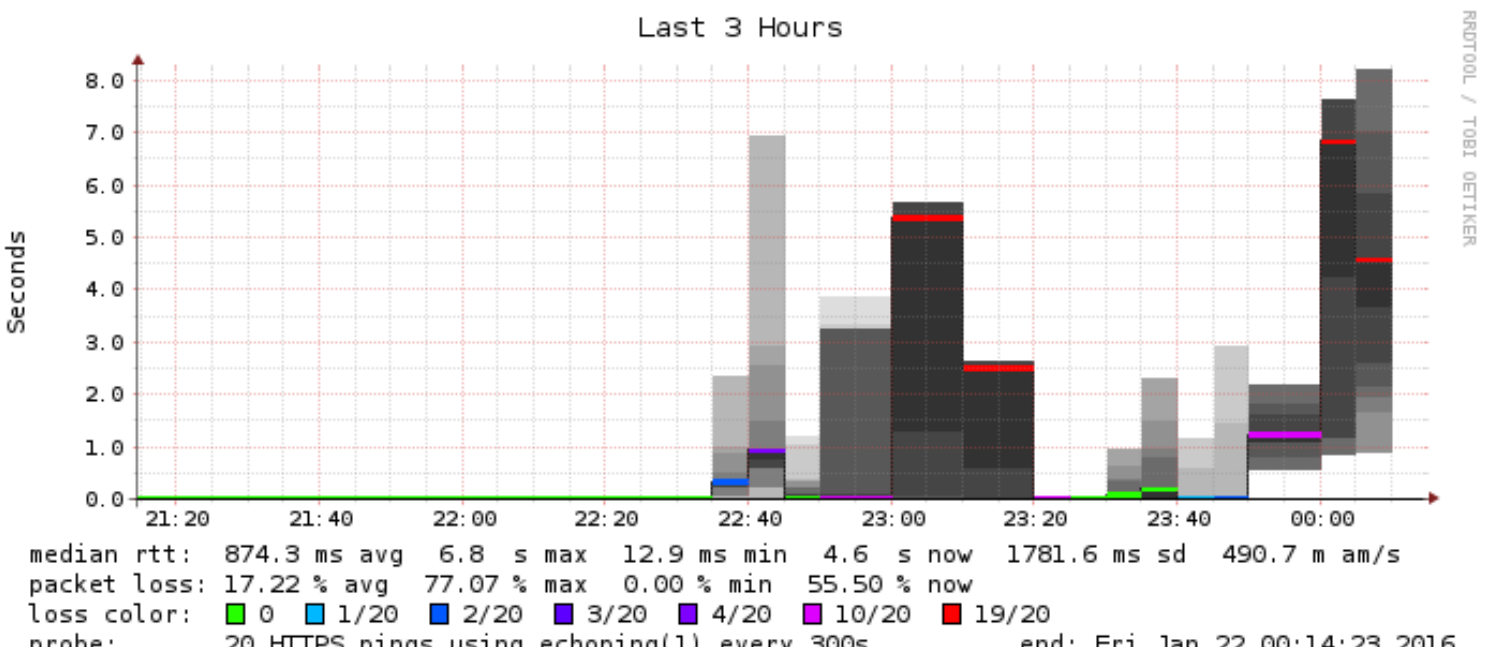
Rocky Horror Picture Show - 2



Oh no 500.000 pps UDP attacks work?

Rocky Horror Picture Show - 3

Oh no spoofing attacks work?



Advanced and High Performance Testing

- Hping is not the fastest tool, which is fine when we don't want full speed
- I can produce millions of packets, but it requires multiple CPU cores with Hping
- We DO want to test maximum speed at some point, full 10Gbit and 14.8Million pps (Mpps)
- Modern CPUs (for many years) support methods for sending and receiving high speed
- Data Plane Development Kit (DPDK) is an open source software project which is very popular in this space https://en.wikipedia.org/wiki/Data_Plane_Development_Kit

Enter MoonGen and Dedicated Hardware

- Modern computer with modern CPU and PCIe x8 or better
  I currently use a few cheap Dell devices Precision 3640 Tower / Precision 3240 Compact (not a recommendation)
- Supported card - I am using the old Intel 82599 based 10Gbit cards
- DPDK and software – I use MoonGen https://github.com/emmericp/MoonGen
- A huge thanks to Paul Emmerich emmericp for programming and publishing his works!
- Maybe the easiest way to use DPDK currently – "Craft all packets in user-controlled Lua scripts"

Running MoonGen

```
root@penguin01:~/projects/MoonGen#  ./build/MoonGen ./examples/l3-tcp-syn-flood.lua 0 -d 192.0.2.138
[INFO]  Initializing DPDK. This will take a few seconds...
EAL: Detected 16 lcore(s)
[INFO]  Found 1 usable devices:
   Device 0: 00:25:90:32:9F:F3 (Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection)
PMD: ixgbe_dev_link_status_print():  Port 0: Link Down
[INFO]  Device 0 (00:25:90:32:9F:F3) is up: 10000 MBit/s
[INFO]  Detected an IPv4 address.
...
[Device: id=0] TX: 14.88 Mpps, 7619 Mbit/s (9999 Mbit/s with framing)
[Device: id=0] TX: 14.48 Mpps, 7414 Mbit/s (9730 Mbit/s with framing)
[Device: id=0] TX: 14.88 Mpps, 7619 Mbit/s (10000 Mbit/s with framing)
```

- Installed Debian Linux – little bit of disable secure boot, RAID/AHCI settings, ...
- After install – tuning and enabling Hugepages
- Clone the repository https://github.com/emmericp/MoonGen build and run
- **Note: the full 14.8Mpps is done using a single core!**

Turn up and down as you please with the -r rate option

```
root@penguin01:~/projects/MoonGen#  ./build/MoonGen ./examples/l3-tcp-syn-flood.lua 0 -r 5000 -d 192.0.2.138
[INFO]  Initializing DPDK. This will take a few seconds...
EAL: Detected 16 lcore(s)
[INFO]  Found 1 usable devices:
   Device 0: 00:25:90:32:9F:F3 (Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection)
PMD: ixgbe_dev_link_status_print():  Port 0: Link Down
[INFO]  Device 0 (00:25:90:32:9F:F3) is up: 10000 MBit/s
[INFO]  Detected an IPv4 address.

[Device: id=0] TX: 9.77 Mpps, 5000 Mbit/s (6562 Mbit/s with framing)
[Device: id=0] TX: 9.68 Mpps, 4955 Mbit/s (6504 Mbit/s with framing)
[Device: id=0] TX: 9.77 Mpps, 5000 Mbit/s (6562 Mbit/s with framing)
```

IPv6 and UDP, replace tcp with udp in example:

```
./build/MoonGen ./examples/l3-tcp-syn-flood.lua 0 -r 5000 -d 2001:DB8:ABCD:0053::138 -i 2001:DB8:ABCD:0053::1

./build/MoonGen ./examples/l3-udp-flood-hlk.lua 0 -r 5000 -d 2001:DB8:ABCD:0053::138 -i 2001:DB8:ABCD:0053::1
```

Penguinping – re-implementing hping3 with Lua

```
root@penguin01:~/projects/MoonGen#  ./build/MoonGen ./examples/penguinping-02.lua 10.0.49.1 -a 10.1.2.3 -r 10000 -S -p 80
[INFO]  Initializing DPDK. This will take a few seconds...
[INFO]  Found 2 usable devices:
   Device 0: 00:25:90:32:9F:F2 (Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection)
[INFO]  Device 0 (00:25:90:32:9F:F2) is up: 10000 MBit/s
TCP mode get TCP packet
IP4 10.1.2.3 > 10.0.49.1 ver 4 ihl 5 tos 0 len 46 id 0 flags 0 frag 0 ttl 64 proto 0x06 (TCP) cksum 0x0000 [-]
TCP 52049 > 80 seq# 1 ack# 0 offset 0x5 reserved 0x00 flags 0x02 [-|-|-|-|SYN|-] win 10 cksum 0x0000 urg 0 []
  0x0000:   0000 0000 0000 0025 9032 9ff2 0800 4500
  0x0010:   002e 0000 0000 4006 0000 0a01 0203 0a00
  0x0020:   3101 cb51 0050 0000 0001 0000 0000 5002
  0x0030:   000a 0000 0000 0000 0000 0000

[Device: id=0] TX: 14.88 Mpps, 7619 Mbit/s (9999 Mbit/s with framing)
[Device: id=0] TX: 14.78 Mpps, 7568 Mbit/s (9933 Mbit/s with framing)
[Device: id=0] TX: 14.88 Mpps, 7619 Mbit/s (10000 Mbit/s with framing)
```

- Using Lua we can implement the same attacks from Hping3 easily
- Only about 230 lines of Lua using MoonGen and LibMoon
- Can run at specific rate up to full 10Gbps / 14.8 Million packets per second using a single CPU core

Comparable to real DDoS?

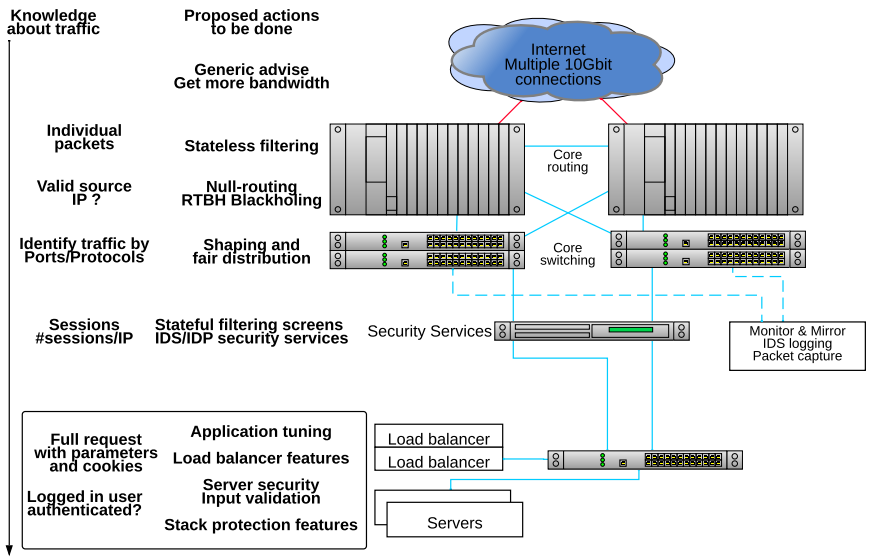
Tools are simple and widely available but are they actually producing same result as high-powered and advanced criminal botnets. We can confirm that the attack delivered in this test is, in fact, producing the traffic patterns very close to criminal attacks in real-life scenarios.

- We can also monitor logs when running a single test-case
- Gain knowledge about supporting infrastructure
- Can your syslog infrastructure handle 800.000 events in $<$ 1 hour?

Main difference are that attackers are free to switch attack types and mix them. While we try specifically to keep using one type, to see the worst and which ones that hurt the most.

I also start at the bottom, and work my way up – while an attacker may begin attacking HTTP/HTTPS directly.

Protection – Enable More Packet filtering



- Packet filtering can be done one single packets – stateless filtering
- We can save information about direction and ongoing traffic – stateful filtering/firewalling
- *Filtering* can also be setting a maximum number of packets for a protocol – rate limit by protocol

Designing the protection – bandwidth and rate limit

Protocol	Mbps	Prefix
TCP	Up to full bandwidth 10Gbps	192.0.2.0/25
UDP	Less than 1Gbps	192.0.2.128/25
ICMP	Less than 10Mbps	192.0.2.0/24

- Create an address plan for your services
- Monitor your traffic – how much UDP and TCP do you have, roughly
- Above is a simplified example – dig deeper into your traffic

Designing the protection – address families & protocols

Address family	Protocol	Services and ports	Prefix
IPv4	TCP	25, 80, 8003, 443, 4443	192.0.2.0/25
IPv4	UDP	53	192.0.2.128/25
IPv6	UDP	53	2001:DB8:ABCD:0053::/60
IPv6	TCP	80 443	2001:DB8:ABCD:1000::/56

- Direction is also very important – servers that never initiate connections have fewer requirements
- How much traffic do you have that uses IPv6 yet? Should an IPv6 DDoS take up all resources?
- Maybe let IPv4 only use a part, so at least some customers can visit using IPv6?
- Maybe you can do an allow list for allocated networks, since not all is used yet

Stateless firewall filter limit protocols

```
term limit-icmp {
    from {
        protocol icmp;
    }
    then {
        policer ICMP-100M;
        accept;
    }
}
term limit-udp {
    from {
        protocol udp;
    }
    then {
        policer UDP-1000M;
        accept;
    }
}
```

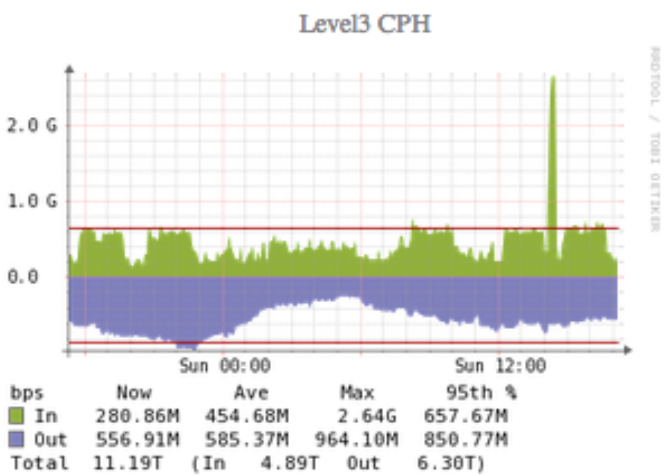
Routers also have extensive Class-of-Service (CoS) tools today, and in general rate limiting stuff is nice

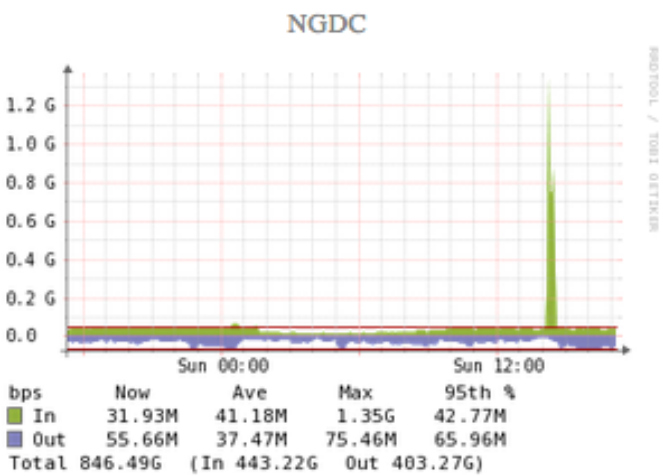
Strict filtering for some servers, still stateless!

```
term some-server-allow {
    from {
        destination-address {
            192.0.2.0/25;
        }
        protocol tcp;
        destination-port [ 25 80 8003 443 4443 ];
    } then accept;
}
term some-server-block-unneeded {
    from {
        destination-address {
            192.0.2.0/25; }
        protocol-except icmp;  }
    then { count some-server-block; discard;
    }
}
```

Wut - no UDP, yes only TCP service is used on these servers

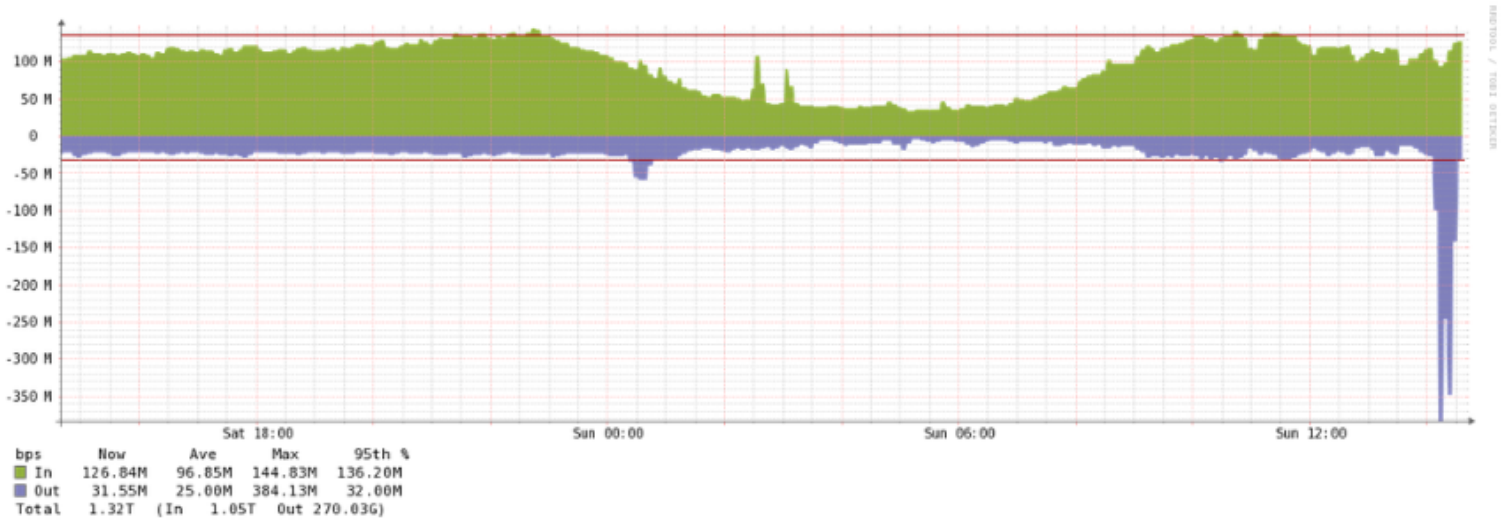
Results from implementing – DDoS traffic before filtering





Only two links shown, at least 3Gbit incoming for this single IP

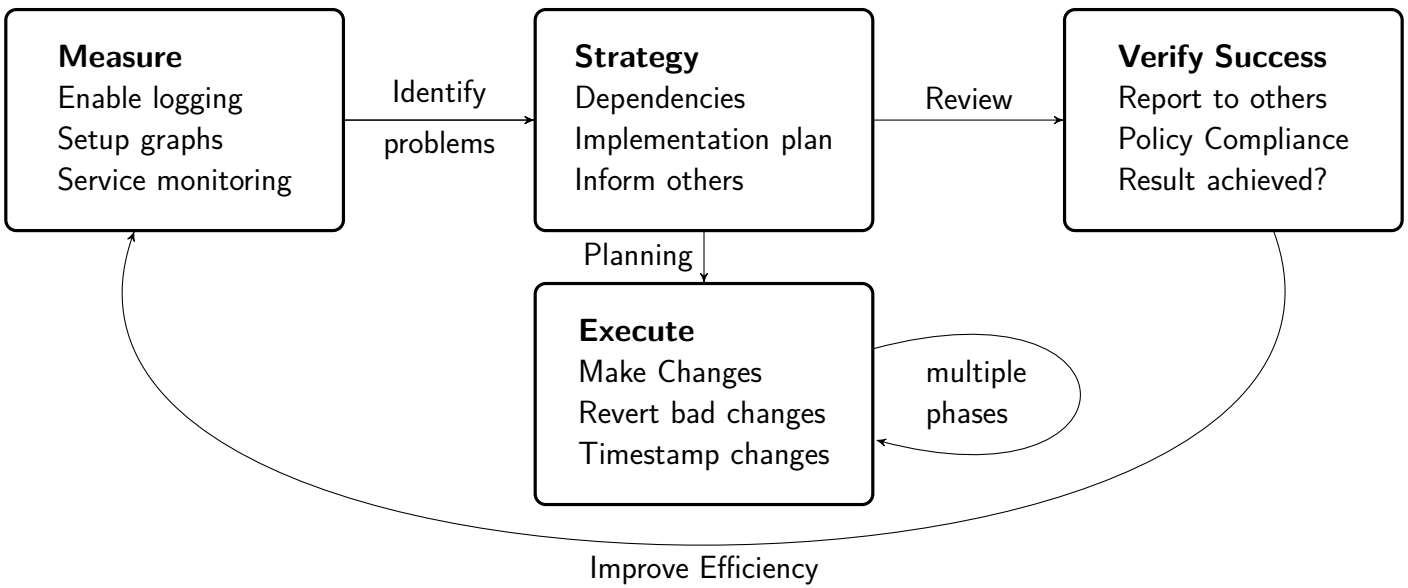
DDoS traffic after filtering



Link toward server (next level firewall actually) about 350Mbit outgoing

Knowing what it going on, is half the battle

Make incremental changes



Improvements seen after testing

Turning off unneeded features - free up resources

Tuning sesions, max sessions src / dst

Tuning firewalls, max sessions in half-open state, enabling services

Tuning network, drop spoofed src from inside net ☺

Tuning network, can follow logs, manage network during attacks

...

And organisation has better understanding of DDoS challenges

Including vendors, firewall consultants, ISPs etc.

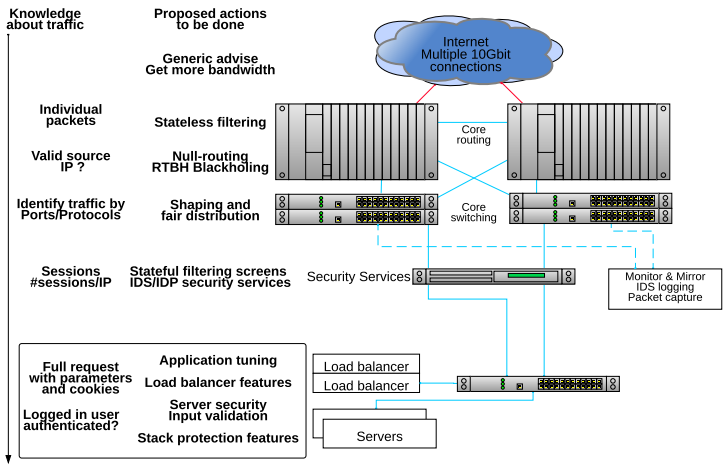
After tuning of **existing devices/network** improves results 10-100 times

Interesting findings

```
| snmp-info:
|   enterprise: ciscoSystems
|   engineIDFormat: mac
|   engineIDData: 00:08:4f:xx:yy:zz
|   snmpEngineBoots: 4
|_  snmpEngineTime:  732d07h09m04s
```

- Customers have ISPs which have routers with 1800 days uptime
  Seen using SNMP – so missing critical updates, and we can affect performance
- Even big companies still have 1Gbit internet connections, recommend getting 10Gbps
- Most customers fail at least 3 scenarios which can be countered *without loss of functionality*
  Example we used a stateless filter to only allow TCP traffic towards a /24 with web services (which don't need or want 10Gbps ICMP/UDP)
- Ohh we lost our VPN into the environment, ohh the fw console is dead

Conclusion DDoS and network attacks



- You really should try testing, and investigate your existing devices all of them
- Choose which devices does which part – discard early to free resources for later devices to dig deeper
- This is just one small part of your security posture, extra slides has my take on enterprise network security

Questions?

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

You are always welcome to send me questions later via email

Mobile: +45 2026 6000

Thank you for coming. I'll be around until friday. I will try to gather this on a project web site https://penguinping.org/

Concrete advice for enterprise networks

- Portscanning - start using portscans in your networks, verify how far malware and hackers can travel, and identify soft systems needing updates or isolation
- Have separation – anywhere, starting with organisation units, management networks, server networks, customers, guests, LAN, WAN, Mail, web, ...
- Use Web proxies - do not allow HTTP directly except for a short allow list,
  do not allow traffic to and from any new TLD
- Use only your own DNS servers, create a pair of Unbound servers,
  point your internal DNS running on Windows to these
  Create filtering, logging, restrictions on these Unbound DNS servers
  https://www.nlnetlabs.nl/projects/unbound/about/ and also https://pi-hole.net/
- Only allow SMTP via your own mail servers, create a simple forwarder if you must

Allow lists are better than block list, even if it takes some time to do it

Capture data and logs!

- Run DNS query logs – when client1 is infected with malware from domain malwareexample.com, then search for more clients i nfected
- Run Zeek and gather information about all HTTPS sessions – captures certificates by default, and we can again search for certificate related to malwareexample.com
- Run network logging – session logs in enterprise networks are GREAT
  (country wide illegal logging is of course NOT)

  Make sure to check with employees, inform them!

DROP SOME TRAFFIC NOW

- Drop some traffic on the border of everything
- Seriously do NOT allow Windows RPC across borders
- Border here may be from regional country office back to HQ
- Border may be from internet to internal networks
- Block Windows RPC ports, 135, 137, 139, 445
- Block DNS directly to internet, do not allow clients to use any DNS, fake 8.8.8.8 if you must internally
- Block SMTP directly to internet
- Create allow list for internal networks, client networks should not contact other client networks but only relevant server networks

You DONT need to allow direct DNS towards internet, except from your own recursive DNS servers

If you get hacked by Windows RPC in 2022, you probably deserve it, sorry for being blunt

Best would be to analyze traffic and create allow lists, some internal networks to not need internet at all

Stateless firewall filter throw stuff away

Example how to do it wirespeed – with Junos

```
hlk@MX-CPH-02> show configuration firewall filter all | no-more
/* This is a sample, better to use BGP flowspec or BGP based RTBH */
term edgeblocker {
    from {
        source-address {
            84.xx.xxx.173/32;
...
            87.xx.xxx.171/32;
        }
        destination-address {
            192.0.2.16/28;
        }
        protocol [ tcp udp icmp ];
    }
    then {
        count edge-block;
        discard;
    }
}
```

Hint: can also leave out protocol and then it will match all protocols

Default permit

One of the early implementers of firewalls Marcus J. Ranum summarized in 2005 The Six Dumbest Ideas in Computer Security https://www.ranum.com/security/computer_security/editorials/dumb/ which includes the always appropriate discussion about default permit versus default deny.

> **#1) Default Permit**
> This dumb idea crops up in a lot of different forms; it's incredibly persistent and difficult to eradicate. Why? Because it's so attractive. Systems based on "Default Permit" are the computer security equivalent of empty calories: tasty, yet fattening.
> The most recognizable form in which the "Default Permit" dumb idea manifests itself is in firewall rules. Back in the very early days of computer security, network managers would set up an internet connection and decide to secure it by turning off incoming telnet, incoming rlogin, and incoming FTP. Everything else was allowed through, hence the name "Default Permit." This put the security practitioner in an endless arms-race with the hackers.

- Allow all current networks today on all ports for all protocols *is* an allow list
  Which tomorrow can be split into one for TCP, UDP and remaining, and measured upon
- Measure, improve, repeat

We cannot do X

> We cannot block SMTP from internal networks, since we do not know for sure if vendor X equipment needs to send the MOST important email alert at some unspecific time in the future

Cool, then we can do an allow list starting today on our border firewall:

```
table <smtp-exchange> { $exchange1 $exchange2 $exchange3 }
table <smtp-unknown> persist file "/firewall/mail/smtp-internal-unknown.txt"
# Regular use, allowed
pass out on egress inet proto tcp from smtp-echange to any port 25/tcp
# Unknown, remove when phased out
pass out on egress inet proto tcp from smtp-internal to any port 25/tcp
```

Year 0 the unknown list may be 100% of all internal networks, but new networks added to infrastructure are NOT added, so list will shrink – evaluate the list, and compare to network logs, did networks send ANY SMTP for 1,2,3 years?

Zeek is a framework and platform

The Zeek Network Security Monitor

While focusing on network security monitoring, Zeek provides a comprehensive platform for more general network traffic analysis as well. Well grounded in more than 15 years of research, Zeek has successfully bridged the traditional gap between academia and operations since its inception.

https://www.Zeek.org/ Does useful things out of the box using more than 10.000 script lines

Suricata IDS/IPS/NSM

Suricata is a high performance Network IDS, IPS and Network Security Monitoring engine.

http://suricata-ids.org/ http://openinfosecfoundation.org

Suricata, Zeek og DNS Capture – it a nice world, use it!

https://github.com/kramse/security-courses/tree/master/courses/networking/suricatazeek-workshop

Firewall – Another definition

I am also fond of this longer and technical definition from RFC4949:

> $ firewall
>
> 1. (I) **An internetwork gateway that restricts data communication traffic to and from one of the connected networks** (the one said to be "inside" the firewall) and thus protects that network's system resources against threats from the other network (the one that is said to be "outside" the firewall). (See: guard, security gateway.)
>
> 2. (O) **A device or system that controls the flow of traffic between networks using differing security postures.** Wack, J. et al (NIST), "Guidelines on Firewalls and Firewall Policy", Special Publication 800-41, January 2002.
>
> Tutorial: A firewall typically protects a smaller, secure network (such as a corporate LAN, or even just one host) from a larger network (such as the Internet). The firewall is installed at the point where the networks connect, and the firewall applies policy rules to control traffic that flows in and out of the protected network.

Firewall – Another definition

$ firewall, continued

A firewall is not always a single computer. For example, a firewall may consist of a pair of filtering routers and one or more proxy servers running on one or more bastion hosts, all connected to a small, dedicated LAN (see: buffer zone) between the two routers.

The external router blocks attacks that use IP to break security (IP address spoofing, source routing, packet fragments), while proxy servers block attacks that would exploit a vulnerability in a higher-layer protocol or service. The internal router blocks traffic from leaving the protected network except through the proxy servers.

The difficult part is defining criteria by which packets are denied passage through the firewall, because a firewall not only needs to keep unauthorized traffic (i.e., intruders) out, but usually also needs to let authorized traffic pass both in and out.

Routing Security

- Use MD5 passwords or better authentication for routing protocols 🚩
- TTL Security – avoid routed packets
- Max prefix – of course, only allow expected networks
- Prefix filtering – only parts of IPv6 space is used
- TCP Authentication Option [RFC 5925] replaces TCP MD5 [RFC 2385]
- Turn ON RPKI for both IPv4 and IPv6 prefixes, 🚩
  https://nlnetlabs.nl/projects/rpki/about/
- Drop bogons on IPv4 and IPv6, article with multiple references YMMV
  https://theinternetprotocolblog.wordpress.com/2020/01/15/some-notes-on-ipv6-bogon-filtering/

Mutually Agreed Norms for Routing Security (MANRS)

> Mutually Agreed Norms for Routing Security (MANRS) is a global initiative, supported by the Internet Society, that provides crucial fixes to reduce the most common routing threats.

Source: https://www.manrs.org/wp-content/uploads/2018/09/MANRS_PDF_Sep2016.pdf

- Problems related to incorrect routing information
- Problems related to traffic with spoofed source IP addresses
- Problems related to coordination and collaboration between network operators
- Also BCP38 RFC2827 *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*

You should all ask your internet providers if they know about MANRS, and follow it. We should ask our government and institutions to support and follow MANRS and good practices for network on the Internet

uRPF unicast Reverse Path Forwarding

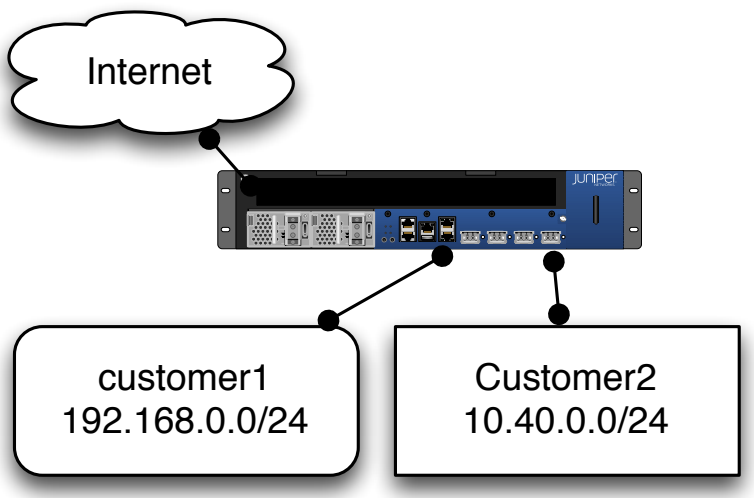
> Reverse path forwarding (RPF) is a technique used in modern routers for the purposes of ensuring loop-free forwarding of multicast packets in multicast routing and to help prevent IP address spoofing in unicast routing.

Source: http://en.wikipedia.org/wiki/Reverse_path_forwarding

> **Configuring Unicast RPF Strict Mode**
> In strict mode, unicast RPF checks whether the incoming packet has a source address that matches a prefix in the routing table, **and whether the interface expects to receive a packet with this source address prefix.**

This means, random source spoofed packets – not matchin a current route entry in the global table, will get discarded early on!

Strict vs loose mode RPF




```
user@router# show interfaces
ge-0/0/0 {
        unit 2 {
                family inet {
                        rpf-check fail-filter rpf-special-case-dhcp;
                        address 192.168.0.254/24;
                }
        }
}
ge-0/0/1 {
        unit 2 {
                family inet {
                        rpf-check fail-filter rpf-special-case-dhcp;
                        address 10.40.0.254/24;
                }
        }
}
```