New Features in the SI6 Networks' IPv6 Toolkit

Fernando Gont

IPv6 Security Summit 2015
Heidelberg, Germany. March 16-17, 2015

About...

- Security Researcher and Consultant at SI6 Networks
- Published:
  - 20 IETF RFCs (9 on IPv6)
  - 10+ active IETF Internet-Drafts
- Author of the SI6 Networks' IPv6 toolkit
  - http://www.si6networks.com/tools/ipv6toolkit
- Admin of the IPv6 Hackers mailing-list
  - ipv6hackers@lists.si6networks.com
- More information at: http://www.gont.com.ar

Agenda

"I've never met anybody who really did spend blood on something who wasn't eager to describe what they've done and how they did it and why"

-- Ken Thompson (in "Coders at Work: Reflections on the Craft of Programming")

This talk is about new features in the

SI6 Network's IPv6 Toolkit

Introduction

SI6 Networks' IPv6 Toolkit: Intro

- Brief history:
  - Produced as part of a project funded by UK CPNI on IPv6 security
  - Maintenance and extension taken over by SI6 Networks
- Goals:
  - Security analysis and trouble-shooting of IPv6 networks and implementations
  - Clean, portable, and secure code
  - Good documentation

SI6 Networks' IPv6 Toolkit: Intro (II)

- Supported OSes:
  - Linux, FreeBSD, NetBSD, OpenBSD, Mac OS, and OpenSolaris
- License:
  - GPL (free software)
- Home:
  - http://www.si6networks.com/tools/ipv6toolkit
- Collaborative development:
  - https://www.github.com/fgont/ipv6toolkit.git

SI6 Networks' IPv6 Toolkit: Philosophy

IDEAS TOOLS IPV6 NETWORK

"*an interface between your brain and your IPv6 network*"

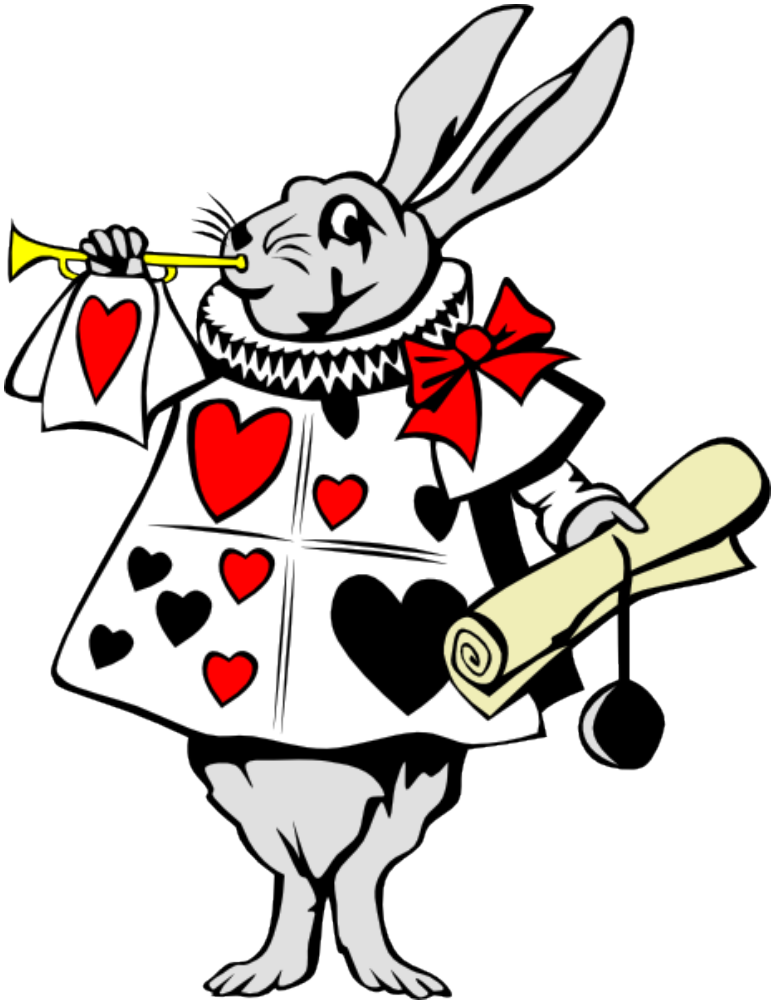
Some find this is NOT a useful approach, though! ☺

SI6 Networks' IPv6 toolkit: Tools

- addr6: An IPv6 address analysis tool
- scan6: An IPv6 address scanner
- path6: A versatile IPv6-based traceroute
- frag6: Play with IPv6 fragments
- tcp6: Play with IPv6-based TCP segments
- udp6: Play with UDP datagrams
- ns6: Play with Neighbor Solicitation messages
- na6: Play with Neighbor Advertisement messages
- script6: Rather complex tasks made easy

SI6 Networks' IPv6 toolkit: Tools (II)

- rs6: Play with Router Solicitation messages
- ra6: Play with Router Advertisement messages
- rd6: Play with Redirect messages
- icmp6: Play with ICMPv6 error messages
- ni6: Play with Node Information messages
- flow6: Play with the IPv6 Flow Label
- jumbo6: Play with IPv6 Jumbograms



IPv6 Toolkit v2.0!

SI6 NETWORKS

Overview

What's new in SI6 IPv6 v2.0 (Guille)

- Lots of bug fixes!
  - You MUST update :-)
- An additional supported platform
  - OpenSolaris
- New tools:
  - **`script6`**
  - **`blackhole6`**
  - **`udp6`**
- New features:
  - **`tcp6`**'s --close-mode, --data, etc.
  - **`scan6`**'s automatic smart scanning

Address Scanning

Address Scanning

- scan6 is **the most comprehensive IPv6 address scanner**
- It now supports heuristic address scanning:
  - It automatically detects address patterns
  - Then automatically targets such address patterns
- Employing heuristic scanning:

```
scan6 -d DOMAIN/64

scan6 -d IPV6ADDR/64
```

Host Scanning Demo

IPv6-base TCP/UDP port scanning

- scan6 incorporates all known TCP and UDP port-scanning techniques
- Specifying a protocol and port range:

  `--port-scan {tcp,udp}:port_low[-port_hi]`

- Specifying a TCP scan type:

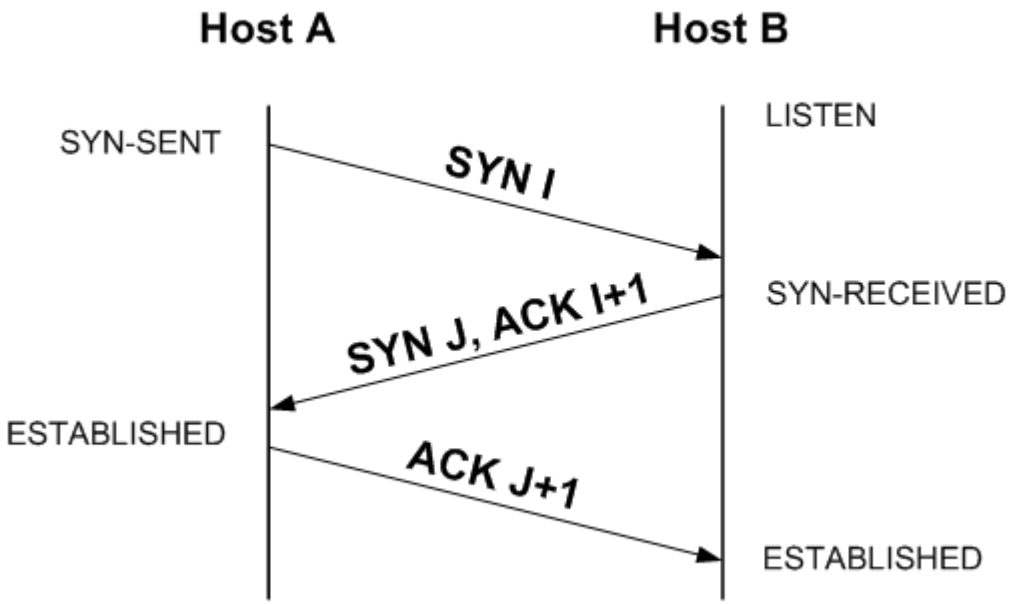
  `--tcp-scan-type {syn,fin,null,xmas,ack}`

- Example:

  `--port-scan tcp:1-1024 --tcp-scan-type syn`

TCP port scanning: Intro/Overview

- TCP connection-establishment in a nutshell:



TCP port scanning: connect() scan

- Implements the full 3WHS
- Slow (requires two RTTs)
- Notifies the target application of the communication attempt
- Ties resources on both ends of the connection
- **Not implemented in scan6**

TCP port scanning: SYN scan

- Does not implement the full 3WHS
  - Send a SYN, process response packet
  - SYN/ACK= Open, RST= Closed
- It is fast
- Does not tie resources on our end
- **Implemented in scan6**

TCP port scanning: FIN, NULL, and XMAS

- Does not implement the full 3WHS
  - Send a packet without A bit set, wait for response
  - RST= Closed, Timeout= Open
- It is rather slow (need to wait for a timeout)
- Does not tie resources on an side
- **Implemented in scan6**

Port Scanning Demo

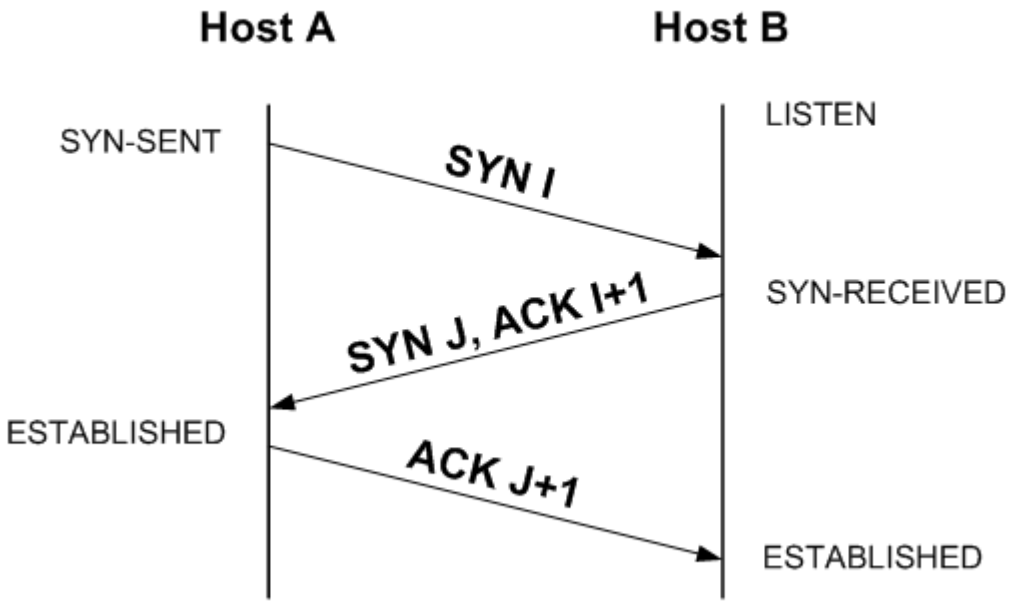
Playing with TCP Packets

tcp6: Introduction

- Tool originally developed out of “frustration”
  - There was not even an IPv6-based SYN flooder
- But continued as a kind of nice *deja vu*
  - My early work on protocols involved TCP
  - IPv4-based TCP attack tools were/are rather rudimentary

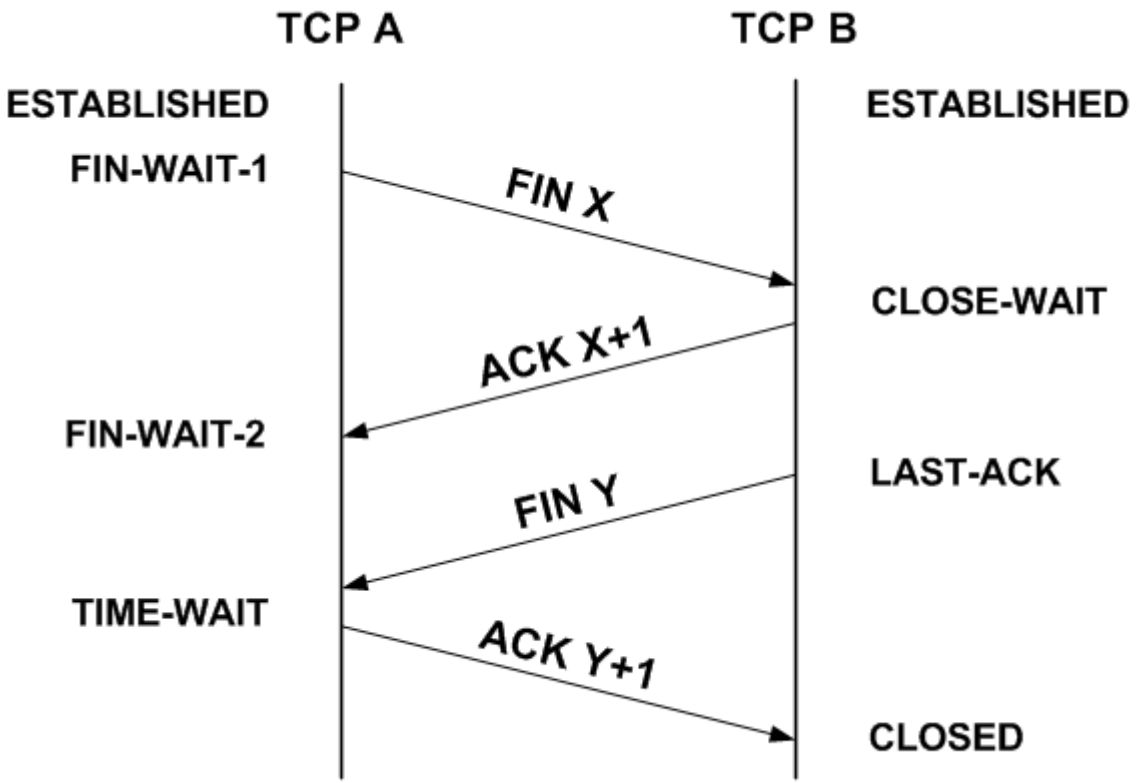
TCP fundamentals: Conn. Establishment

- Typically involves three segments:



TCP fundamentals: Conn. Termination

- Typically involves four segments:



TCP fundamentals: State

- Some TCP states are typically handled by the kernel, e.g.:
  - SYN-RECEIVED
  - FIN-WAIT-2
  - TIME-WAIT
- Others are “visible” by the application, e.g.:
  - ESTABLISHED
  - CLOSE-WAIT
- This affects who can mitigate a variety of DoS attacks

tcp6: Open Mode

- tcp6 can force specific “connection-establishment” sequences:
  - **active**: We initiate the connection with a SYN
  - **passive**: We listen for incoming connections
  - **simultaneous**: Crossing SYN segments
  - **abort**: We reject incoming connections with an RST
- Open mode is selected with:

  `--open-mode`

tcp6: Close Mode

- tcp6 can force specific “connection-termination” sequences:
  - simultaneous: Force crossing FINs
  - passive: Wait for the other end to initiate connection-termination phase
  - active: Start the connection-termination phase
  - abort: Reset the connection
  - FIN-WAIT-1: Cause remote TCP to remain in FIN-WAIT-1 state
  - FIN-WAIT-2: Cause remote TCP to remain in FIN-WAIT-2 state
  - LAST-ACK: Cause remote TCP to remain in LAST-ACK-1 state
- Close mode is selected with:

  `--close-mode`

tcp6: Connection flooding attacks

- SYN-floods:

```
tcp6 -i IFACE -s SRCPRF -d TARGET -a DSTPORT -X
S -F 100 -l -z 1 -v
```

- Connection floods:

```
tcp6  -i  IFACE  -s SRCPRF -d TARGET -a DSTPORT
-L -l --flood-sources 10 -z 1 --tcp-flags auto
-v
```

- Other connection-floods:

```
sudo tcp6 -i IFACE -s SRCPREF -d TARGET -a
DSTPORT -L -l --flood-sources 10 -z 1 --tcp-
flags auto -v --close-mode last-ack
```

IPv6-based Netkill

- Shalunov devised a very nasty attack against TCP:
  - Establish a large number of (throw-away) TCP connections
  - Send a request (e.g., HTTP GET for a long file)
  - Close the receive window
- The net effect is that resources are wasted on:
  - TCBs (as for any connection)
  - TCP send buffers

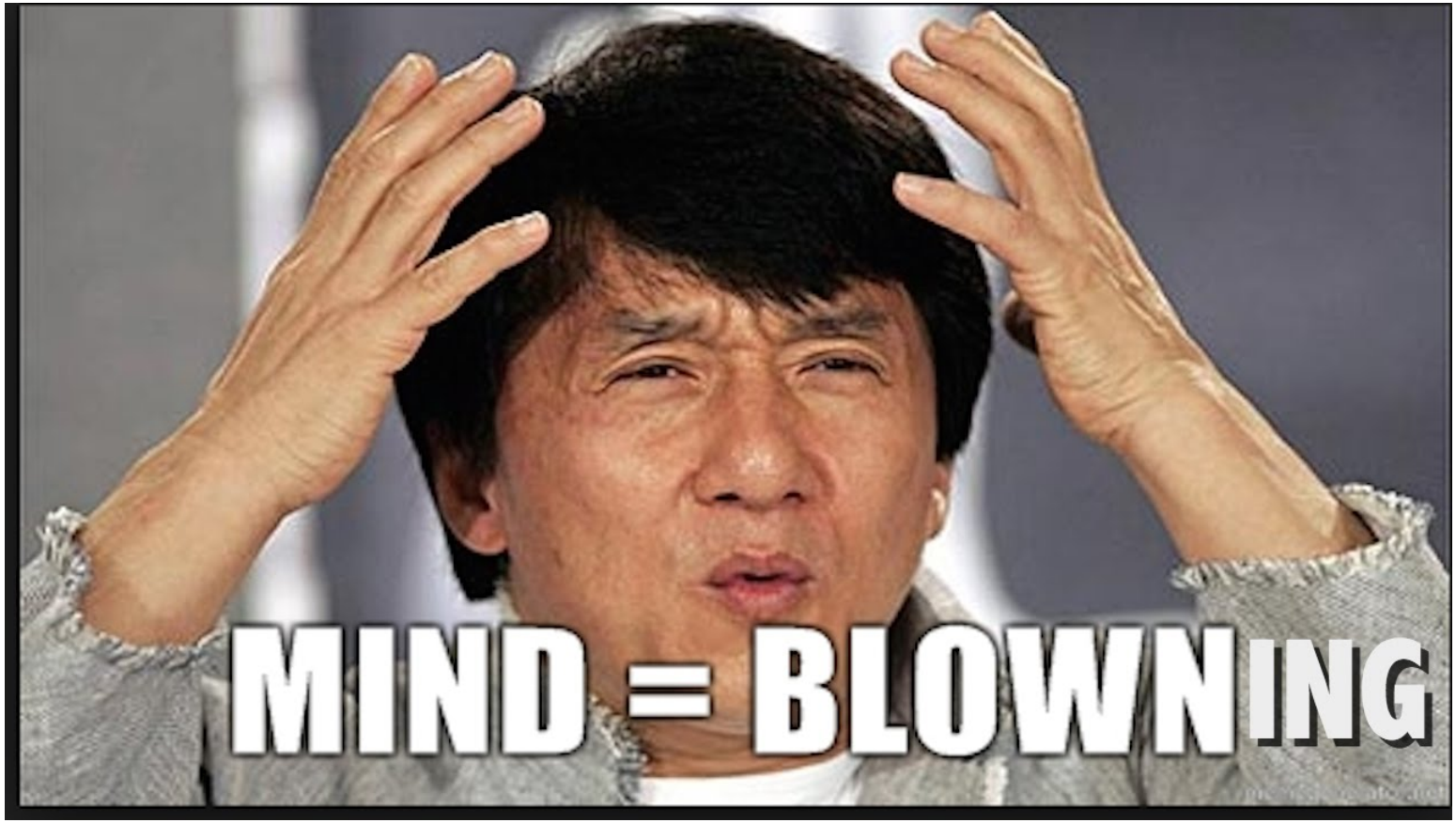
IPv6-based Netkill with tcp6

- TCP payloads can be sent with the **`--data`** command
- A Netkill attack could be implemented as:

```
tcp6 -i eth0 -d TARGET_IPV6 -a 80 -L -s
SOURCE_PREF -l -r 1pps --tcp-flags auto -v
--data "GET / HTTP/1.0\r\n\r\n" --flood-ports
10 --window-mode close
```

Playing with TCP Packets Demo

More about TCP-based attacks?



http://www.gont.com.ar/papers/tn-03-09-security-assessment-TCP.pdf

Playing with UDP datagrams

udp6: Play with UDP datagrams

- Can send arbitrary IPv6-based UDP datagrams
  - Use EHs
  - Flood a specific endpoint with datagrams from different sources and ports
  - Supports customized filters
  - Supports a **`--data`** option to embed a payload
- New in SI6 Networks IPv6 toolkit v2.0 (Guille)

Get interesting addresses

Get domains and IPv6 addresses

- **`script6`** can do batch-processing of domain names
- Available commands:
  - **`get-aaaa`**
  - **`get-mx`**
  - **`get-ns`**

Get domains and IPv6 addresses (II)

- Get mailserver domains:

```
$ cat domains.txt | script6 get-mx
```

- Get IPv6 addresses:

```
$ cat domains.txt | script6 get-aaaa
```

- Get mailserver addresses:

```
$ cat domains.txt | script6 get-mx | script6 get-aaaa
```

Get interesting addresses Demos

Obtaining AS-related info

Obtaining AS-related info

- Given an IPv6 address, the corresponding AS identifies the corresponding organization, e.g.
  - who should I contact when an IPv6 address is attacking me?
  - who should I contact when a given router is dropping my packets?
- script6 can query AS-related information:

```
script6 get-as

script6 get-asn
```

Obtaining AS-related info Demo

Tracing IPv6 Routes

path6 tool

- No existing traceroute tool supported IPv6 extension headers
  - e.g., How far do your IPv6 EH-enabled packets get?
- Hence we produced our path6 tool
  - Supports IPv6 Extension Headers
  - Can employ TCP, UDP, or ICMPv6 probes
  - It's faster ;-)
- Example:

```
# path6 -u 100 -d fc00:1::1
```

Dst Opt Hdr

Tracing IPv6 Routes Demo

Finding IPv6 blackholes

blackhole6: Finding IPv6 blackholes

- It is useful to find out who is dropping specific packets:
  - Troubleshooting
  - Network reconnaissance
  - ... or just checking if you EH-enabled attacks would work
- blackhole6 does this (and more) auto-magically:

```
blackhole6 DESTINATION [EHTYPE[EHSIZE]]
[PROTOCOL [PORT]]]
```

blackhole6: Methodology

1) Run “normal” path6 to target (D), and save route (ROUTE)
2) Check that last “hop” in route is D
3) Run EH-enabled path6, and find last responding address (M)
4) Find “M” in “ROUTE” -> dropping system is next in ROUTE (M+1)
5) Compare AS(M) with AS(M+1), and produce other stats

blackhole6: Methodology (II)

- Given the output of path6 for no-EH and EHs:

	No EHs	With EHs
	1. fc00:1:1:1000::1	1. fc00:1:1:1000::1
	2. fc00:1:1:2000::4	2. fc00:1:1:2000::4
	3. fc00:1:2:4000::1	3. fc00:1:2:4000::1
	4. fc00:2:1:4000::1	4. fc00:2:1:4000::1
	5. fc00:a:2:1000::1	5. fc00:a:2:1000::1
	6. fc00:a:4:4000::1	6. fc00:a:4:4000::1
DROP	7. fc00:b:1:1000::1	
	8. fc00:b:2:5000::1	
	9. fc00:b:4:5000::1	
	10. fc00:d::1	

blackhole6: Methodology (III)

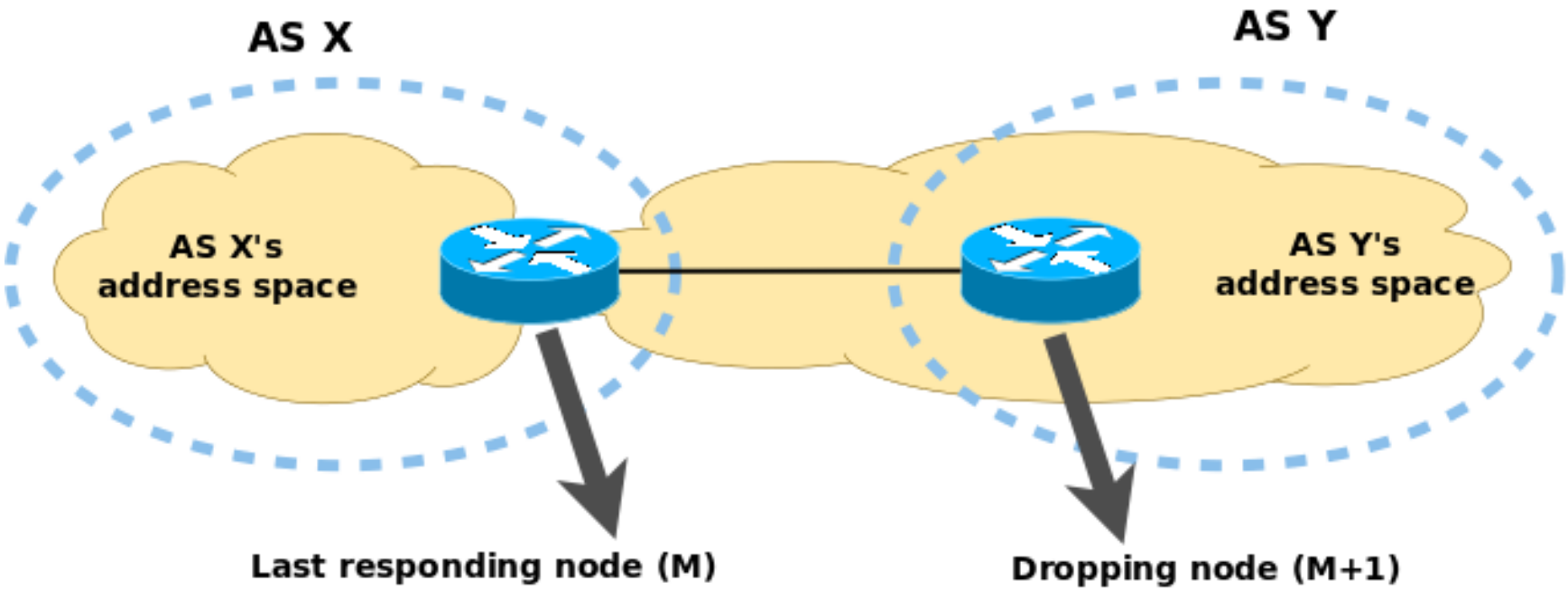
- We assume ingress filtering...
- Otherwise dropping node actually is M rather than M+1

blackhole6: ASes

- Lookup ASN of dropping node, but...
- There may be ambiguity when finding the AS of the dropping node:
  - who provides the address space for the peering?

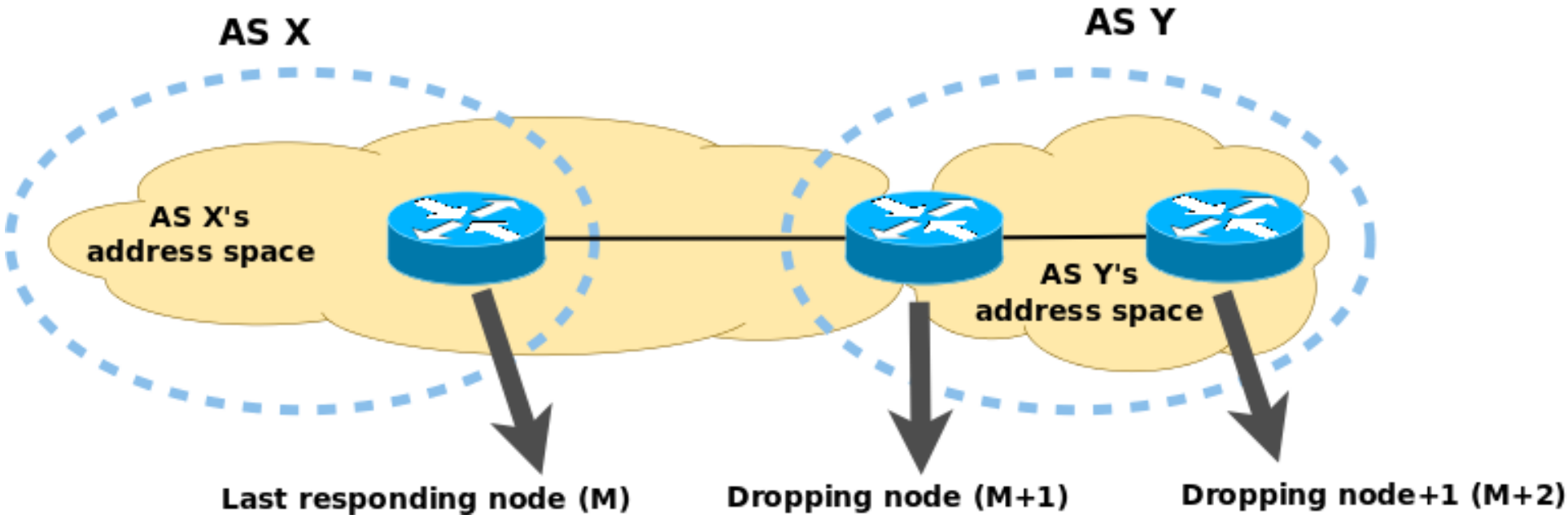
blackhole6: ASes (II)

- Case 1: Address space provided by AS Y



blackhole6: ASes (III)

- Case 2: Address space provided by AS X



Finding IPv6 blackholes Demo

Statistics about IPv6 EH Support

Introduction

- A number of questions surrounded the use of IPv6 EHs:
  - Can be reliably employed on the public IPv6 Internet?
  - Anyway, are they effective for penetration-testing/attack purposes?
- There was not much real world data
- Tools we had to produce:
  - **`path6`**: EH-enabled traceroute
  - **`script6 get-alexa-domains`**: Obtain domains from Alexa's Top-1M file
  - **`script6 get-{mx,ns,aaaa}`**: Obtain different types of DNS RRs
  - **`addr6`**: Filter out uninteresting addresses (we had this one! ;-))
  - **`script6 get-trace6`**: Produce trace record for a number of targets
  - **`script6 get-trace6-stats`**: Produce stats based on the get-trace6 data

Obtaining sets of addresses

- Web server addresses (e.g., AAAA records of Alexa domains):

```
cat top-1m-domains.txt | script6 get-aaaa > top-1m-web.txt
```

- Mail server addresses:

```
cat top-1m-domains.txt | script6 get-mx | script6 get-aaaa > top-1m-mail.txt
```

- Nameserver addresses:

```
cat top-1m-domains.txt | script6 get-ns | script6 get-aaaa > top-1m-dns.txt
```

Filtering uninteresting addresses

- For measurements, we only want non-duplicate global unicast addresses
- **`addr6`** can filter out everything else:

```
cat top-1m-web.txt | addr6 -i -q -B multicast
-B unspec -k global > top-1m-web-filtered.txt
```

Measuring a list of addresses

- script6 get-trace6 ca mesure a list of IPv6 addresses

  `script6 get-trace6 DESTINATION [EHTYPE[EHSIZE]] [PROTOCOL [PORT]]`

- Target addresses are read from stdin one at a time
- For each target, we obtain:
  - Last responding node and delta-hops for non-EH probes
  - Last responding node and delta hops for EH-enabled probes
  - M+1 (dropping) node
  - M+2 node
- Output really intended to be processed by script6 get-trace6-stats

Producing statistics

- **`script6 get-trace6-stats`** produces stats based on the output of of **`script6 get-trace6`**
- Among the results:
  - Packet drop rate for EH-enabled packets
  - Drop rate at different ASes (best-case and worst-case scenarios)
  - List of dropping ASes
  - etc.

Statistics about IPv6 EH Support Demo

Some conclusions

Some conclusions

- Coding IPv6 tools:
  - Portability harder than expected (harder than it “should”)
  - Increased usage -> increased code quality
- Using IPv6 tools
  - There is a lot to learn through practice
- **Please use the toolkit and report back to us**

Questions?

Acknowledgements

- Thanks to Enno, Niki, Ayhan, and the rest of the Troopers crew!

Thanks!

Fernando Gont

fgont@si6networks.com

IPv6 Hackers mailing-list

http://www.si6networks.com/community/

www.si6networks.com