Advanced IPv6 Network Reconnaissance

Fernando Gont

IPv6 Security Summit @ Troopers16
Heidelberg, Germany. March 14-15, 2016

About...

- Security Researcher and Consultant at SI6 Networks
- Published:
  - 25 IETF RFCs (13 on IPv6)
  - 10+ active IETF Internet-Drafts
- Author of the SI6 Networks' IPv6 toolkit
  - http://www.si6networks.com/tools/ipv6toolkit
- I have worked on security assessment of communication protocols for:
  - UK NISCC (National Infrastructure Security Co-ordination Centre)
  - UK CPNI (Centre for the Protection of National Infrastructure)
- More information at: http://www.gont.com.ar

Introduction

- IPv6 changes the “Network Reconnaissance” game
- Brute force address scanning attacks undesirable (if at all possible)
- Security guys need to evolve in how they do net reconnaissance
  - Pentests/audits
  - Deliberate attacks
- Network reconnaissance support in security tools has traditionally been **very poor**

New IETF RFC!

IETF RFC 7707 on
"Network Reconnaissance in IPv6 Networks"!

IPv6 Address Scanning
Dismantling a Myth

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

Is the search space for a /64 really 2^{64} addresses?

IPv6 addresses in the real world

- Malone originally measured (*) the address generation policy of hosts and routers in real networks

Address type	Percentage
SLAAC	50%
IPv4-based	20%
Teredo	10%
Low-byte	8%
Privacy	6%
Wordy	<1%
Others	<1%

Hosts

Address type	Percentage
Low-byte	70%
IPv4-based	5%
SLAAC	1%
Wordy	<1%
Privacy	<1%
Teredo	<1%
Others	<1%

Routers

Malone, D., "Observations of IPv6 Addresses", Passive and Active Measurement Conference (PAM 2008, LNCS 4979), April 2008, <http://www.maths.tcd.ie/~dwmalone/p/addr-pam08.pdf>.

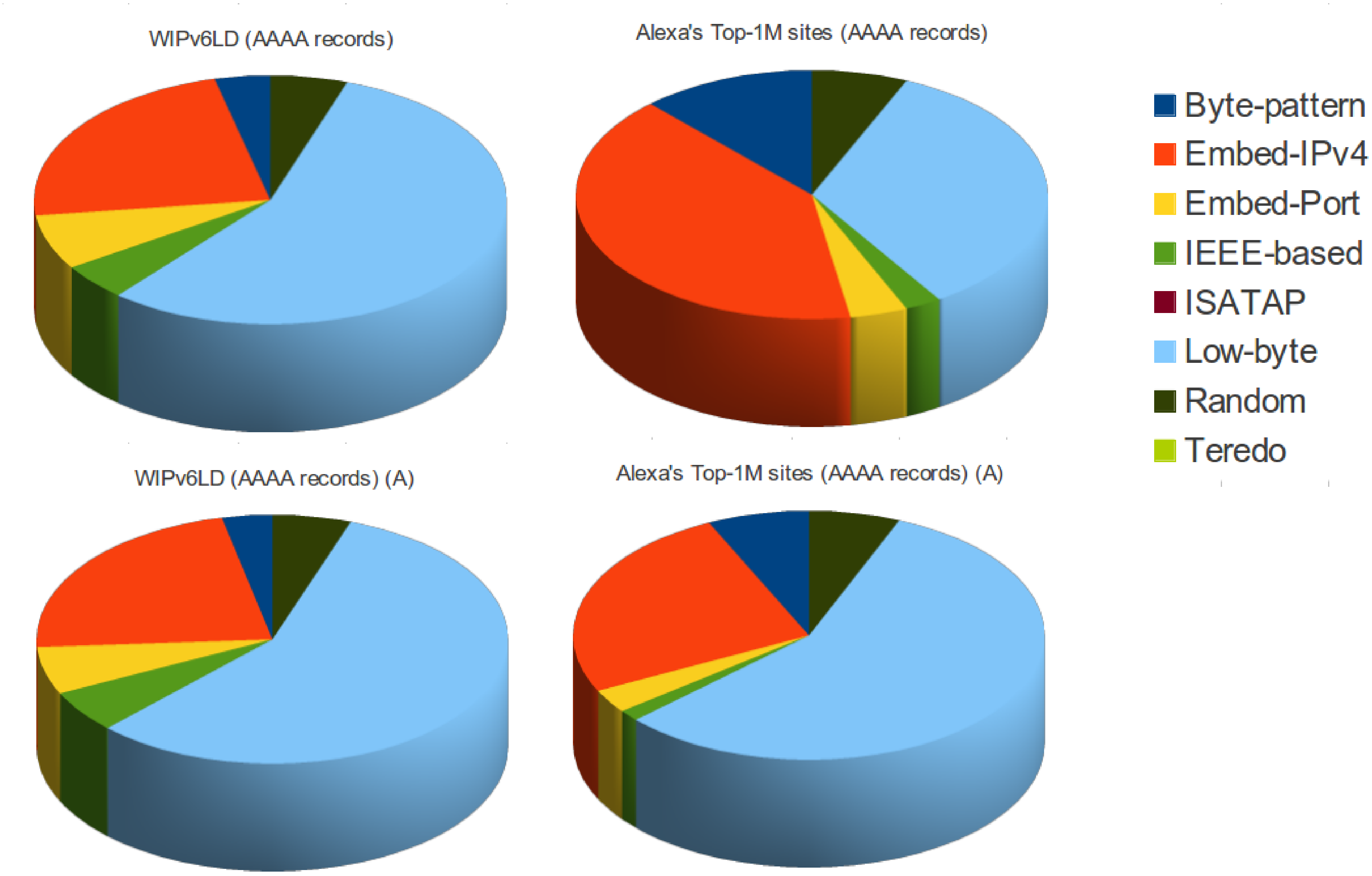
Some take aways from Malone's work

- **IPv6 addresses do follow patterns!**
- Some limitations of Malone's work:
  - Possibly dated results
    - Widespread use of transition technologies for clients
    - Widespread use of manual configuration for clients
  - It does not contain data for servers
- This motivated our study on the topic

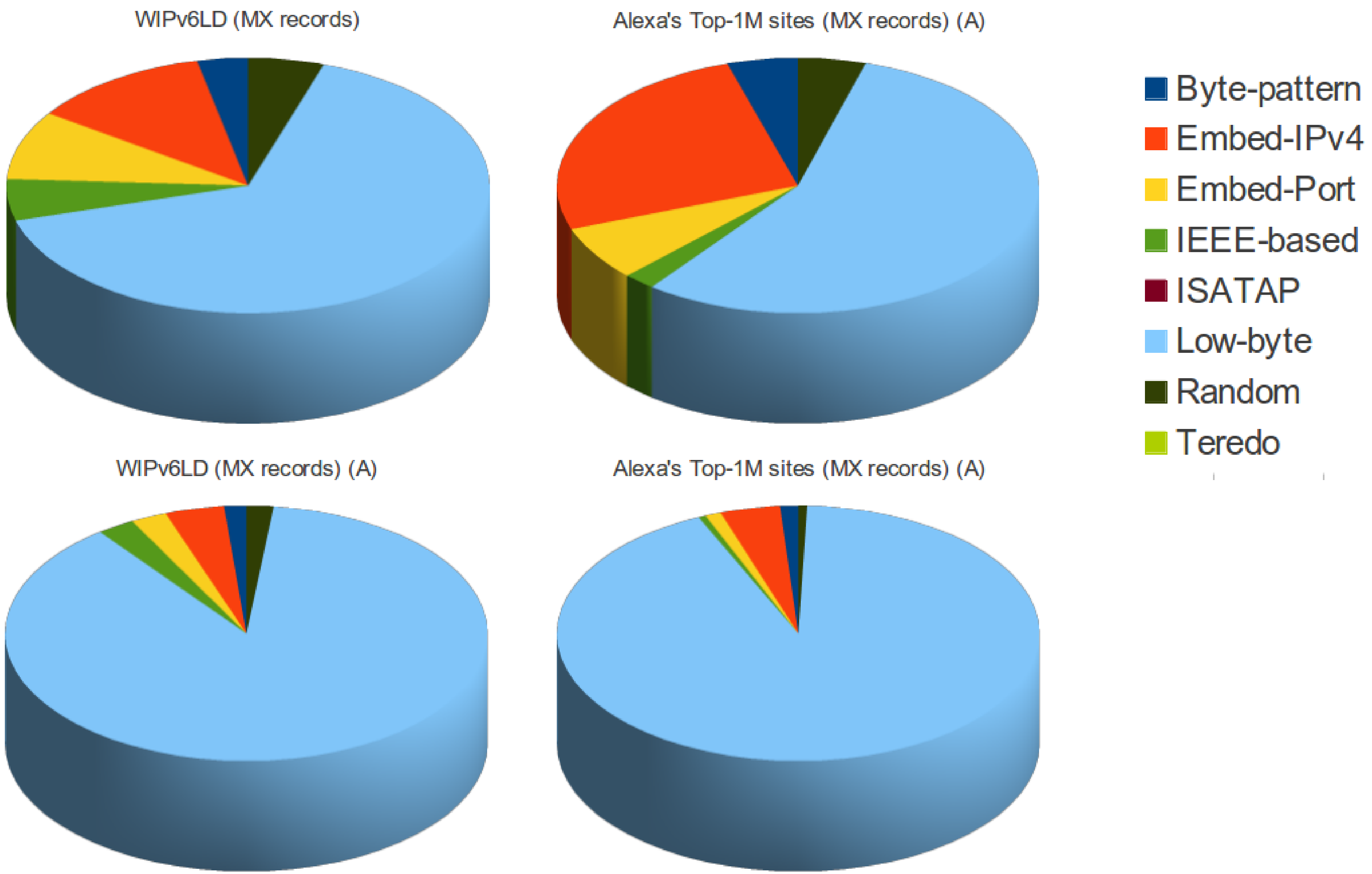
Our experiment

- Find “a considerable number of IPv6 nodes” for address analysis:
  - Alexa Top-1M sites -> **`script6`** -> **`addr6`**
  - World IPv6 Launch Day site -> **`script6`** -> **`addr6`**
- For each domain:
  - AAAA records
  - NS records -> AAAA records
  - MX records -> AAAA records
- What did we find?

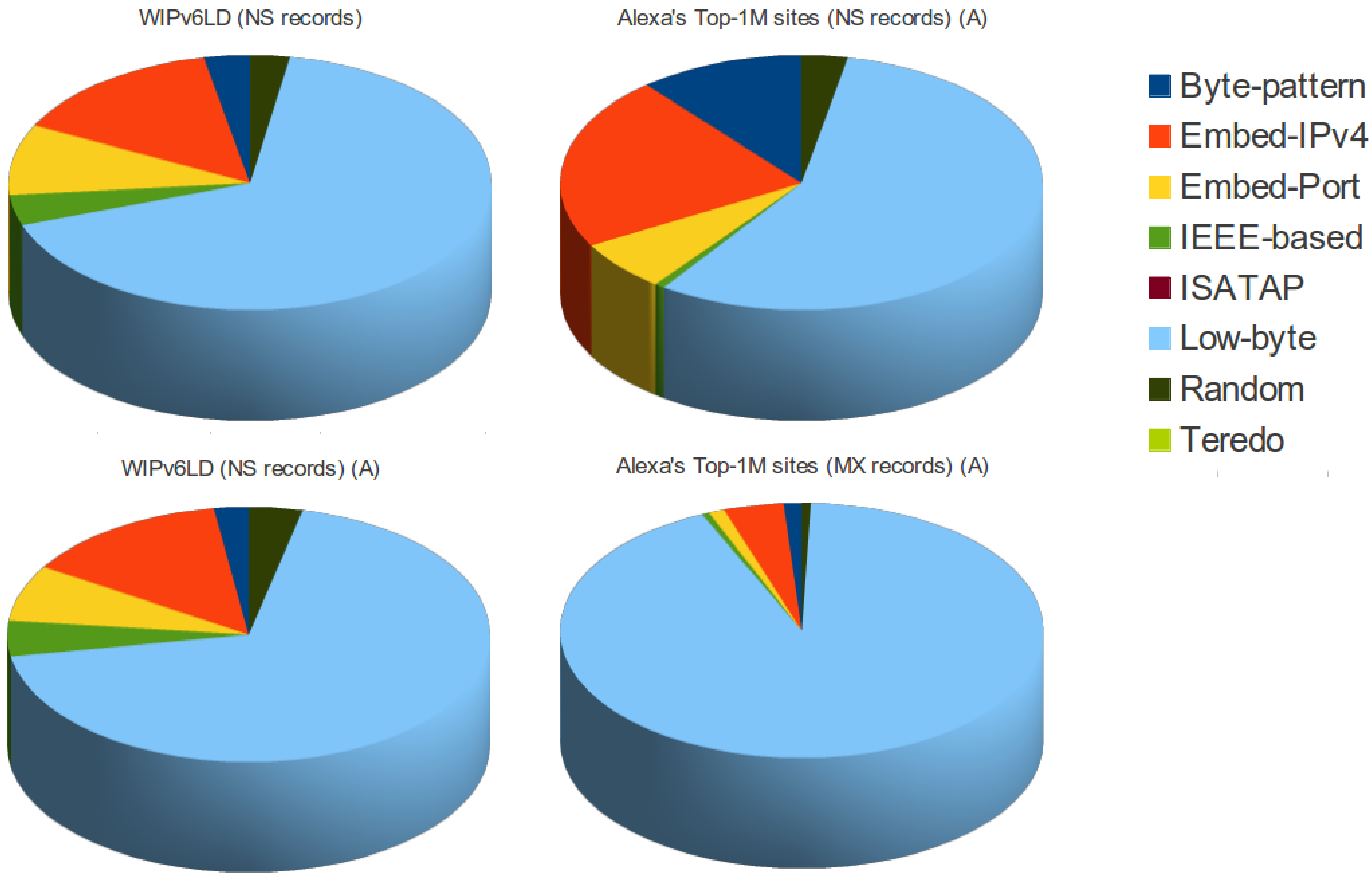
IPv6 address distribution for the web



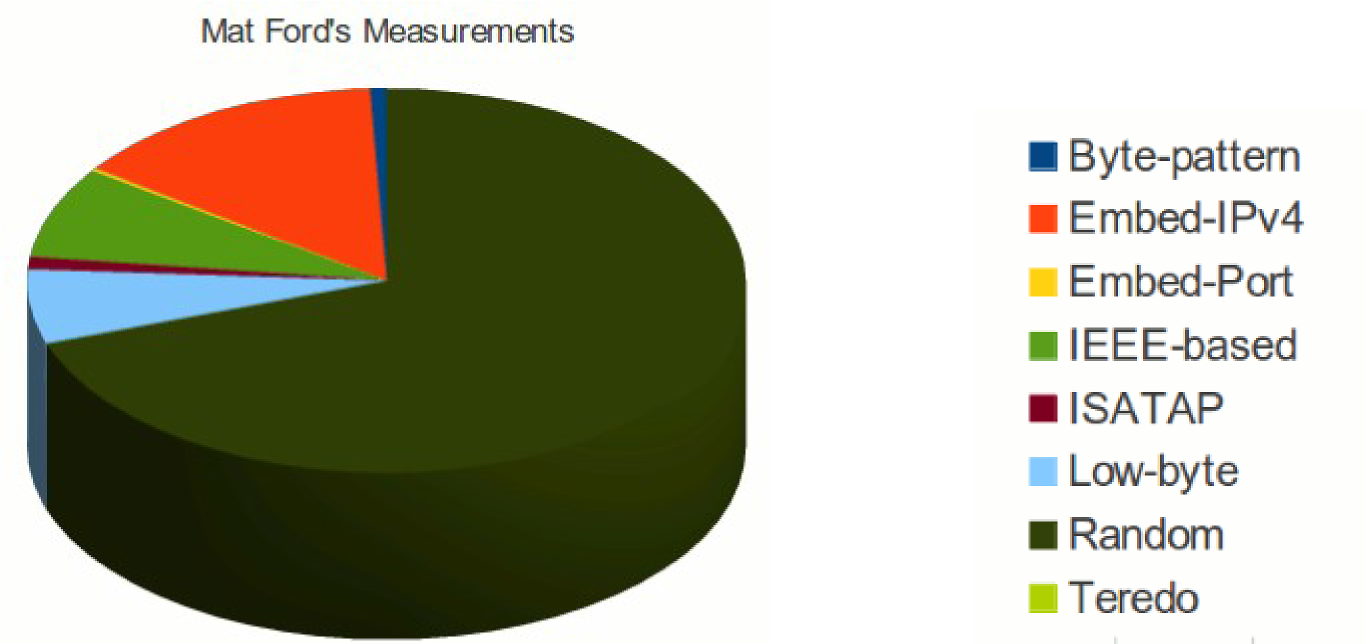
IPv6 address distribution for mail servers



IPv6 address distribution for the DNS



Client addresses



- Caveats:
  - Graphic illustrates IID types used for outgoing connections.
  - No data about IID types used for stable addresses when RFC4941 is employed.

Source: <http://www.internetsociety.org/blog/2013/05/ipv6-address-analysis-privacy-transition-out>

Some take-aways from our study

- Server addresses do follow patterns
  - The majority of addresses follow patterns with a small search space
- Passive measurements on client addresses are of little use
  - Due to IPv6 temporary addresses (RFC4941)

IPv6 Addressing Scanning
Leveraging Address Patterns

IPv6 addresses embedding IEEE IDs

24 bits	16 bits	24 bits
IEEE OUI	**FF FE**	Lower 24 bits of MAC
Known or guessable	Known	Unknown

- In practice, the search space is at most ~2^{24} bits – **feasible!**
- The low-order 24-bits are not necessarily random:
  - An organization buys a large number of boxes
  - In that case, MAC addresses are usually consecutive
  - Consecutive MAC addresses are generally in use in geographically-close locations

IPv6 addresses embedding IEEE IDs (II)

- Virtualization technologies present an interesting case
- Virtual Box employs OUI 08:00:27 (search space: ~2^{24})
- VMWare ESX employs:
  - Automatic MACs: OUI 00:05:59, and next 16 bits copied from the low order 16 bits of the host's IPv4 address (search space: ~2^{8})
  - Manually-configured MACs:OUI 00:50:56 and the rest in the range 0x000000-0x3fffff (search space: ~2^{22})
- Examples:

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

# scan6 -d fc00::/64 -V vbox

# scan6 -d fc00::/64 -V vmware -Q 10.10.0.0/16
```

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!
- Examples:

```
# scan6 -d fc00::/64 -B all -Q 10.10.0.0/16
```

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 -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 -d fc00::/64 --tgt-low-byte
```

scan6 coolness

- “What if I'm lazy enough to 'set' an appropriate address pattern?”
  - scan6 infers the address pattern for you!
- Examples:

```
sudo scan6 -d DOMAIN/64 -v

sudo scan6 -d ADDRESS/64 -v
```

IPv6 Addressing Scanning

The low-hanging fruit

Overview

- Leverage IPv6 all-nodes link-local multicast address
- Employ multiple probe types:
  - Normal multicasted ICMPv6 echo requests (don't work for Windows)
  - Unrecognized options of type 10xxxxxx
- Combine learned IIDs with known prefixes to learn all addresses
- Example:

```
# scan6 -i eth0 -L
```

Working with IPv6 addresses

`addr6` to the rescue!

Introduction

- Given a set of IPv6 address, you may want to:
  - Discard duplicate addresses
  - Discard addresses of specific scope
  - Analyze the address type
  - Produce statistics
- We created `addr6` for that!

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

- addr6 has a number of features to filter IPv6 addresses
- Filter duplicate addresses:

  cat LIST.TXT | addr6 -i -q

- Accept (or block) specific prefixes:

  cat LIST.TXT | addr6 -i --accept 2001:db8::/16

  cat LIST.TXT | addr6 -i --block 2001:db8::/16

- Accept (or block) address types:

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

  cat LIST.TXT | addr6 -i --block-type TYPE

  - Types: unicast, unspec, multicast

Filtering IPv6 addresses (II)

- Accept (or block) address scopes:

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

  - Scopes: interface, link, admin, site, local, global...
- Accept (or block) unicast address types:

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

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

Producing statistics

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

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

Canonic IPv6 addresses

- Which of these addresses are equivalent?
  1) fc00:1:0:0:0:0:0a0a:0a0a
  2) fc00:1::a0a:a0a
  3) fc00:1:0000:0000:0000:0000:0a0a:0a0a
  4) fc00:1::10.10.10.10
  5) fc00:1::aa:aa
  6) fc00:1::0a0a:0a0a
  7) fc00:1:0::a0a:a0a
  8) fc00:1:0000::a0a:a0a
- Moral of the story?

Canonic IPv6 addresses (II)

- Text-based comparisons must be made between canonic IPv6 addresses
- addr6 can print the canonic version of an IPv6 address:

```
addr6 -a fc00::10.10.10.10 -c
```

IPv6 Extension Headers

In Network Reconnaissance

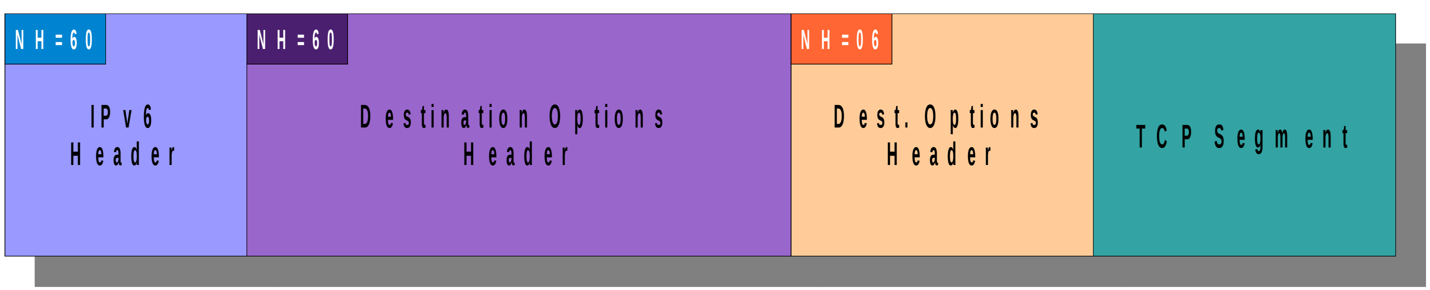
SI6 NETWORKS

IPv6 Extension Headers

Overview

General IPv6 packet format

- Consists of an IPv6 header chain and an (optional) payload
- Each Extension Header is typically encoded as TLV (Type-Length-Value)
- Any number of instances of any number of different headers are allowed
- Each header may contain an arbitrary number of options



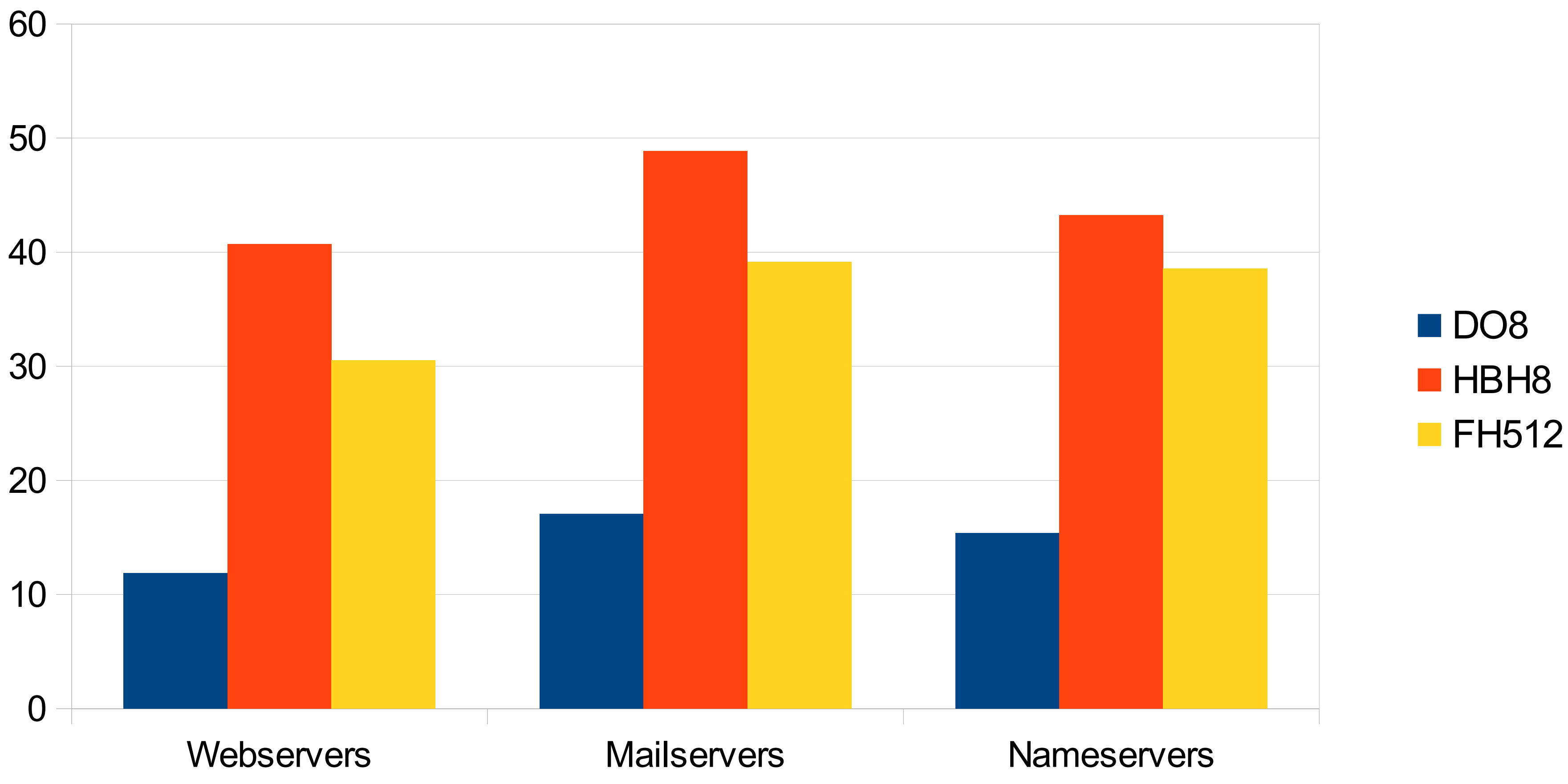
Processing the IPv6 header chain

- Implications for inspecting “boxes”:
  - Large number of headers/options may have a negative impact on performance
  - Many routers can only look into a few dozen bytes into the packet
  - It becomes harder (if at all possible) to enforce layer-4 ACLs
  - Fragmentation represents similar challenge as in IPv4
- Potential benefits for network reconnaissance:
  - Evasion

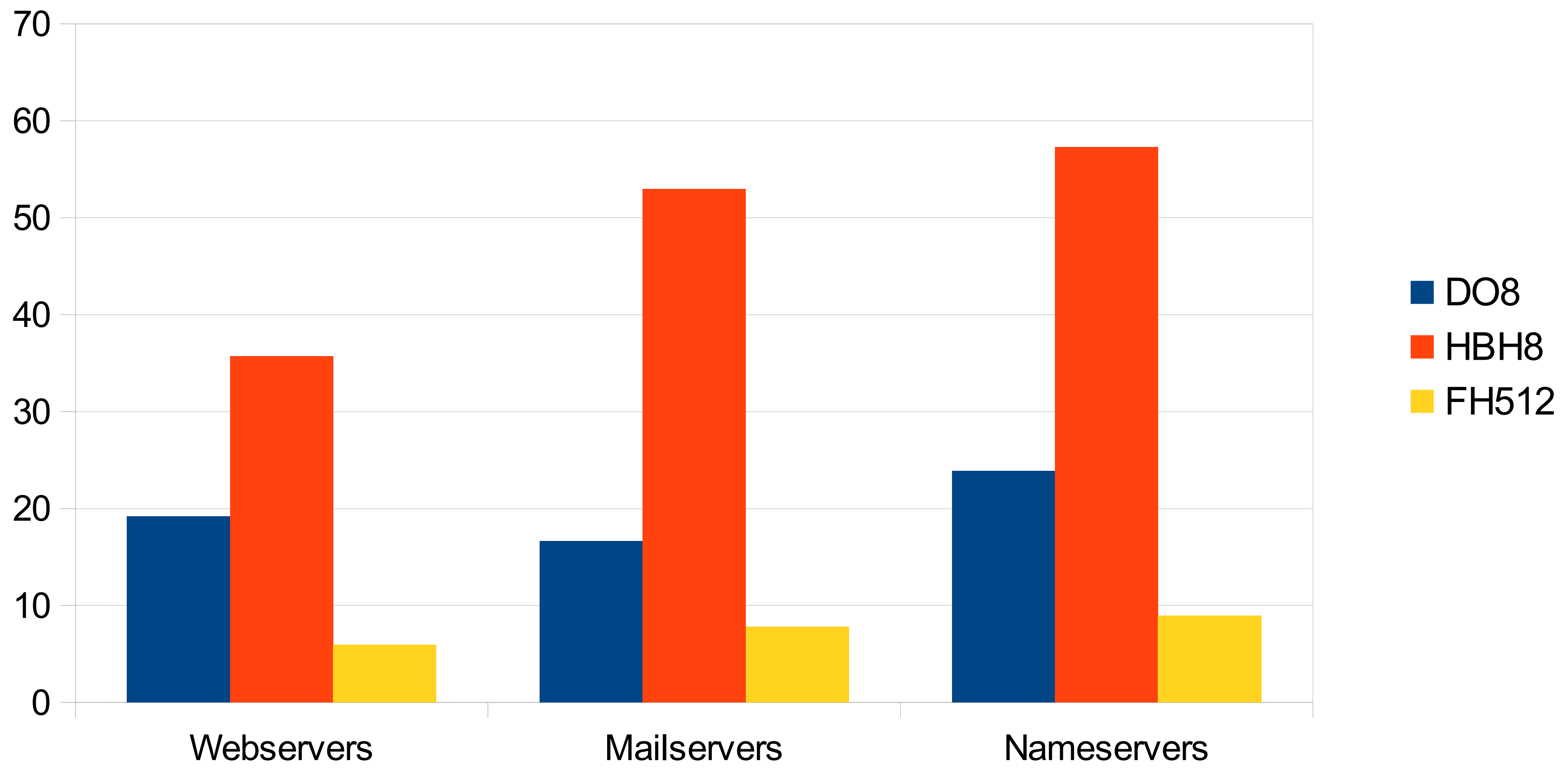
IPv6 Extension Headers

In The Real World

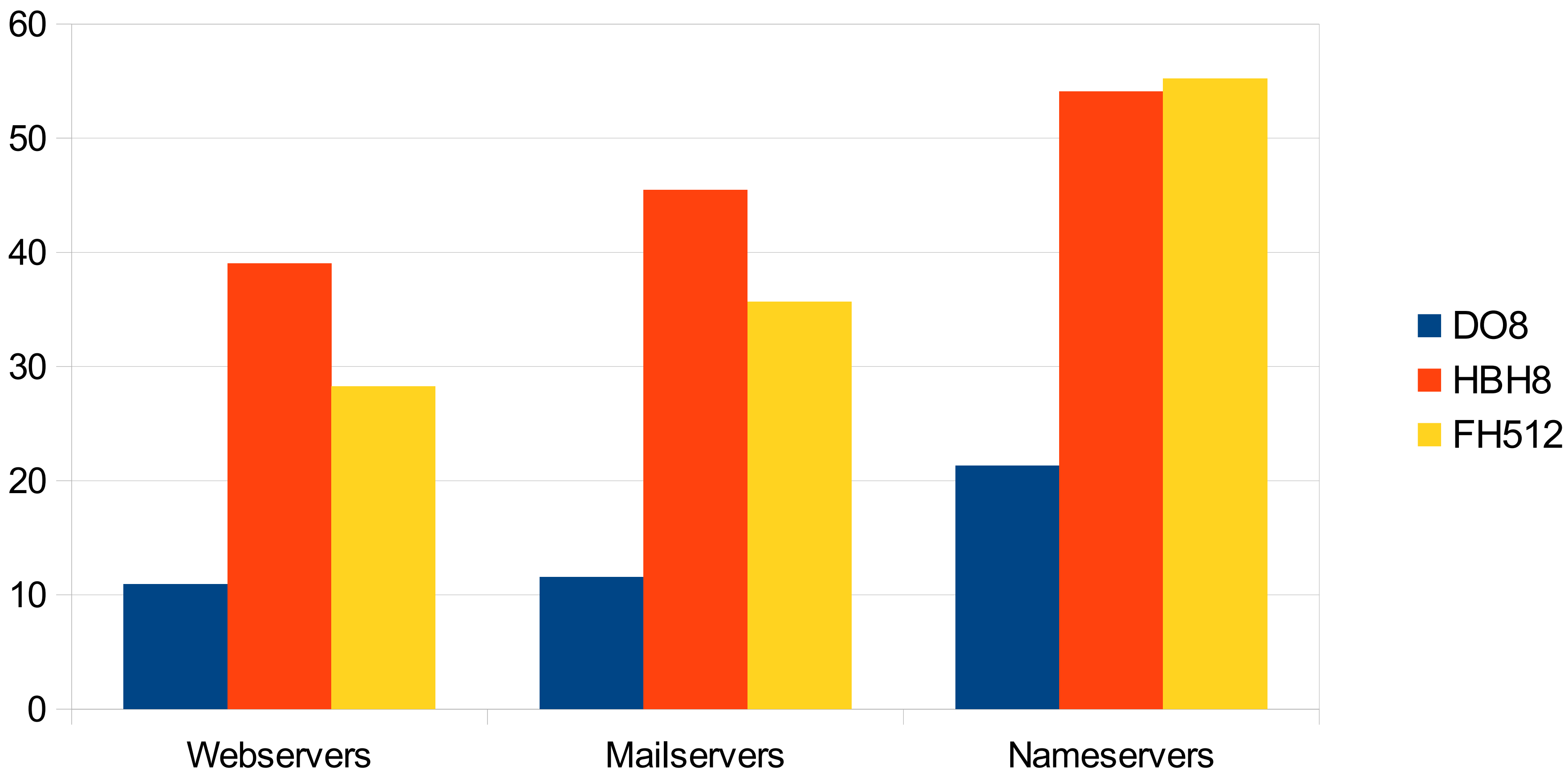
WIPv6LD dataset: Packet Drop rate



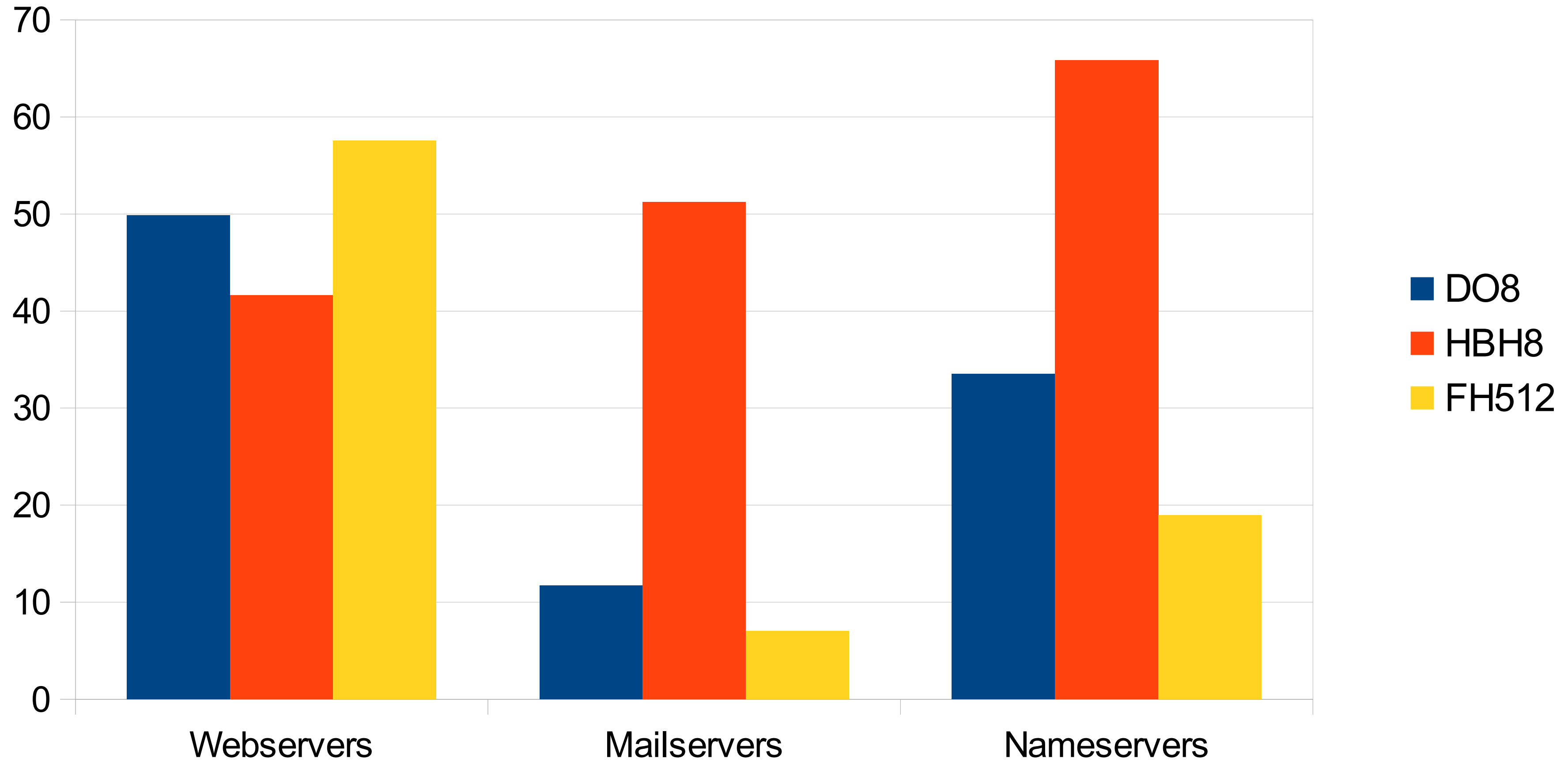
WIPv6LD dataset: Drops by diff. AS



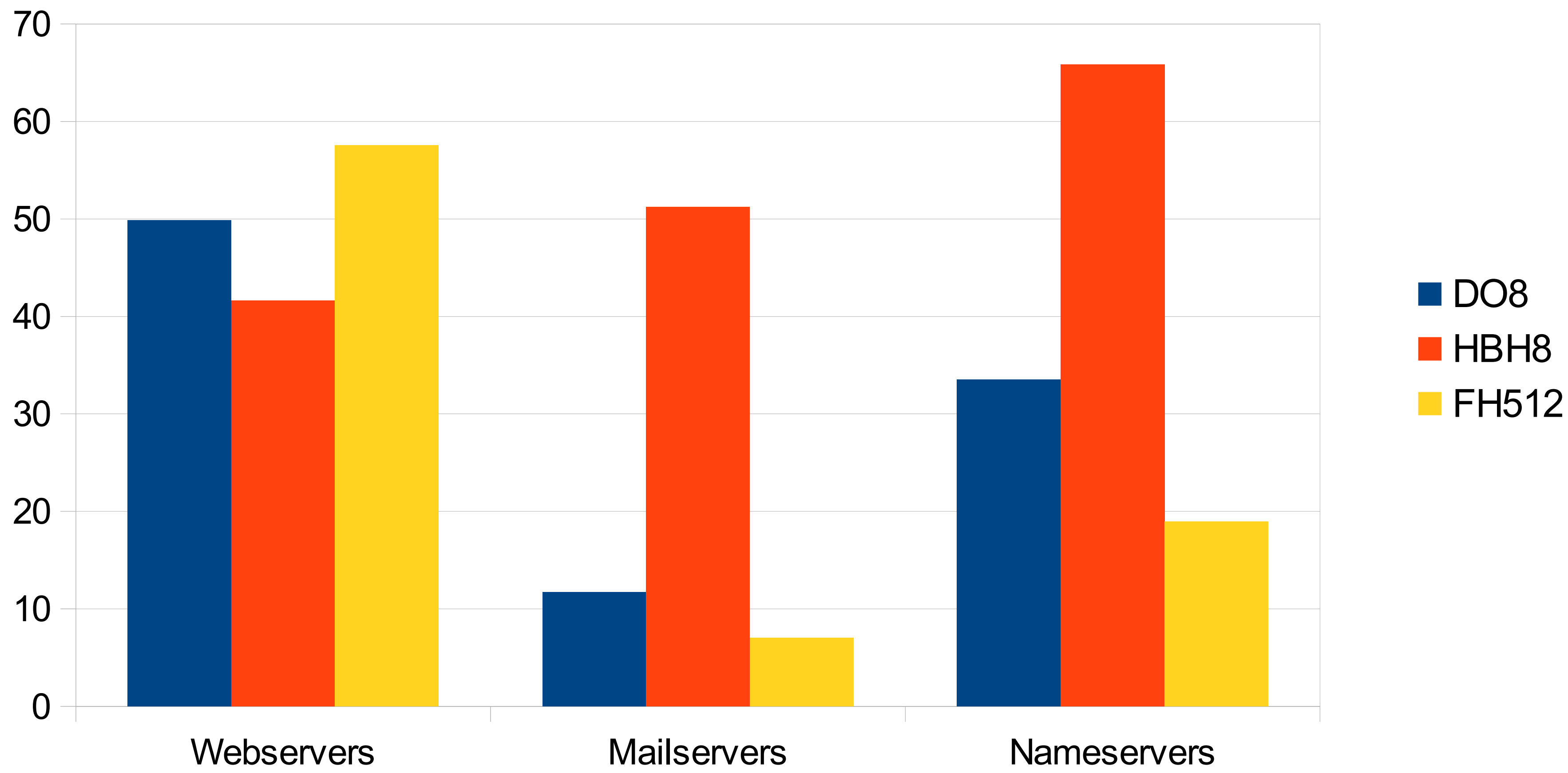
Alexa dataset: Packet Drop rate



Alexa dataset: Drops by diff. AS



Alexa dataset: Drops by diff. AS



So... what does this all mean?

- IPv6 EHs “not that cool” for evasion or reconnaissance

  ...at least when doing remote IPv6 network reconnaissance!

IPv6 Extension Headers

Use in network reconnaissance

path6: An EH-enabled traceroute

- How far do your IPv6 EH-enabled packets get?
- No existing traceroute tool supported IPv6 extension headers
- 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

path6: An EH-enabled traceroute (II)

- Example of traceroute with 8-byte DOH:

```
# path6 -d DEST -u 8 -p icmp
```

- Example of traceroute with fragmentation:

```
# path6 -d DEST -p icmp -P 500 -y 256
```

- Example of traceroute with TCP payload:

```
# path6 -d DEST -p tcp -a 80
```

blackhole6: Finding IPv6 blackholes

- How it works?
  - path6 without EHs + path6 with EHs + a little bit of magic

```
fgont@satellite:~$ sudo blackhole6 www.google.com do8
SI6 Networks IPv6 Toolkit v2.0
blackhole6: A tool to find IPv6 blackholes
Tracing www.google.com (2607:f8b0:400b:807::1012)...

Dst. IPv6 address: 2607:f8b0:400b:807::1012 (AS15169 - GOOGLE - Google
Inc.,US)
Last node (no EHs): 2607:f8b0:400b:807::1012 (AS15169 - GOOGLE - Google
Inc.,US) (13 hop(s))
Last node (DO 8): 2001:5a0:12:100::72 (AS6453 - AS6453 - TATA
COMMUNICATIONS (AMERICA) INC,US) (7 hop(s))
Dropping node: 2001:4860:1:1:0:1935:0:75 (AS15169 - GOOGLE - Google
Inc.,US || AS15169 - GOOGLE - Google Inc.,US)
```

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 (L)
4) Find “L” in “ROUTE” -> dropping system (X) is next in ROUTE
5) Compare AS(X) with AS(D), 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
7. fc00:b:1:1000::1 (DROP)	
8. fc00:b:2:5000::1	
9. fc00:b:4:5000::1	
10. fc00:d::1	

SI6 NETWORKS

blackhole6: Methodology (III)

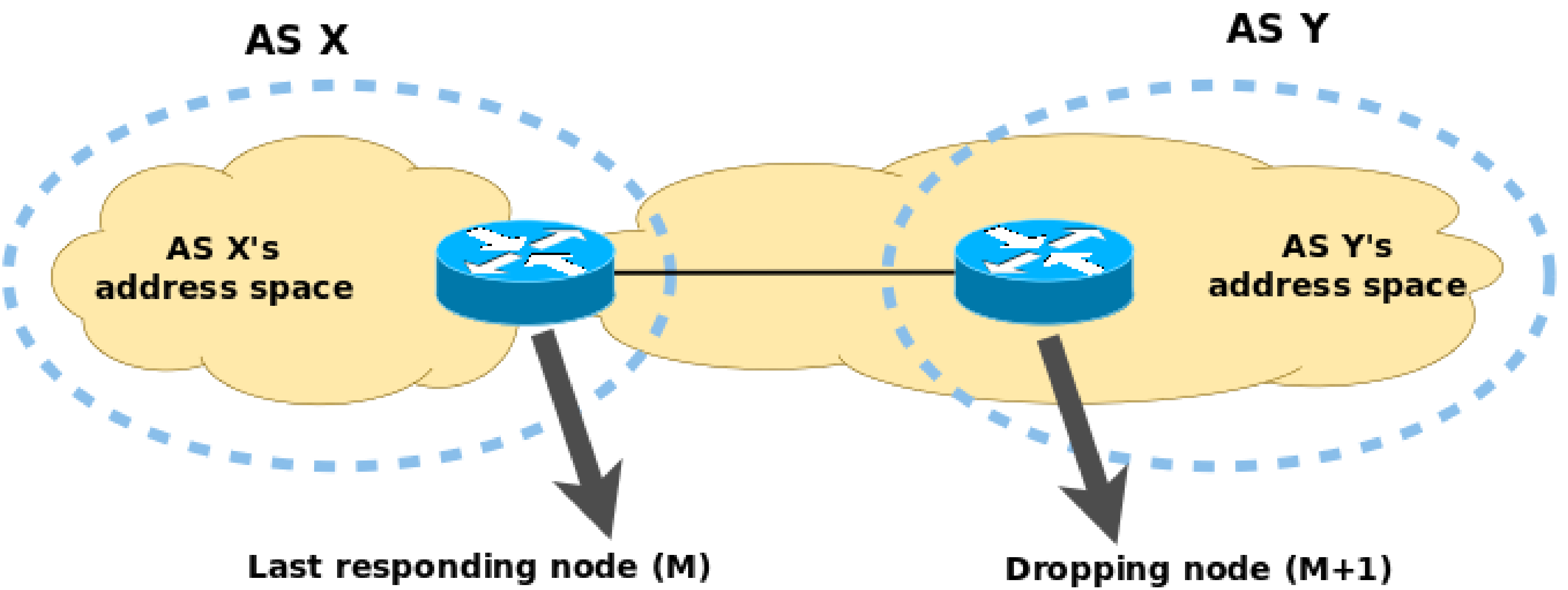
- We assume ingress filtering...
- Otherwise dropping node actually is M rather than M+1

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?

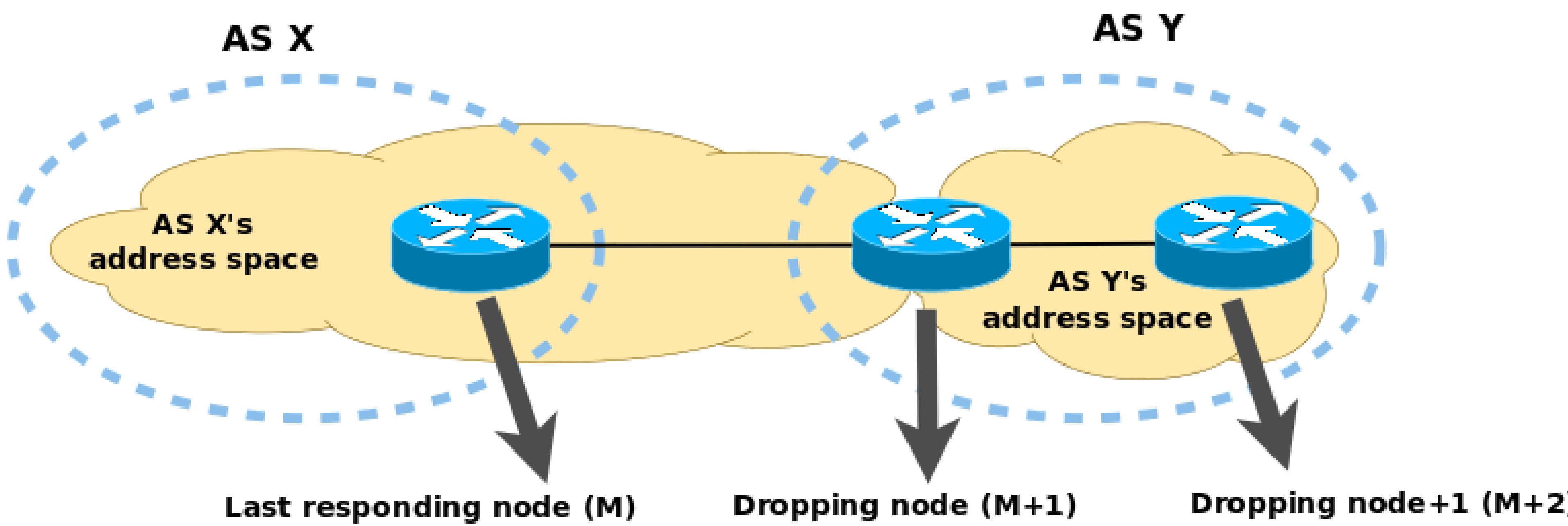
blackhole6: ASes (II)

- Case 1: Address space provided by AS Y



blackhole6: ASes (III)

- Case 2: Address space provided by AS X



Port scanning
The basics

IPv6-based 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:

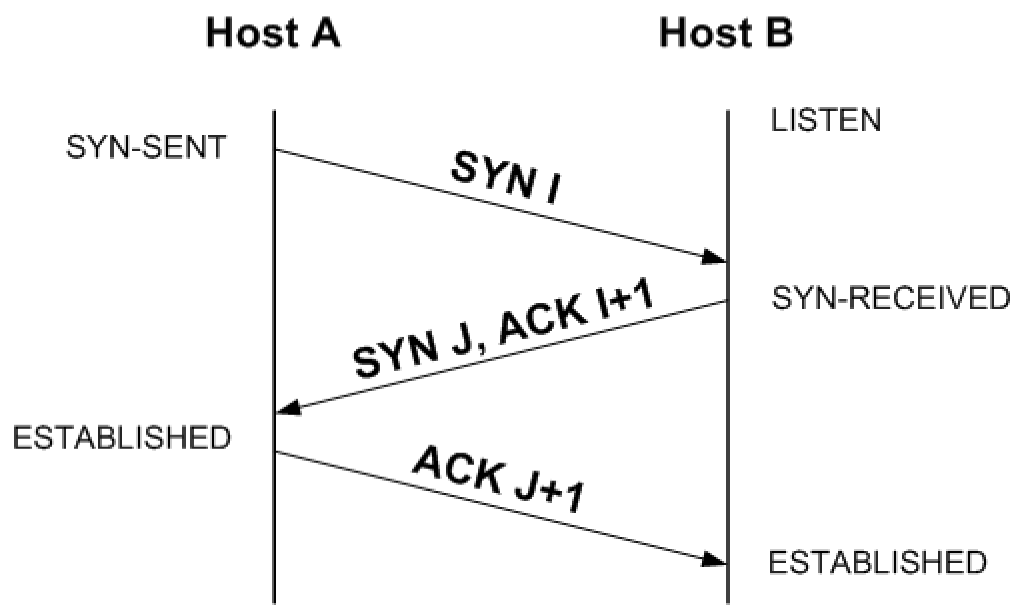
```
--tcp-scan-type {syn,fin,null,xmas,ack}
```

- 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**

TCP/UDP most popular ports

- scan6 can target the most frequently open ports
- All top ports for all protocols:

`--port-scan all:top:all`

- Top N of all protocols:

`--port-scan all:top:N`

- All TCP top ports:

`--port-scan tcp:top:all`

- Top N TCP ports

`--port-scan tcp:top:N`

Port Scanning

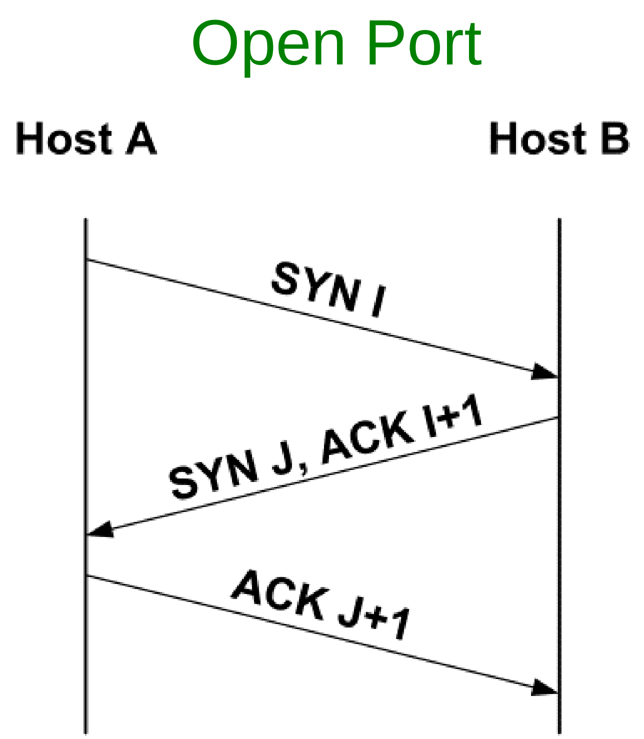
EH-based IPv6 Idle Scan

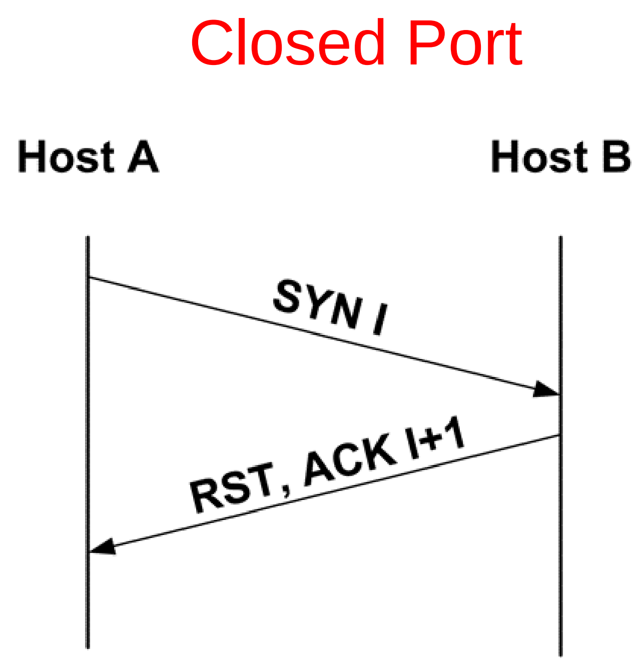
Idle scan: Introduction

- Stealth port scanning technique
- No need to contact the target with our Source Address
- Prevents easy tracing of the attacker
- The attacker only needs a host that employs predictable Identification values.

Idle scan: TCP 3WHS review

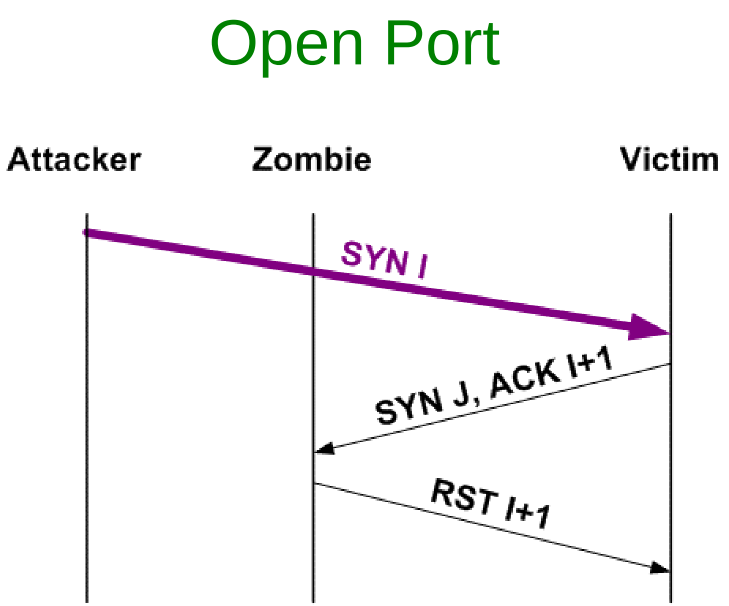
- Normal TCP 3WHS

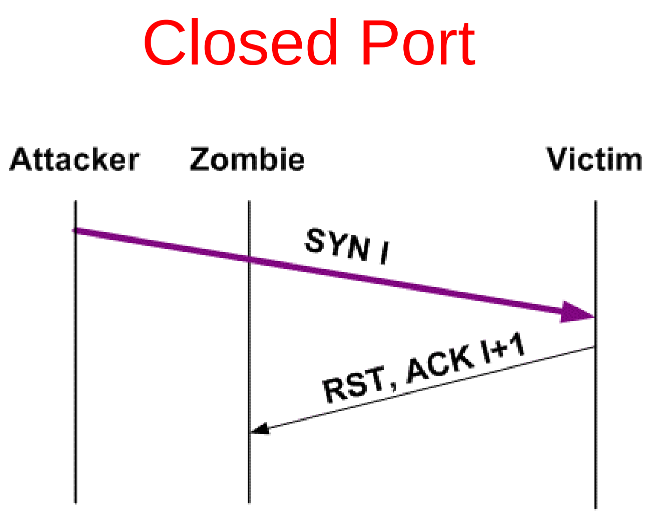




Idle scan: TCP 3WHS review

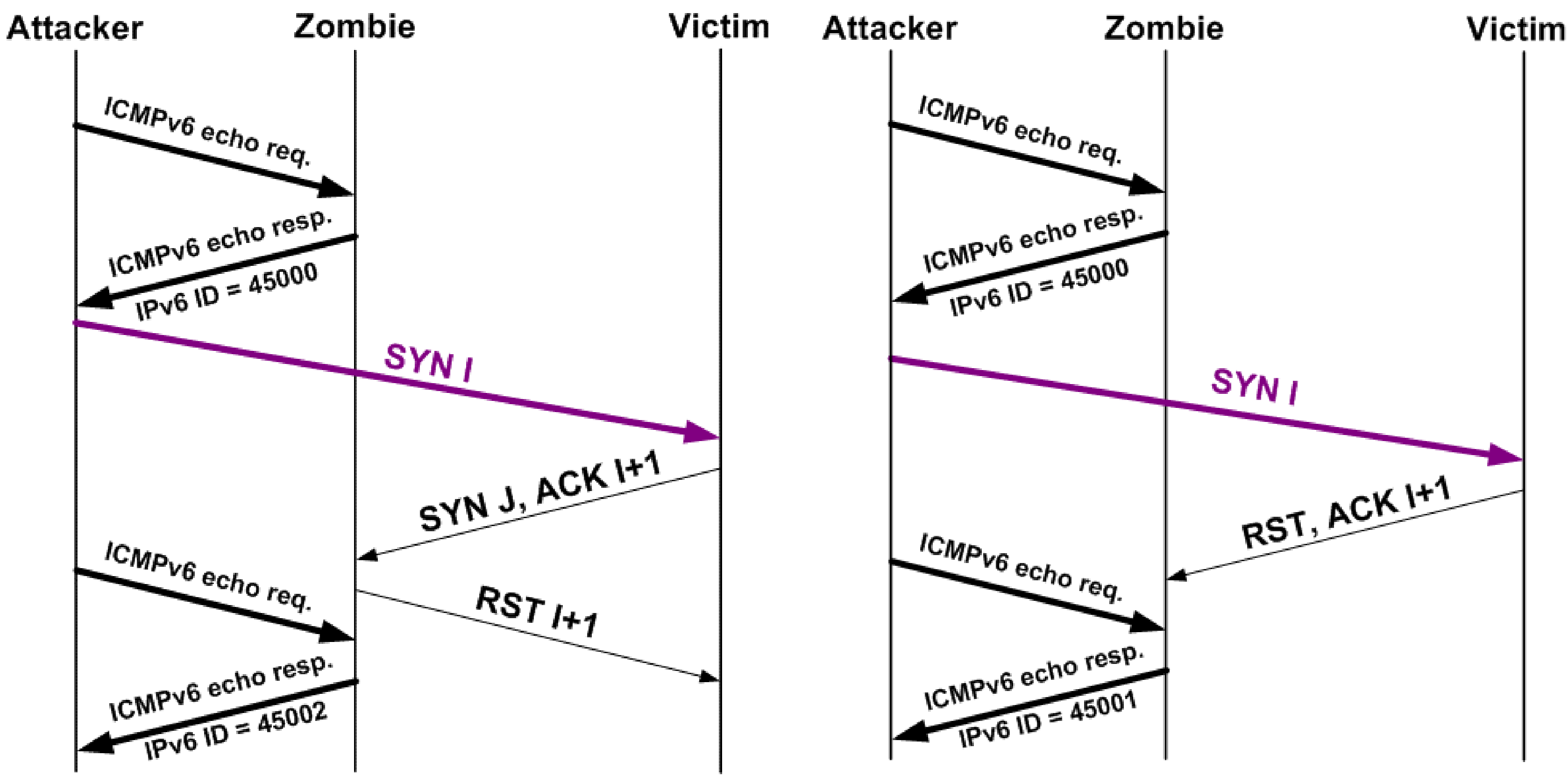
- TCP 3WHS with spoofed segments





Idle scan “implementation”

Open Port　　　　Closed Port

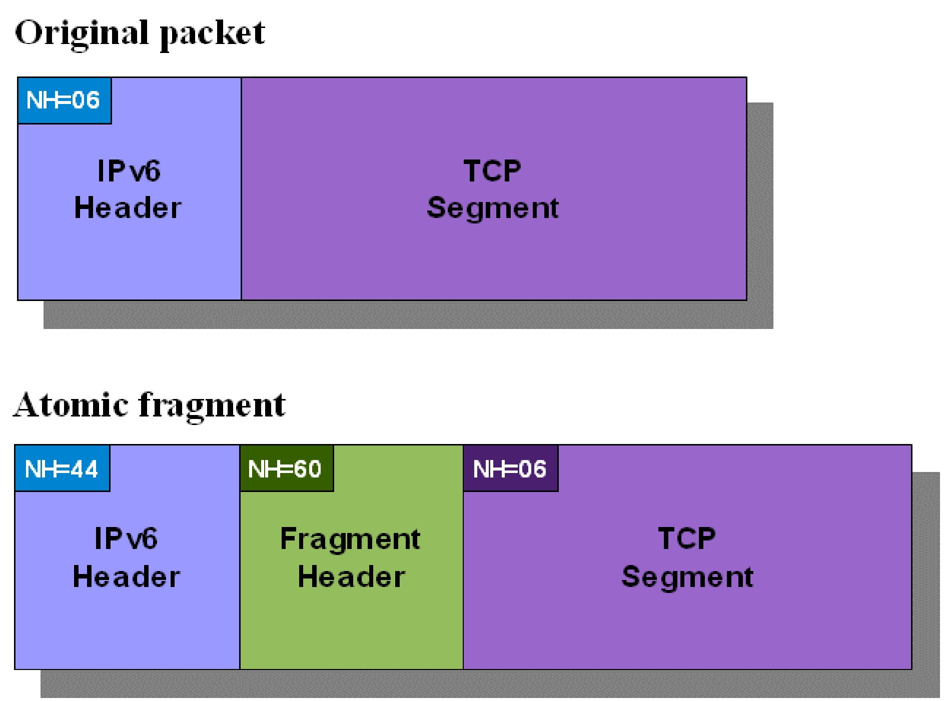


Idle scan: Challenge in IPv6

- Base IPv6 header does not contain a Frag ID
- Only way to exploit the Frag ID is when a FH is present
- But...How do we trigger/elicit fragmentation?

IPv6 “atomic” fragments

- ICMPv6 PTB < 1280 triggers inclusion of a FH in all packets to that destination (not actual fragmentation)
- Result: IPv6 atomic fragments (Frag. Offset=0, More Frag.=0)

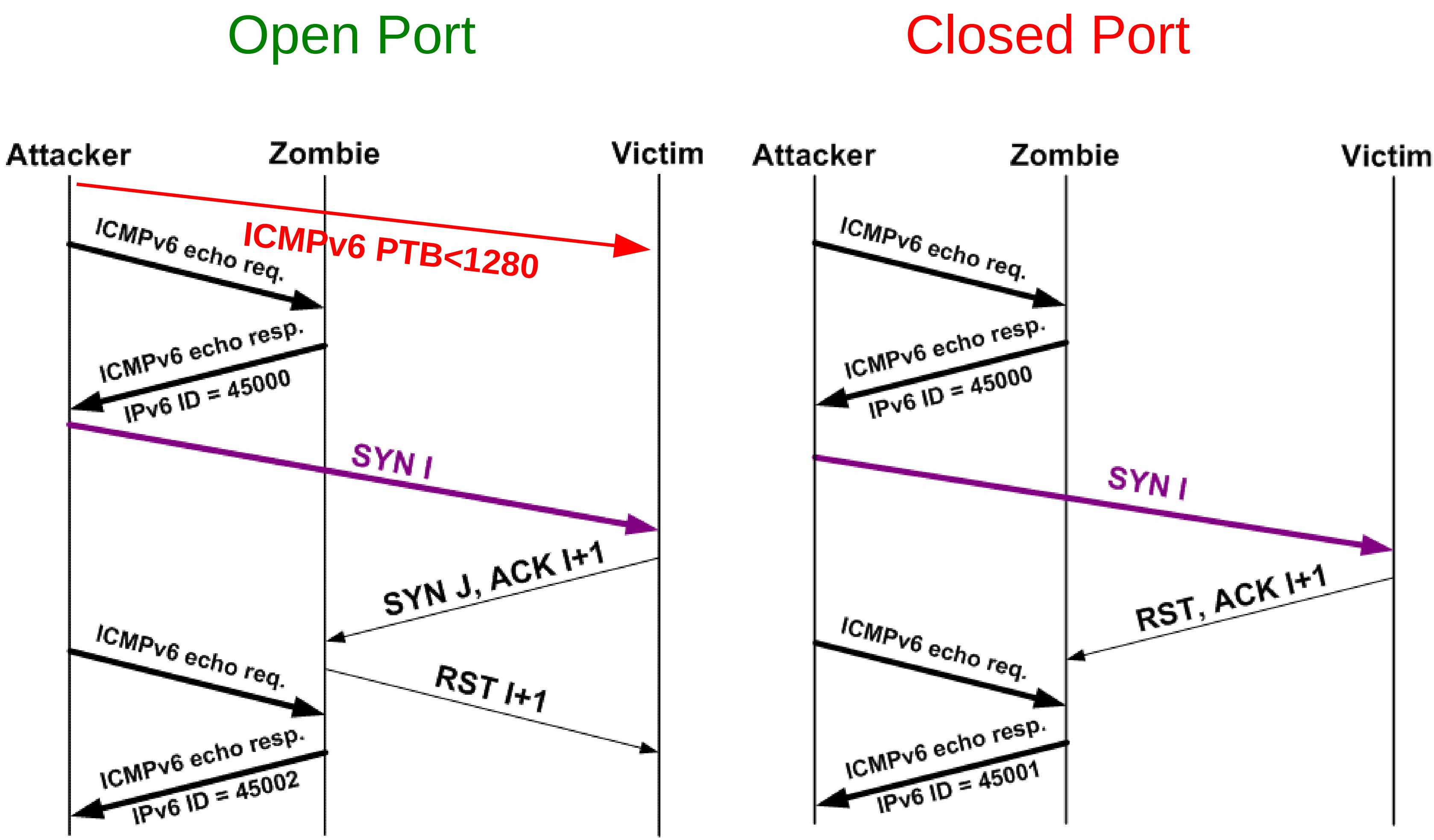


Handling of IPv6 atomic fragments

Operating System	Atomic Frag. Support	Improved processing
FreeBSD 8.0	No	No
FreeBSD 8.2	Yes	No
FreeBSD 9.0	Yes	No
Linux 3.0.0-15	Yes	Yes
NetBSD 5.1	No	No
OpenBSD-current	Yes	Yes
Solaris 11	Yes	Yes
Windows Vista (build 6000)	Yes	No
Windows 7 Home Premium	Yes	No

At least OpenBSD patched in response to our IETF I-D – more patches expected!

Idle scan full implementation



Idle scan: nmap implementation

- IPv6 idle scan available in nmap version > vx.x
- Implementation by Mathias Morbitzer
- Example:

Idle scan: My take :-)

- Idle scan is a cool idea
- The IPv6 version is even more “creative”
- However,
  - Use of EHs makes probes unreliable
  - Generation of IPv6 atomic fragments is being deprecated. See:
    - draft-ietf-6man-deprecate.atomfrag-generation
    - draft-ietf-6man-rfc2460bis

ICMPv6 Informational Messages

ICMPv6 Informational Messages

- Echo Request/Echo response:
  - Used to test node reachability (“ping6”)
  - Widely supported, although disabled by default in some OSes
- Node Information Query/Response
  - Specified by RFC 4620 as “Experimental”, but supported (and enabled by default) in KAME.
  - Not supported in other stacks
  - Used to obtain node names or addresses.

ICMPv6 Informational Messages

Some not-so-widely-known gems

Node Information Query/Response

- Specified in RFC 4620 as “Experimental”, but included (and enabled by default) in KAME
- Allows nodes to request certain network information about a node in a server-less environment
  - Queries are sent with a target name or address (IPv4 or IPv6)
  - Queried information may include: node name, IPv4 addresses, or IPv6 addresses
- Node Information Queries can be sent with the ping6 command (“-w” and “-b” options)

Node Information Query/Response

- Response to Node Information Queries is controlled by the sysctl net.inet6.icmp6.nodeinfo:
  - 0: Do not respond to Node Information queries
  - 1: Respond to FQDN queries (e.g., “ping6 –w”)
  - 2: Respond to node addresses queries (e.g., “ping6 –a”)
  - 3: Respond to all queries
- net.inet6.icmp6.nodeinfo defaults to 1 in OpenBSD, and to 3 in FreeBSD.
- My take: unless you really need your nodes to support Node Information messages, disable it (i.e., “sysctl –w net.inet6.icmp6.nodeinfo=0).

NI Query/Response: Examples

- Query node names

```
$ ping6 -w ff02::1%vic0

PING6(72=40+8+24 bytes) fe80::20c:29ff:feaf:194e%vic0 --> ff02::1%vic0
41 bytes from fe80::20c:29ff:feaf:194e%vic0: openbsd46.my.domain.
30 bytes from fe80::20c:29ff:fe49:ebdd%vic0: freebsd
41 bytes from fe80::20c:29ff:feaf:194e%vic0: openbsd46.my.domain.
30 bytes from fe80::20c:29ff:fe49:ebdd%vic0: freebsd
41 bytes from fe80::20c:29ff:feaf:194e%vic0: openbsd46.my.domain.
30 bytes from fe80::20c:29ff:fe49:ebdd%vic0: freebsd
--- ff02::1%vic0 ping6 statistics ---
3 packets transmitted, 3 packets received, +3 duplicates, 0.0% packet loss
```

NI Query/Response: Examples (II)

- Use the NI multicast group

```
$ ping6 -I vic0 -a Aacgls -N freebsd

PING6(72=40+8+24 bytes) fe80::20c:29ff:feaf:194e%vic0 --> ff02::1%vic0
76 bytes from fe80::20c:29ff:fe49:ebdd%vic0:
  fe80::20c:29ff:fe49:ebdd(TTL=infty)
  ::1(TTL=infty)  fe80::1(TTL=infty)

76 bytes from fe80::20c:29ff:fe49:ebdd%vic0:
  fe80::20c:29ff:fe49:ebdd(TTL=infty)
  ::1(TTL=infty)  fe80::1(TTL=infty)

76 bytes from fe80::20c:29ff:fe49:ebdd%vic0:
  fe80::20c:29ff:fe49:ebdd(TTL=infty)
  ::1(TTL=infty)
  fe80::1(TTL=infty)

--- ff02::1%vic0 ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
```

Network Reconnaissance

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
```

DNS support for IPv6

Brief Overview and Considerations

- AAAA (Quad-A) records enable the mapping of domain names to IPv6 addresses
- The zone “ip6.arpa” is used for the reverse mapping (i.e., IPv6 addresses to domain names)
- DNS transport can be IPv4 and/or IPv6
- Troubleshooting tools such as “dig” already include support for IPv6 DNS features
- Security implications:
  - Increased size of DNS responses due to larger addresses might be exploited for DDoS attacks

DNS for Network Reconnaissance

- Most of this ground is well-known from the IPv4-world:
  - DNS zone transfers
  - DNS bruteforcing
  - etc.
- DNS reverse-mappings particularly useful for “address scanning”

Get domains and IPv6 addresses

- script6 can do batch-processing of domain names
- Get IPv6 addresses:

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

- Get nameserver addresses:
- **`$ cat domains.txt | script6 get-ns | script6 get-aaaa`**
- Get mailserver addresses:

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

Bruteforce domain names

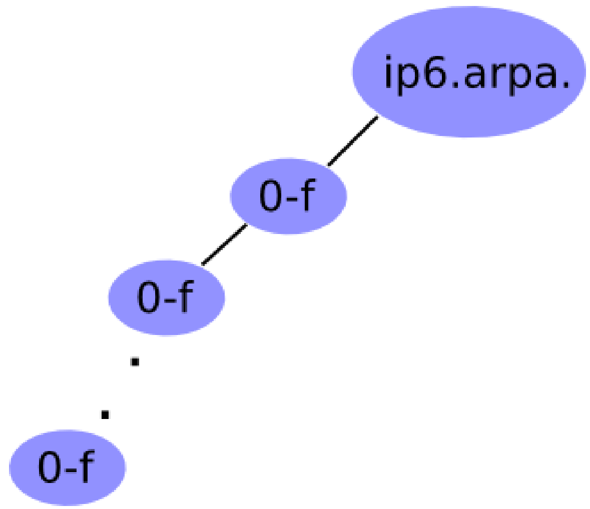
- script6 can bruteforce domain names and get the corresponding AAAA records
- For a single domain:

```
$ script6 get-bruteforce-aaaa DOMAIN
```

- Pipelined:

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

IPv6 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
```

Caveats for DNS reverse mappings

- Some DNS software responds with NOERROR for ENT (Empty Non-Terminals)
  - Please see draft-ietf-dnsop-nxdomain-cut

Aplication-based IPv6 Network Reconnaissance

Application-based Network Recon

- Many application-layer protocol deal with domain-names or IPv6 addresses.
- Some applications even leave publicly trails of data exchanges
- Examples:
  - P2P aplications
  - email
  - etc.

Application-based Network Recon (II)

- Sample email header:

```
X-ClientAddr: 46.21.160.232
Received: from srv01.bbserve.nl (srv01.bbserve.nl [46.21.160.232])
       by venus.xmundo.net (8.13.8/8.13.8) with ESMTP id p93Ar0E4003196
       for <fernando@gont.com.ar>; Mon, 3 Oct 2011 07:53:01 -0300
Received: from [2001:5c0:1000:a::943]
       by srv01.bbserve.nl with esmtpsa (TLSv1:AES256-SHA:256)
       (Exim 4.76)
       (envelope-from <fgont@si6networks.com>)
       id 1RAg8k-0000Qf-Hu; Mon, 03 Oct 2011 12:52:55 +0200
Message-ID: <4E8993FC.30600@si6networks.com>
Date: Mon, 03 Oct 2011 07:52:44 -0300
From: Fernando Gont <fgont@si6networks.com>
Organization: SI6 Networks
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.23)
Gecko/20110922 Thunderbird/3.1.15
MIME-Version: 1.0
To: Fernando Gont <fernando@gont.com.ar>
Subject: Prueba
```

Inspection of local data structures

Inspection of local data structures

- Local data structures store valuable network information:
  - IPv6 addresses of local nodes
  - IPv6 addresses of “known” nodes
  - Routing information
  - etc
- loopback6 (upcoming) aims at collecting such information from the local nod
- Example:

```
# loopback6 --all
```

Inspection of system configuration & log files

System configuration and log files

- Yet another source of possibly interesting names/addresses
- Trivial approach:
  - Walk the tree and look virtually everywhere
- Improved approach:
  - Look at interesting places depending on the local operating system
- audit6 (upcoming) aims at collecting such information from the local system
- Example:

```
# audit6 --all
```

Snooping routing protocols

System configuration and log files

- Some sites employ interior routing protocols (RIP, OSPF, etc.)
- Snooping/participating in the protocol can provide useful info
  - Internal subnets
  - Internal routers

Questions?

Thanks!

Fernando Gont

fgont@si6networks.com

IPv6 Hackers mailing-list

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

www.si6networks.com