IPv6 for the InfoSec Pro On the Go

Allan Stojanovic
University of Toronto
#include disclaimer.h
Before We Begin

- First RFCs published December 1995
- 432 RFCs about or affecting IPv6
  - 234 Standards Track
  - 163 Informational
  - 27 Experimental
  - 8 Best Current Practice
  - 46 Obsolete
- Numbers are a little old (mid 2015?)

Nothing I tell you is false, but not all may prove to be true.
Agenda

- IPv6 Basics
- Default Behaviour
- Infrastructure notes
- Tools and Technology
- My Goal
Basics
IPv6 Basics

- Size
- Representation
- Interfaces
- Transitions

RFC2460
Internet Protocol Version 6 Specification
IPv6 Basics - Size

- IPv6 addresses are 128 bits long
- Each Regional Internet Registry gets multiple /23 blocks
  - Divided into 512 /32 blocks
- One /32 for each ISP
  - Divided into 65,536 /48 blocks
- One /48 for each ISP's Customer
  - Divided into 65,536 /64 networks internally

Still in Flux? Some talk of /56 assignments
IPv6 Basics – Size (2)

- **All IPv6**: $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$
- **ISP**: $2^{96} = 79,228,162,514,264,337,593,543,950,336$
- **Customer**: $2^{80} = 1,208,925,819,614,629,174,706,176$
- **Smallest Subnet**: $2^{64} = 18,446,744,073,709,551,616$

*Wolfram Alpha*
Describe the amount of space in terms of the number of Internets!

Let $cI$ (classic Internet) = $2^{32}$

Therefore

Smallest IPv6 Subnet of $2^{64}$ hosts = $cI^2$

AKA

An Internet of Internets!
IPv6 Basics - Representation

- 8 groups of 16 bits each written as 4 hex characters
- From any group, leading zeros can be removed
  - 2001:34:0:0:0:FF45:A6B3:D3B
- Any consecutive set of zeros can be “collapsed” to “::”
  - 2001:34::FF45:A6B3:D3B

- Therefore Loopback:
  - 0000:0000:0000:0000:0000:0000:0000:0001
- Becomes:
  - ::1
IPv6 Basics – Representations (2)

- Networks are represented as CIDR Prefixes
  - 2001:2::/48 or 2001:4:112::/48

- Reserved spaces
  - 2000::/3 Global Unicast ( ~ cI^{3.90} )
  - FE80::/10 Link Scoped Unicast ( ~ cI^{3.69} )
    - i.e. FE80::1111%eth0 or FE80::1111%1
  - FF00::/8 Multicast
  - Others and subsets
• When writing your own code, remember to normalize your IPv6 addresses.

• Check that software you rely upon does as well.

• **Hint:** Try not to store them as strings

• But you already do this right?
IPv6 Basics – Representations (3)

• IPv4 Compatible addresses
  - Defined as ::/96 (::192.168.0.2 or ::C0A8:2)
  - Used to connect IPv6 over IPv4 networks
  - Deprecated

• IPv4 Mapped addresses
  - Defined as ::FFFF::/96
  - ::FFFF:192.168.0.2 or ::FFFF:C0A8:2
  - Used to connect IPv4 applications to IPv6 Sockets

Both of these are transition technologies
• By the way “::” is the IPv6 equivalent of “0.0.0.0”

• Also “::FFFF:” is the same as “::ffff:”
  • but capitals are easier to read
IPv6 Basics – Interfaces

• Each interface can have multiple IPv6 addresses
  – How many is OS dependent it seems
  – Linux configuration limit of 16
  – Sometime necessary to add %interface to the address
• “All modern devices run both IPv4 and IPv6”
  – Some VoIP phones do not
  – Android SLAAC only?
  – Desktop and server OS’s are OK
• All in addition to the usual IPv4 setup
IPv6 Basics – Transitions

- NAT64/DNS64
- 6to4 (community wants this deprecated 2011)
- ISATAP (look for prefix fe80::0200:5efe:)
- IPv6 Automatic Routing
- 6over4 (uses IPv4 multicast)
- Teredo (Miredo software last changes 2013?)
- PortProxy (MS only, not just IPv6)
NAT64 and DNS64

- Meant for pure IPv6 networks
- NAT64 ([RFC6146](https://tools.ietf.org/html/rfc6146))
  - An interface in each space
  - A gateway translating IPv6 to IPv4
- DNS64
  - A DNS server that returns temp IPv6 addresses for IPv4 only services
  - Used to get IPv6-only endpoints to the NAT64
Default Behaviour
Default Behaviours

• Stateless Address Auto-configuration (RFC4862)
• Privacy extensions (RFC4941)
• Neighbor Discovery Protocol (RFC4861)
  – Neighbour Solicitation and Advertisement
  – Router Solicitation and Advertisement
  – Redirection
Default - SLAAC

• Derived from IEEE addresses (if available)
  – 00:01:02:AA:BB:CC

• Split it in half, and insert “FF:FE”
  – 00:01:02:FF:FE:AA:BB:CC

• Flip the second last bit in the first eight
  – 02:01:02:FF:FE:AA:BB:CC (64 bit address)

• Prepend the prefix and rewrite
  – FE80::0201:02FF:FEAA:BBCC (128 bit link address)

SLAAC: Stateless Auto Address Configuration
• This is why the smallest network is a /64 or always will be

• RFCs state that a interface identifier should be used to generate a unique address.

• Hardware address is a natural fit, but there needs to be other ways (non-ether protocols?)
Default – SLAAC (2)

- Treat newly generated address as “Tentative”
- Perform Duplicate Address Detection (DAD)
- If no duplicate address detected
  - Assign to the interface
  - Set expiry (always “forever” on link-local)
  - Set preference (always “preferred” on link-local)
- Else
  - Stop (choke; no retry; no other MAC?)
• If you monitor for DAD and respond, you can DoS networks.

DAD: Duplicate Address Detection
Default – Privacy Extensions

• Create additional global addresses for new outbound connections!
• Make them temporary
  – a few hours to a few days
• Make them random
  – next address always unpredictable
  – But re-creatable!
• Each address is also a “listening” address (naturally)

• Privacy extensions are not on by default in Mac OS X.
Default – Neighbor Discovery Protocol

- Protocol for node communication on the same link (RFC 4861)
- Sort of like:
  - IPv4’s ARP
  - ICMP Router Discovery
  - ICMP Redirect
- Also has Neighbor Unreachability Detection
- Uses multi-cast addressing (not broadcast)
  - FF02::1 for all-nodes
  - FF02::2 for all-routers
  - And others...

All done via ICMPv6
• ICMPv6 provides much more functionality than ICMPv4. Be very careful what you choose to block at firewalls.

• But blocking unsolicited router advertisements is a good idea.
Default – Neighbor Solicitation and Advertisement

• Solicitations
  – Multicast ICMP to resolve an address
    • Multicast address FF02::1
  – Unicast to confirm reachability

• Advertisements
  – In response to solicitation
  – Without solicitation, used to propagate changes
There are many more.

If you ping6 these addresses, you will get interesting results! Including multiple “Duplicate” responses similar to pinging broadcast addresses.

These 3 should be particularly interesting to pentesters.

PS: there is no broadcast address any more
Default – Router Solicitation and Advertisement

• Same as Neighbor Solicitation and Advertisement except also contains:
  – address space advertised (global address prefix)
  – Preference/priority (high, med, low)
  – Router lifetime / prefix lifetime
  – Flags including “Managed” and “OtherConfig”

• Does not contain anything that could be considered “authentication” or “proof-of-source”
• 33:33:00:00:00:00 – 33:33:FF:FF:FF:FF are reserved for IPv6 Multicast at the ethernet layer.

• It is possible to (re)play an “advertisement” packet directly to an end node by changing the dest ether address to that of the end node.

• In the case of router advertisements, the end node could be tricked into assigning itself an address in a different prefix.
Default – DHCPv6

- RFC3315
- If router advertisement has “Managed” or “OtherConfig” flag set, get info from DHCPv6
- Requires endpoint configuration
  - Linux NetworkManager is SLAAC by default can be set to use DHCPv6 (ignores flags?)
  - DHClient must run a second instance with “-6” switch
Default – DUID

• The host identifier is this DHCP Unique Identifier
• In RFC3315
  – Type 1: Link Layer Address plus Time (DUID-LLT)
  – Type 2: Enterprise Number (DUID-EN)
  – Type 3: Link Layer Address (DUID-LL)
• In RFC6355
  – Type 4: A created UUID (DUID-UUID)

• DHCPv6 does not define the use of MAC as an identifier
• A lot of alternative servers popping up that do

DUID: DHCP Unique IDentifier
• In ISC DHCPv6 address reservations are set by DUID, not hardware address

• In some cases, hardware address is extractable from the DUID but not if DUID-EN or DUID-UUID is used.

• Ubuntu uses DUID-UUID
• Android doesn’t support DHCPv6

• But you can get hardware address from a DHCP proxy (same box?)
Infrastructure Notes

![Diagram of a gear shift with gears 1, 2, 3, 4, 5, and R (reverse).]
Infrastructures

- All-default (non-)configuration
  - AKA the “ignored”
- Minimal Configuration
  - AKA the “explored”
- Turned off and/or disabled
  - AKA the “removed”
- Complete Configuration
  - AKA the “matured”
• BTW, did I mention that there is no NAT?
Infrastructures - Ignored

• Devices have FE80:: addresses

• Devices are asking for router information
  – But not finding any

• Some internal communication
  – Service discovery

• No controls / detection mechanisms
  – And sometimes incapable controls (ie. Firewalls)

• Probably the majority for years to come
• Firewalls that cannot control IPv6 just let it though (Cisco FWSM)

• Blanket denies in IPv4 can be forgotten in IPv6

• iptables and ip6tables: two separate commands
Infrastructures - Explored

- Has a router advertising an IPv6 prefix
- Might have a DHCPv6 server
- Dual Stacks
  - Traffic defaults to IPv6 when available
- Set DNS server via prefix advert (RFC6106)
- Tunnelled?
PRO TIP!

- If you set your IPv6 router to the highest priority, it will be slightly harder to introduce a rogue router.

- Forging redirects may still be an effective attack even if you do the above.
Infrastructures - Removed

• “Don’t understand / can’t support. Therefore turning it off.”

• Easy to do with central configuration mgmt
  – Or via gold images?

• A little harder for IoT and “appliances”
  – There will always be something that remains

• Actively blocks / alerts on IPv6
If you choose to disable IPv6 on your network(s), then setting up alerts when IPv6 is detected is an excellent way to detect a rogue device.

Isolating devices that cannot have IPv6 turned off may also be a good idea as well.
Infrastructures - Matured

- One or more routers advertising large prefixes
- DHCPv6 possibly with dynamic DNS
- DNS64 and NAT64
- SEcure Neighbor Discovery (RFC3971 RFC6494)
- Ready to run as a pure IPv6 network
  only anecdotal; haven’t seen one

SEND: SEcure Neighbor Discovery
“Naming” servers and subnets is strongly suggested.

- If you have 2001:501:1DD:0000::/56
  - Workstations: 2001:501:1DD:1::/64
  - Guest: 2001:501:1DD:FF::/64

- Notice not using :0000::/64 to avoid possible confusion
- Also, easier to see where traffic is from/to
Tools and Technology
Tools and Technology

- SEND (RFC3971 RFC6494)
- THC-IPv6
- Scanners
- Python and Scapy Library
Secure Neighbor Discovery (SEND)

- Extended Neighbor discovery options that provide:
  - An authorization delegation discovery process
  - an address ownership proof mechanism
  - and requirements for the use of these components in Neighbor Discovery Protocol

- Requires pretty extensive PKI
  - Cryptographically Generated Addresses
  - Certificates and “trust anchors”
  - Signing all ND packets (in RSA with max 2048 bit?)
Secure Neighbor Discovery (SEND)

- RFC says “SEND is applicable in environments where physical security on the link is not assured …”
  - What did physical security look like in 2005?

- Do we trust the physical security of ANY network?

- Wireless is called out specifically as not physically secure (surprise)
THC-IPv6 Attack Framework

- https://www.thc.org/thc-ipv6/
- An excellent set of tools and libraries
  - Now with public development on github
- In my env’s some work and some do not
- Old, but kept up to date. v3.0 released this year.
- Strongly recommend running “alive6” as part of any recon.
Scanners

• Specifically calling out NMAP and MASSCAN

• NMAP
  – Scans known IPv6 targets as expected
  – Implements Neighbor Discovery scan as -6 -PR
  – Takes a long time to ND scan even a /64
  – Can address smaller than /64

• MASSCAN
  – Cannot do IPv6 today
  – But if it could, and it scaled linearly, a /64 would take 33 days with a single 10GBit interface
Python and Scapy library

- Scapy is a “packet rolling” library for python
- It has rich support for IPv6
- Very easy to use
- This is how to generate a Router Advertisement packet:

  ```python
  Ether()/IPv6()/ICMPv6ND_RA()/ICMPv6NDOptPrefixInfo(prefix='aaaa:aaaa:aaaa:aaaa::',prefixlen=64)
  ```
My Goals
Automated MITM

• Includes:
  – A rogue router
  – A dhcpv6 server
  – A NAT64/DNS64 install

• Interfaces
  – One IPv4 and One IPv6
  – Wifi connectivity

• Trying to see how far I can get with each of the infrastructure types
Automated MITM (2)

- Abilities:
  - Find and suppress installed routers (if necessary)
  - Find and suppress DHCPv6 servers (might not need to)
  - Target specific nodes or entire networks
  - Redirect traffic via local NAT64/DNS64 and do the usual tricks

- Today:
  - Advertise a specific prefix to any node chosen by MAC
  - Suppress targeted routers
    - but this breaks the network at the moment
Help?
Help?

- Transition technologies
- IPv6 and Mobile
- SEND implementation (and TCO)
- IPv6 and IoT
- Automated MITM
Epilogue
RFC1924 Base85

• Base85 representation of IPv6 addresses
• Makes this:
  – 2606:aa00:400:402:3008:f8b:5c7c:f645
• Look like this:
  – b7gxONQAg42t8cj~PXPU

• Turns out it was an April Fool’s RFC from 1996
  – I spent longer on this than I care to admit
Questions?

@allansto  allan.stojanovic(at)utoronto.ca
BACKUPS!
IPv6 Basics – Multi-Routing

- Each stack can have multiple DEFAULT routes
  - Plus routers can tell you to go elsewhere
- Plus each DHCPv6 static route
- Plus tunnel / transition routes
- Plus all the classic IPv4 routing stuff

More on this later
Home routing Multiple ISPs?
Shortness of IPv6 /64?
IPv6 on mobile
Transitions and Translations
Point to point encryption
SEND

IPv6mcast_ff:3d:66:8d (33:33:ff:3d:66:8d)

Android Cyanogen never makes a dhcpv6 request

IOT inventory no-v6 or half-v6?
PRO TIP!
IPv6 Default – Autoconfig

• SLAC
• permanent
• Temporary
• http://www.iana.org/assignments/ipv6-address-space.