

# Security Implications of the Internet Protocol version 6 (IPv6)

**Fernando Gont**

**UTN/FRH**

**BSDCan 2010**

**Ottawa, ON, Canada, May 13-14, 2010**



# Agenda

- Ongoing work on IPv6 security at UK CPNI
- Brief comparison of IPv4 and IPv6
- IPv6 addressing
- Fragmentation and Reassembly
- Internet Control Message Protocol version 6 (ICMPv6)
- Address Resolution
- State-less autoconfiguration
- Personal Rant on IPv6 security
- Questions and (hopefully) answers



# Ongoing work on IPv6 security at UK CPNI

(or “what we’re doing on v6 security”)



# Ongoing work on IPv6 security at CPNI

- The UK CPNI (Centre for the Protection of National Infrastructure) is currently working on a security assessment of the IPv6 protocol suite
- Similar project to the one we carried out years ago on TCP and IPv4:
  - Security assessment of the protocol specifications
  - Security assessment of common implementation strategies
  - Production of assessment/Proof-Of-Concept tools
  - Publication of “best practices” documents
- Currently cooperating with vendors and other parties
- If you're working on a IPv6 implementation, I'd like to hear from you



# **Brief Comparison of IPv4 & IPv6**

**(or “what the small differences are”)**

# Brief comparison of IPv4 and IPv6 (I)

- IPv4 and IPv6 are very similar in terms of functionality

	IPv4	IPv6
Addressing	32 bits	128 bits
Auto-configuration	DHCP & RS/RA	ICMPv6 RS/RA & DHCPv6 (opt)
Address resolution	ARP	ICMPv6
IPsec support	Optional	Mandatory
Fragmentation	Both in hosts and routers	Only in hosts

# Brief comparison of IPv4 and IPv6 (II)

- Header formats:

IPv4 Header

0	4	8	12	16	20	24	28	31
Version	IHL	Type of Service	Total Length					
Identification				Flags	Fragment Offset			
Time to Live		Protocol		Header Checksum				
Source Address								
Destination Address								

IPv6 Header

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	63
Version	Traffic Class		<i>Flow Label</i>					Payload Length			Next Header	Hop Limit				
Source Address																
Destination Address																



# **IPv6 addressing**

**(or “the actual motivator for IPv6”)**

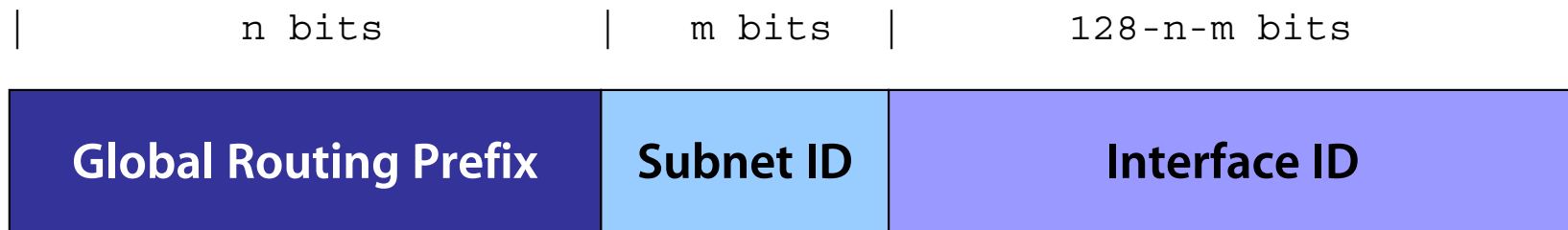


# Types of IPv6 addresses

- Unicast addresses
  - Identify a single interface
  - Packets are delivered to a single interface
- Multicast addresses:
  - Identify a set of interfaces
  - Packets are delivered to that set of interfaces
- Anycast addresses
  - Identify a set of interfaces
  - Packets are delivered to one interface of the aforementioned set
  - Syntactically indistinguishable from Unicast Addresses
- IPv6 has a Scoped Address Architecture, e.g., it supports:
  - Link-local addresses
  - Global addresses

# Global unicast addresses

- Address format:



- The Interface ID is typically 64 bits
- When stateless autoconfiguration is used for network interfaces that have Ethernet Addresses, the Interface ID is set to a value derived from that address (modified EUI-64 format)

# Global addresses & Reconnaissance

*Myth: "It is unfeasible to brute-force scan an IPv6 network for alive nodes, as the IPv6 address space is so large. Such a scan would take ages!"*

- [Malone, 2008] (\*) measured IPv6 address assignment patterns
- For hosts,
  - 50% autoconf, 20% IPv4-based, 10% Teredo, 8% "low-byte"
- For infrastructure,
  - 70% "low-byte", 5% IPv4-based
- Anyway, think about compromised hosts (e.g., botnets): once a host is compromised, brute-force scanning becomes trivial (sniffing, etc.)

***Size matters... only if you use it properly! ;-)***

(\*) Malone, D. 2008. *Observations of IPv6 Addresses*. Passive and Active Measurement Conference (PAM 2008, LNCS 4979), 29–30 April 2008.

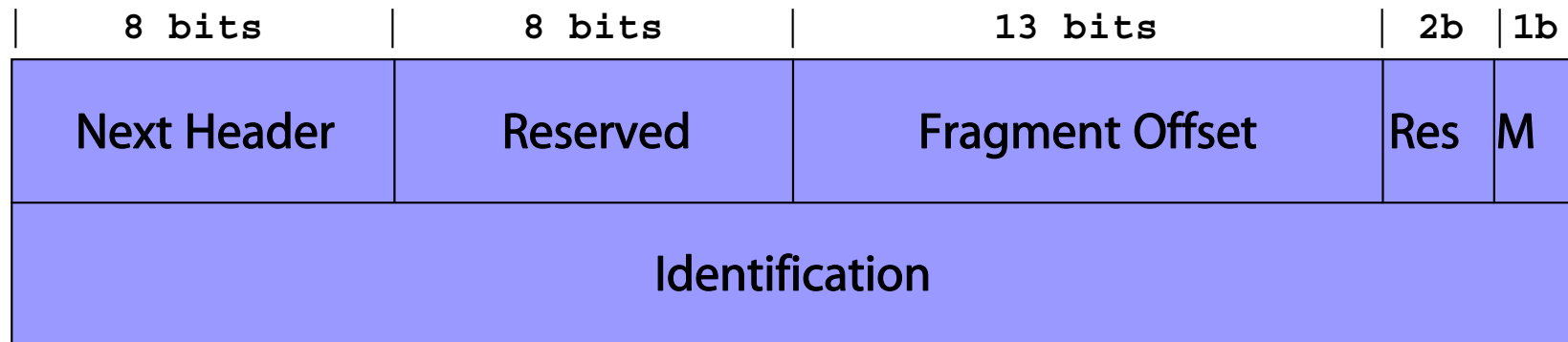


# **Fragmentation and Reassembly**

**(or “what we’re doing on v6 security”)**

# Fragmentation & Reassembly

- The fixed IPv6 header does not include support for fragmentation/reassembly
- If needed, such support is added by an Extension Header (Fragmentation Header)



- **Fragment Offset:** offset of the data following this header, relative to the start of the fragmentable part of the original packet
- **M:** "More Fragments" bit, as in the IPv4 header
- **Identification:** together with the Source Address and Destination Address identifies fragments that correspond to the same packet

# Security Implications of IPv6 fragmentation

- Some are the same as for IPv4 fragmentation:
  - Stateful operation for a stateless protocol: risk of exhausting kernel memory!
- Others are different:
  - The Identification field is much larger: chances of “IP ID collisions” are reduced
  - Not all packets carry an “Identification” number: hence it does not leak information so easily (e.g., think about “dumb scan”, etc.)
  - Overlapping fragments have been recently forbidden (RFC 5722) – although it’s unclear the benefits of this.

# sysctl's for frag/reassembly

- `net.inet6.ip6.maxfragpackets`: maximum number of fragmented packets the node will accept (defaults to 200 in OpenBSD and 2160 in FreeBSD)
  - 0: the node does not accept fragmented traffic
  - -1: there's no limit on the number of fragmented packets
- `net.inet6.ip6.maxfrags`: maximum number of fragments the node will accept (defaults to 200 in OpenBSD and 2160 in FreeBSD)
  - 0: the node will not accept any fragments
  - -1: there is no limit on the number of fragments



# **ICMPv6**

**(or “Internet Control Protocol version 6”)**



# Internet Control Message Protocol version 6

- ICMP is a core protocol of the IPv6 suite, and is used for:
  - Fault isolation (ICMPv6 errors)
  - Troubleshooting (ICMPv6 echo request/response)
  - Address Resolution
  - Stateless address autoconfiguration
- Contrary to ICMPv4, ICMPv6 is mandatory for IPv6 operation

# Fault Isolation (ICMPv6 error messages)

- A number of ICMPv6 error messages are specified in RFC 4443:
  - Destination Unreachable
    - No route to destination
    - Beyond scope of source address
    - Port Unreachable, etc.
  - Packet Too Big
  - Time Exceeded
    - Hop Limit Exceeded in Transit
    - Fragment reassembly time exceeded
  - Parameter Problem
    - Erroneous header field encountered
    - Unrecognized Next Header type encountered
    - Unrecognized IPv6 option encountered
- Clearly, most of them parallel their ICMP counter-parts

# ICMPv6 hard errors

- Some implementation could potentially extrapolate the concept of ICMP(v4) hard errors to ICMPv6 errors (for connections in the synchronized states)
- BSD-derived implementations don't – Good! ;-)

# ICMPv6 Packet Too Big

- ICMPv6 PTB messages are used for Path-MTU discovery
- The security implications of these messages are well-known (remember draft-ietf-tcpm-icmp-attacks back in 2004?)
- The mitigations are straightforward:
  - Check the embedded TCP SEQ and, even better, do not honor the ICMP PTB if there's progress on the connection (see draft-ietf-tcpm-icmp-attacks)
- Anyway, the MTU should not be reduced to a value less than 1280. If a smaller MTU is reported, the receiving node is just required to include a frag header.
- sysctl's (OpenBSD)
  - `net.inet6.icmp6.mtudisc_hiwat` (defaults to 1280): Maximum number of routes created in response to ICMP PTBs
  - `net.inet6.icmp6.mtudisc_lowat` (defaults to 256): Maximum number of routes created in response to (unverified) ICMP PTBs

# ICMPv6 redirects

- ICMP redirects are very similar to the ICMP counterpart, except for:
  - The Hop Limit is required to be 255
- ICMPv6 redirects are an optimization – hence they can be disabled with no interoperability implications
- Whether ICMPv6 are accepted is controlled in \*BSD's with the `sysctl net.inet6.icmp6.rediraccept`. In OpenBSD, it defaults to 1 (on).

# 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 (“-a” and “-b” options)

# Node Information Query/Response (II)

- 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`”).



# **Address Resolution**

**(or “mapping from IPv6 to link-layer”)**



# Address Resolution

- Employs the Neighbor Discovery Protocol (ICMPv6)
- Every node maintains a “Neighbor Cache”, which contains the mappings from IPv6 address to link-layer address, and the state (e.g., REACHABLE, STALE, etc.) of each entry.
- A node creates an entry in the Neighbor Cache for the target address (in the INCOMPLETE state), and sends a Neighbor Solicitation to the corresponding Solicited-node multicast address
- The target node responds with a Neighbor Advertisement that includes its link layer address
- The node stores the link layer address information in the corresponding Neighbor Cache Entry, and marks the entry as Reachable.
- Reachability information for Neighbor Cache entries is updated based on feedback received from the upper layer, or as a result of “probe” packets

# Some Address Resolution games

- Neighbor Cache Poisoning attacks – the v6 version of V4's ARP cache poisoning
  - The attacker simply listens to Neighbor Solicitations for Target addresses he is interested in, and responds with Neighbor Advertisements that contain his own link-layer address
- Advertising “special” link-layer addresses, e.g.,
  - The broadcast Ethernet address (ff:ff:ff:ff:ff:ff)
  - Multicast Ethernet addresses (e.g., 33:33:00:00:01)
  - The link-layer address of the node sending the Neighbor Solicitation – this introduces a forwarding loop if the victim is a router!
  - All BSD variants tested don't check for these special addresses!
- Not much support in layer-2 security boxes to mitigate these attacks
- Open source tools do exist. E.g., NDPMon, available at:  
<http://ndpmon.sourceforge.net>

# sysctl's for Neighbor Discovery (OpenBSD)

- `net.inet6.ip6.neighborgctresh` (defaults to 2048): Maximum number of entries in the Neighbor Cache
- `net.inet6.icmp6.nd6_prune` (defaults to 1): Interval between Neighbor Cache babysitting (in seconds).
- `net.inet6.icmp6.nd6_delay` (defaults to 5): specifies the `DELAY_FIRST_PROBE_TIME` constant from RFC 4861.
- `net.inet6.icmp6.nd6_umaxtries` (defaults to 3): specifies the `MAX_UNICAST_SOLICIT` constant from RFC 4861
- `net.inet6.icmp6.nd6_mmaxtries` (defaults to 3): specifies the `MAX_MULTICAST_SOLICIT` constant from RFC 4861.
- `net.inet6.icmp6.nd6_useloopback` (defaults to 1): If non-zero, uses the loopback interface for local traffic.
- `net.inet6.icmp6.nd6_maxnudhint` (defaults to 0): Maximum number of upper-layer reachability hints before normal ND is performed.



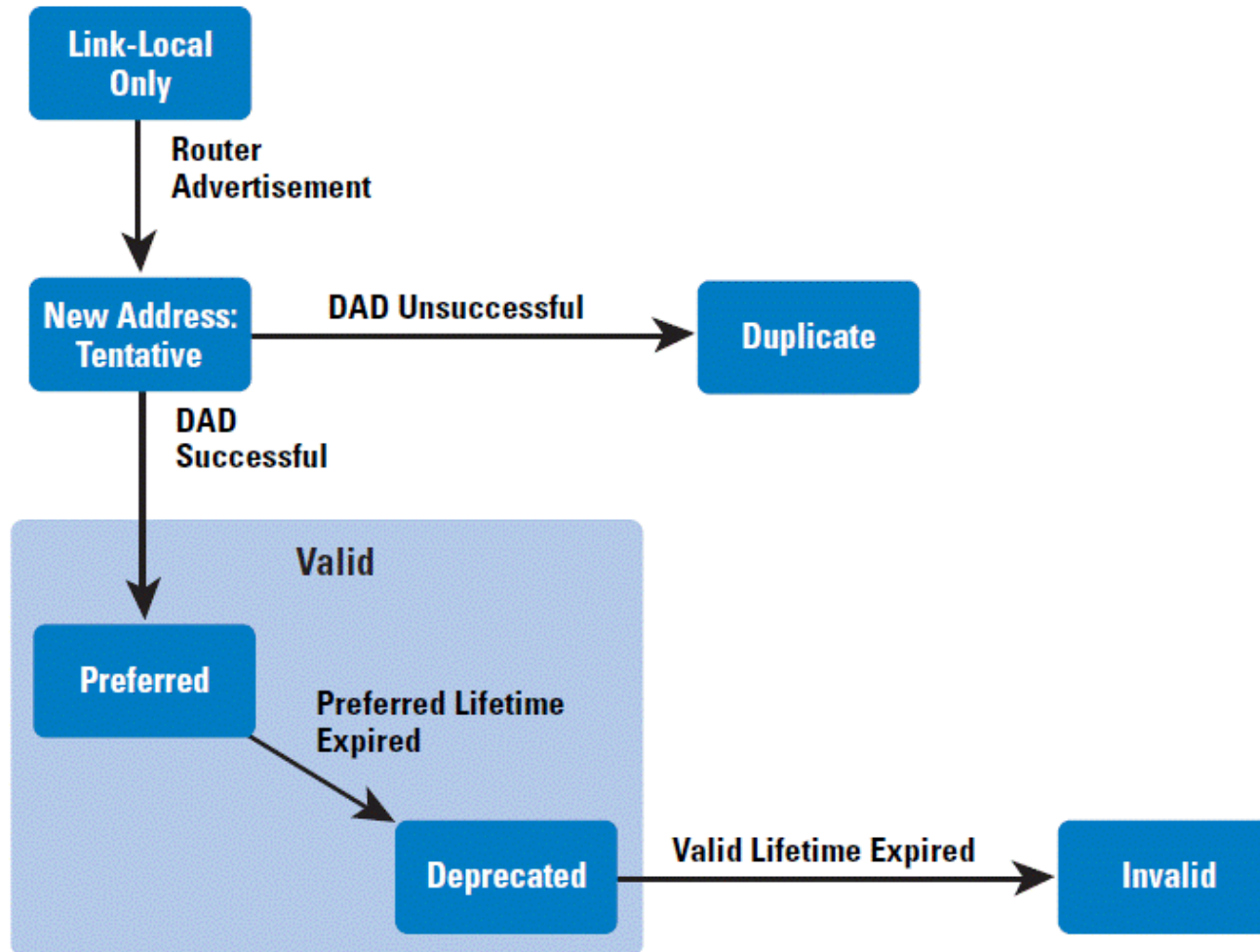
# **Stateless address autoconfiguration**

**(or “what we’re doing on v6 security”)**

# Auto-configuration

- Employs the Neighbor Discovery Protocol (ICMPv6 messages) – DHCPv6 is optional.
- Basic autoconfiguration
  - The node sends a multicast Router Solicitation message to the “all-routers”
  - Routers respond with prefixes for autoconfiguration
  - The node configures its own IPv6 address(es) with the advertised prefixes, plus a locally-generated Interface ID
  - Checks whether the selected address(es) are unique (Duplicate Address Detection)
  - If unique, the address is configured.

# Address autoconfiguration flowchart



# Other autoconf information

- Source Link-Layer Address option: advertises the link-layer address of the sender
- Prefix Information option: advertises “on-link” prefixes, and prefixes to be used for stateless address autoconfiguration.
- Route Information Option: Advertises “more specific routes”.
- Recursive DNS Server option: Advertises a “caching” DNS server
- MTU option: Advertises the MTU to be used for this link

# Some address autoconf games

- Rogue router: an attacker could send solicited/unsolicited Router Advertisements:
  - Advertise itself as a default router
  - Advertise bogus prefixes for on-link determination/autoconfiguration
  - Advertise more specific routes through his malicious node
  - Impersonate another router and cause victim nodes to remove it from their routing table
- Exploiting Duplicate Address Detection
  - Simply respond to all Neighbor Solicitations that are part of the DAD, and cause address autoconfiguration to fail
- Some (not all) of this vulnerabilities can be exploited with THC's "IPv6 attack suite"



# sysctl's for autoconf (OpenBSD)

- `net.inet6.ip6.accept_rtadv` (defaults to 1): Controls whether Router Advertisements are accepted.
- `net.inet6.ip6.dad_count` (defaults to 1): Number of DAD probes sent when an interface is first brought up
- `net.inet6.ip6.maxifprefixes` (defaults to 16): Maximum number of prefixes per interface.
- `net.inet6.ip6.maxifdefrouters` (defaults to 16): maximum number fo default routers per interface.

# Autoconf addresses & Privacy

- Addresses selected as part of stateless autoconfiguration contain a modified version of the MAC address of the interface
- The MAC address is globally-unique, and non-changing (OUI assigned by the IEEE to the vendor, plus a 3-byte number selected by the vendor)
- There were concerns that autoconf addresses hurt privacy, as they could be used to correlate network activity
- Privacy addresses (RFC 4941) were introduced for that purpose
  - They basically set the Interface ID to a random number, and are short
  - They are short-lived
  - They tend to be painful for the purpose of logging

# sysctl's for Privacy Addresses

- Privacy extensions for autoconf is implemented in FreeBSD (but not in, e.g., OpenBSD)
- These sysctl's control their operation:
  - `net.inet6.ip6.use_tempaddr` (defaults to 0)
    - Controls whether Privacy addresses are configured
  - `net.inet6.ip6.temppltime` (defaults to 86400)
    - Specifies the "preferred lifetime" for privacy addresses
  - `net.inet6.ip6.tempvltime` (defaults to 604800)
    - Specifies the "valid lifetime" for privacy addresses
  - `net.inet6.ip6.prefer_tempaddr` (defaults to 0)
    - Controls whether privacy addresses are "preferred" (i.e., whether outgoing "connections" should use privacy addresses)



# **Personal rant on IPv6 security**

**(or “what’s missing in the IPv6 arena?”)**

# Key areas in which further work is needed

- IPv6 Resiliency
  - Implementations have not really been the target of attackers, yet
  - Only a handful of publicly available attack tools
  - Lots of vulnerabilities and bugs still to be discovered.
- IPv6 support in security devices
  - IPv6 transport is not broadly supported in security devices (firewalls, IDS/IPS, etc.)
  - This is key to be able enforce security policies comparable with the IPv4 counterparts
- Education/Training
  - Pushing people to “Enable IPv6” *point-and-click style* is simply insane.
  - Training is needed for engineers, technicians, security personnel, etc., before the IPv6 network is running.



**Questions?**

# Acknowledgements

- UK CPNI, for their continued support
- BSDCan 2010 organizers, for their support to present at this conference

**Fernando Gont**

[fernando@gont.com.ar](mailto:fernando@gont.com.ar)

<http://www.gont.com.ar>