MLD Considered Harmful

Breaking Another IPv6 Subprotocol

Antonios Atlasis, aatlasis@secfu.net
Enno Rey, erey@ernw.de
Jayson Salazar, jsalazar@ernw.de

Who We Are

- Antonios
  - IT security enthusiast
  - Author of *Chiron*
- Enno
  - Old-school networking guy
- Jayson
  - Security researcher at ERNW

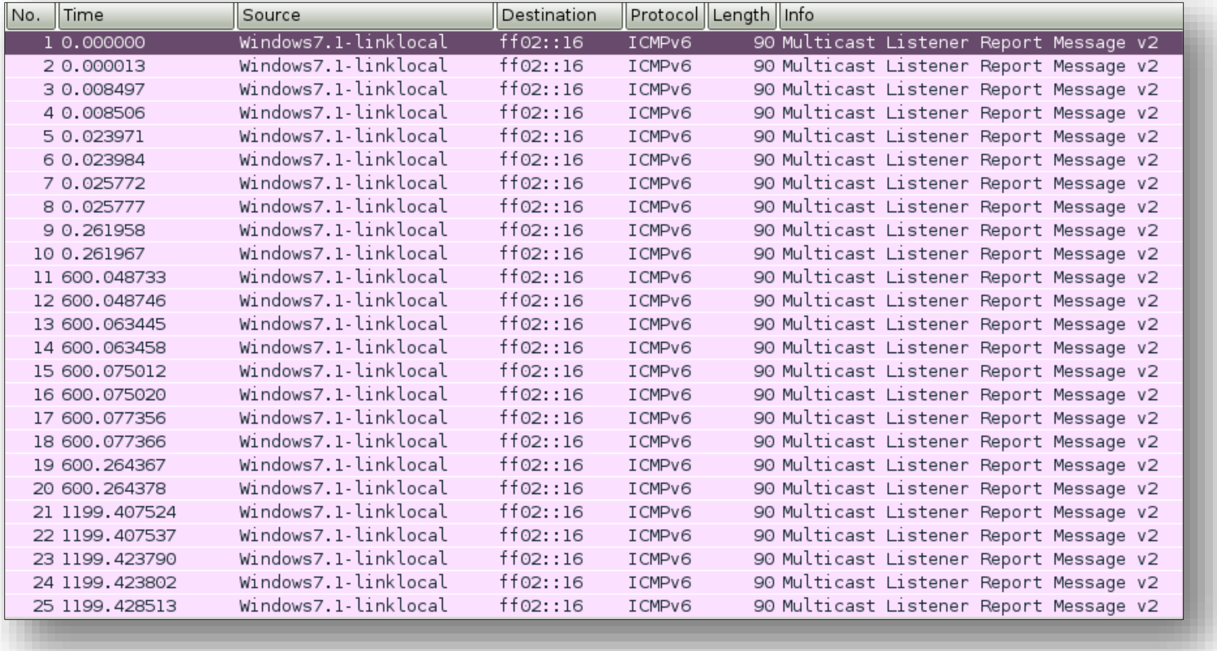
Research inside.™

Agenda (40 min + Q&A)

- The Object of Interest
- How We Tackled It
- What We Observed

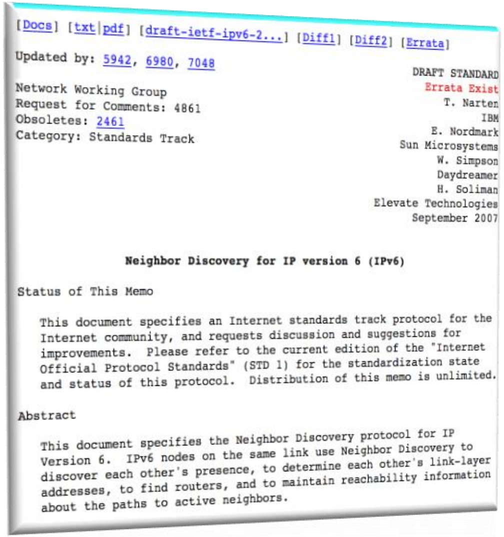
→ What All This Means

Why This Talk (I)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
2	0.000013	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
3	0.008497	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
4	0.008506	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
5	0.023971	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
6	0.023984	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
7	0.025772	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
8	0.025777	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
9	0.261958	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
10	0.261967	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
11	600.048733	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
12	600.048746	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
13	600.063445	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
14	600.063458	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
15	600.075012	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
16	600.075020	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
17	600.077356	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
18	600.077366	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
19	600.264367	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
20	600.264378	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
21	1199.407524	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
22	1199.407537	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
23	1199.423790	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
24	1199.423802	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
25	1199.428513	Windows7.1-linklocal	ff02::16	ICMPv6	90	Multicast Listener Report Message v2

Why This Talk (II)



[Docs] [txt|pdf] [draft-ietf-ipv6-2...] [Diff1] [Diff2] [Errata]

Updated by: 5942, 6980, 7048

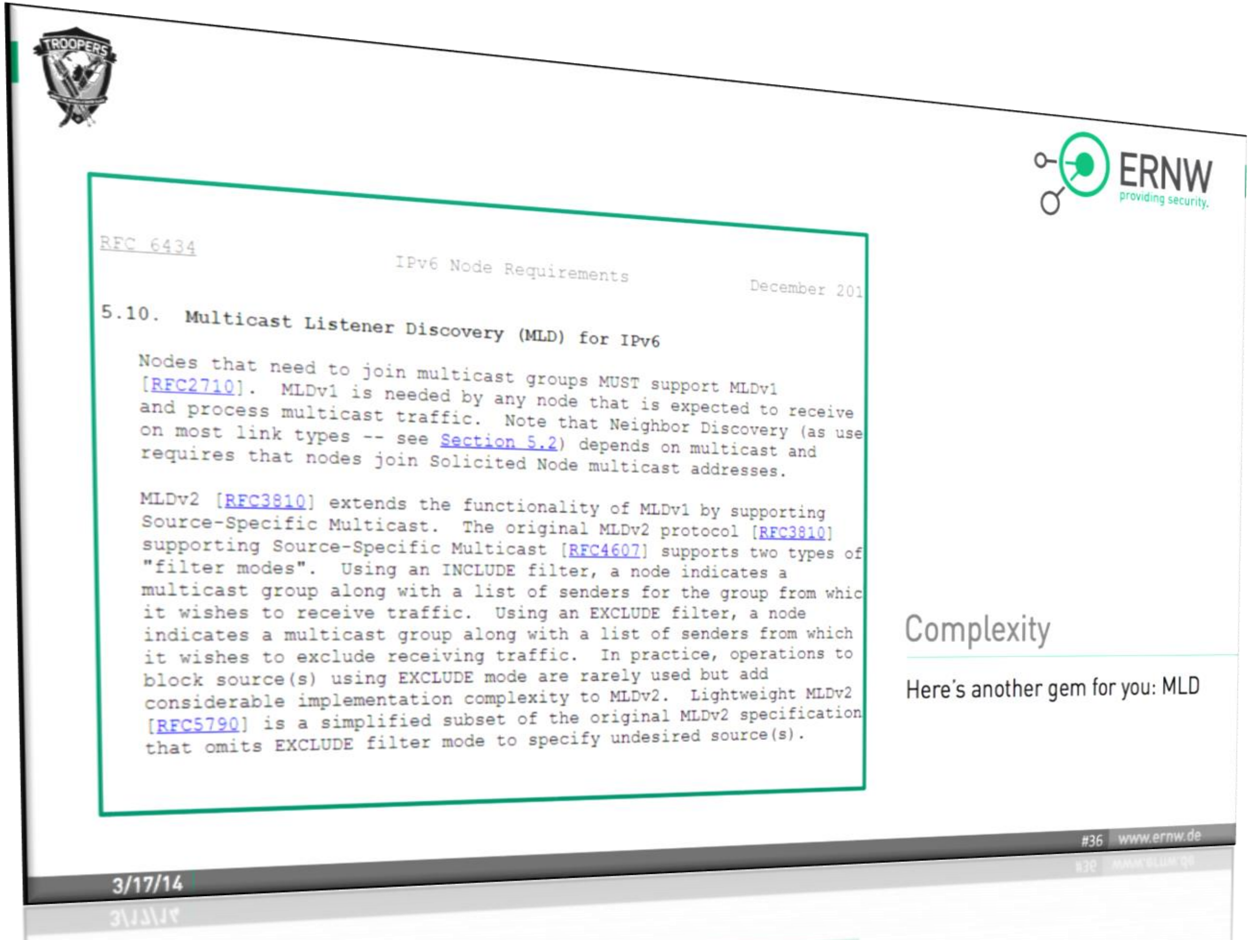
```
                                                          DRAFT STANDARD
                                                            Errata Exist
Network Working Group                                          T. Narten
Request for Comments: 4861                                           IBM
Obsoletes: 2461                                              E. Nordmark
Category: Standards Track                               Sun Microsystems
                                                              W. Simpson
                                                              Daydreamer
                                                               H. Soliman
                                                   Elevate Technologies
                                                          September 2007


              Neighbor Discovery for IP version 6 (IPv6)

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements.  Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol.  Distribution of this memo is unlimited.

Abstract

   This document specifies the Neighbor Discovery protocol for IP
   Version 6.  IPv6 nodes on the same link use Neighbor Discovery to
   discover each other's presence, to determine each other's link-layer
   addresses, to find routers, and to maintain reachability information
   about the paths to active neighbors.
```

RFC 4861 Neighbor Discovery for IP version 6 (IPv6), sect. 7.2.1

Descriptive or prescriptive ("normative")??

- "Joining the solicited-node multicast address is done using a Multicast Listener Discovery such as [MLD] or [MLDv2] protocols."

Why This Talk (III)



RFC 6434   IPv6 Node Requirements   December 201

5.10. **Multicast Listener Discovery (MLD) for IPv6**

Nodes that need to join multicast groups MUST support MLDv1 [RFC2710]. MLDv1 is needed by any node that is expected to receive and process multicast traffic. Note that Neighbor Discovery (as use on most link types -- see Section 5.2) depends on multicast and requires that nodes join Solicited Node multicast addresses.

MLDv2 [RFC3810] extends the functionality of MLDv1 by supporting Source-Specific Multicast. The original MLDv2 protocol [RFC3810] supporting Source-Specific Multicast [RFC4607] supports two types of "filter modes". Using an INCLUDE filter, a node indicates a multicast group along with a list of senders for the group from whic it wishes to receive traffic. Using an EXCLUDE filter, a node indicates a multicast group along with a list of senders from which it wishes to exclude receiving traffic. In practice, operations to block source(s) using EXCLUDE mode are rarely used but add considerable implementation complexity to MLDv2. Lightweight MLDv2 [RFC5790] is a simplified subset of the original MLDv2 specification that omits EXCLUDE filter mode to specify undesired source(s).

Complexity

Here's another gem for you: MLD

#36 www.ernw.de

3/17/14

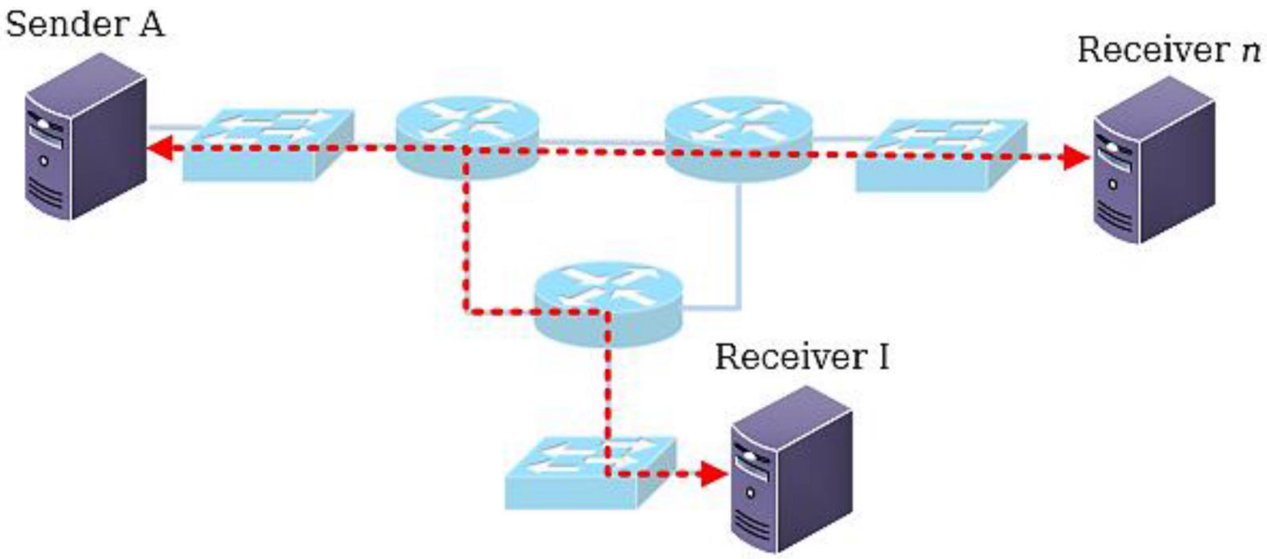
From:
https://www.troopers.de/wp-content/uploads/2013/11/TROOPERS14-Why_IPv6_Security_is_so_hard-Structural_Deficits_of_IPv6_and_their_Implications-Enno_Rey.pdf

So here's a Protocol…

- Apparently every IPv6 stack
  - has to support.
  - might have enabled by default (most do).
- It's not really clear if it is always needed or not.
- It's a complex beast (as we will see).
- Not much public sec research so far.
  - We'll close this gap today ;-)

MLD Fundamentals

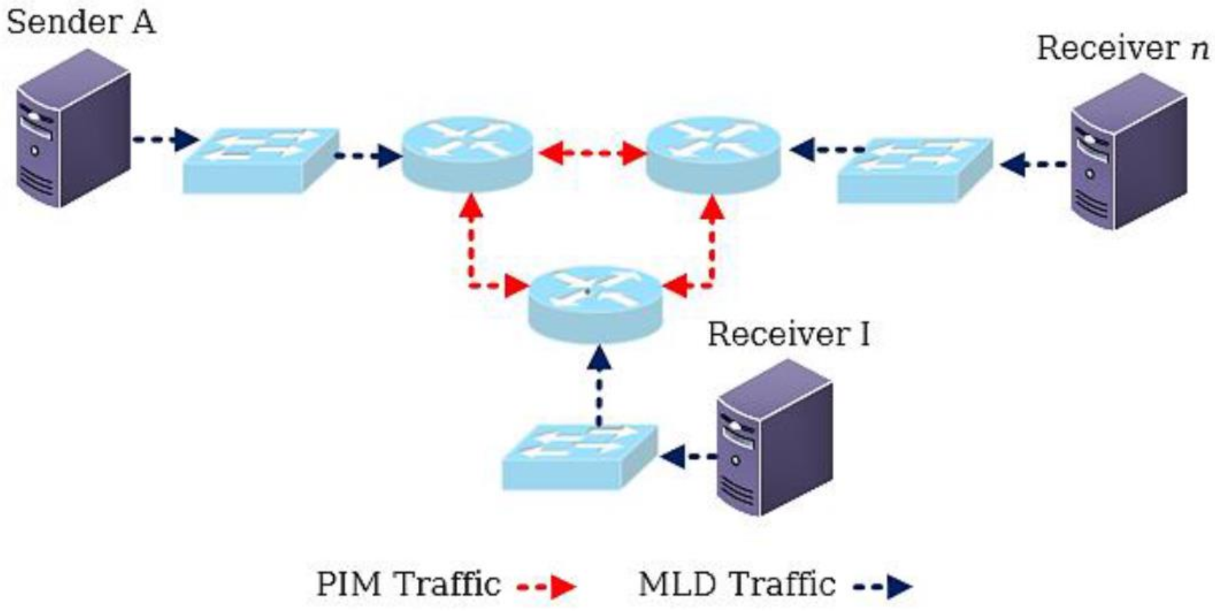
Multicast in a Nutshell (I)



Communication between a [group of] source[s] and several receivers.

Multicast in a Nutshell (II)

Receiver[s] have to signal to the routers that they´re interested in certain channels.



IPv6 Multicast Listener Protocol (MLD)

- Replaces IPv4's IGMP
  - MLDv1 (RFC 2710) based on IGMPv2.
  - MLDv2 based on IGMPv3.

- Queriers & Hosts
  - Querier: network device (usually a router) that sends *query* message to discover which network devices are members of a given multicast group.
  - Receiver: node that sends *report* messages to inform querier about a group membership.

MLD Version 1

- All MLD versions are based on ICMPv6.
- First defined in RFC 2710, derived from IPv4's IGMPv2.
- Used by IPv6 routers for discovering directly attached multicast listeners.
- In its original form MLD doesn't learn the exact identity or number of multicast listeners.

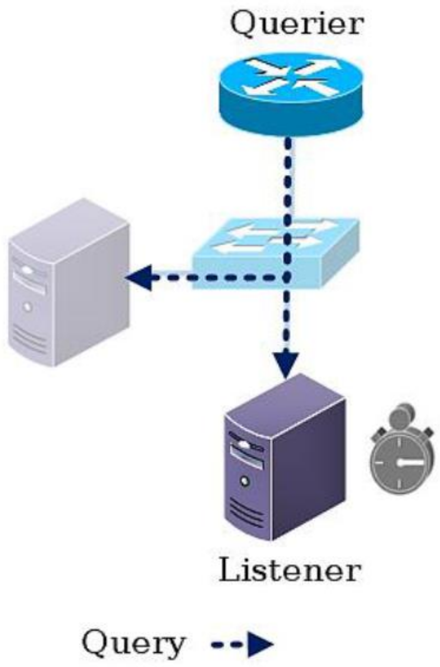
MLD Version 2

- Specified in **RFC 3810** and equivalent to IGMPv3.
- Designed to be **interoperable** with **MLDv1.**
- Adds support for "source filtering“. The nodes can report interest in traffic **only from a set** of source addresses or **from all except a set** source addresses.

MLDv1 Message Types

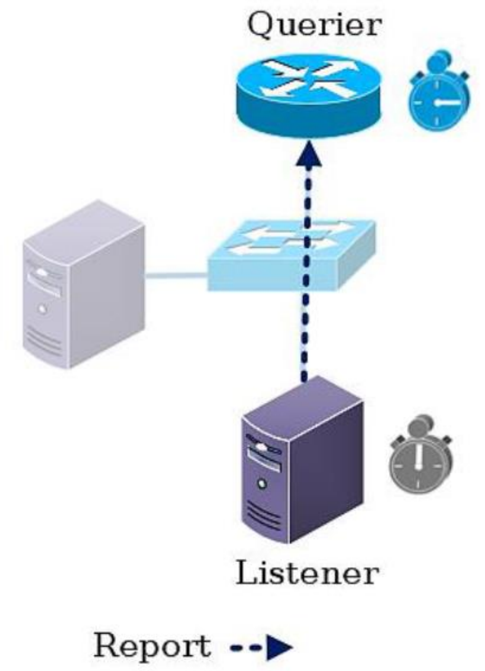
- Query (130)
  - General: Multicast address field set to 0 to learn which multicast addresses have listeners on an attached link.
  - Group/multicast-address specific.
- Report (131)
  - Sender of message (= a “receiver”) indicates which specific IPv6 multicast addresses it listens to.
- Done (132)
  - Sender of message (= a [former] “receiver”) indicates which address it no longer listens to.

MLDv1 Query Messages



- **General Queries**:
  - Asks all listeners about multicast addresses of interest.
  - Sent to **FF02::1** (link-scope all-nodes).
- **Multicast-Address-Specific Queries**:
  - Ask listeners about a particular multicast address.
  - Sent to the multicast address being queried.

MLDv1 Listener Messages



- **Multicast Listener Report**: ICMPv6 Type 131
  - Sent to the multicast address being reported.
- **Multicast Listener Done**: Type 132
  - Sent to **FF02::2** (link-scope all-routers).

MLDv2 Messages

- **General Queries:** ICMPv6 Type 130
  - Sent to **FF02::1.**
- **Specific Queries:** ICMPv6 Type 130
  - Inclusion of Address-and-Source-Specific queries.
  - All specific queries are sent to the multicast address being queried.
- **MLDv2 Reports** : ICMPv6 Type 143
  - Sent to **FF02::16** (all MLDv2-capable routers).
  - No more MLD *Done* messages.

One Particularly Interesting Functionality:

Last Call

aka [The *last listener query*]

MLD Snooping

- Switch based, somewhat proprietary feature that constrains multicast traffic to only the ports that have receivers attached.
- The switch builds an MLD table that basically maps a multicast group to all the switch ports that have requested it.

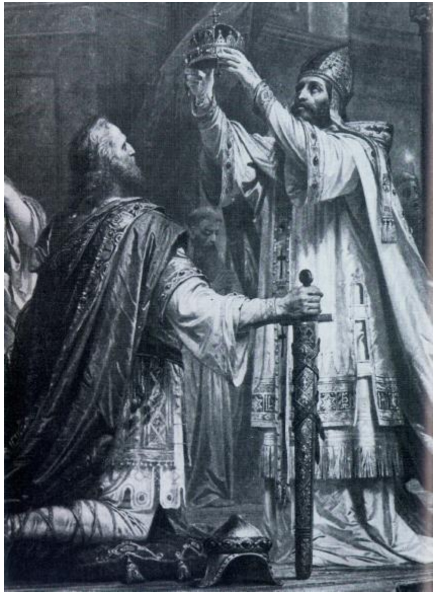
Security Precautions

- **All MLD messages must be sent with:**
  - A *link-local* IPv6 source address.
  - An IPv6 Hop-Limit of 1.
  - A *Router Alert Option* in the Hop-by-Hop extension header.
- Non compliant messages must be dropped.

Convenient RFC Conditions

- A node MUST process any *Query* whose destination address matches **any** of the addresses assigned to the receiving interface, unicast or multicast.
- **Result**: This allows one-to-one communication with the routers and listeners.

Convenient RFC Conditions (II)



- A router in querier mode enters the non-querier state upon receiving a query from a lower IPv6 address than its own. It thus ceases to send queries.
- **Result**: In most networks we can easily become a *Querier*.
  - → “Win the election“.

Convenient RFC Conditions (III)

- In the presence of MLDv1 Routers, MLDv2 hosts **MUST** operate in version 1 compatibility mode.

- In the presence of MLDv1 Multicast Address Listeners, an MLDv2 node **MAY** allow its MLDv2 Report to be suppressed by a Version 1 Report.

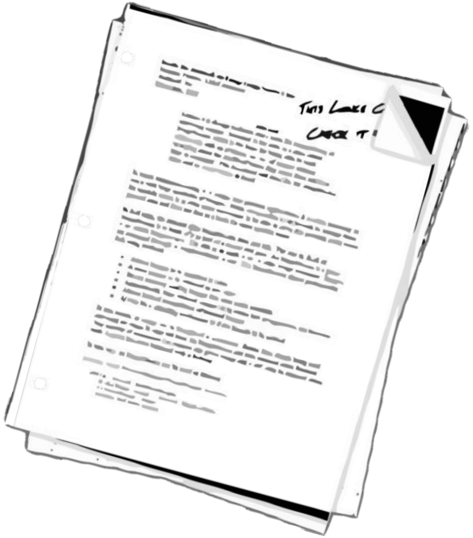
- **Result**: We can easily force MLDv1 to be used.
  - In the 90s we called this a “forced dialect downgrade“…

Myths and Facts

The Other Face of MLD

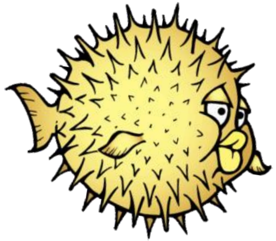
• MYTH •
DEBUNKED!
• MYTH •

Myths and Facts – MLD and ND



- Is **MLD required for Neighbor Discovery (ND)**?
- RFC 4861, par. 7.2.1: “joining the solicited-node multicast address **is done** using a Multicast Listener Discovery protocol such as the [MLD] or [MLDv2] protocols.”
- MLD cannot be disabled in most OSs.

Myths and Facts – MLD and ND (II)



- If disabled in Windows, the Neighbor Discovery (ND) process does **not** work.
  - RFC 4862, sect. 5.4.2: "In the case of Duplicate Address Detection, the MLD report message **is required** in order to inform **MLD-Snooping** switches, rather than routers, to forward multicast packets."
- However, if MLD messages are blocked by a host based firewall, ND works (even in Windows). See MLD/ND discussion on http://www.insinuator.net/.
- In OpenBSD, the ND process works normally without MLD being enabled.

Myths and Facts - MLD and ND (III)

- Moreover, when MLD-Snooping is enabled on a Cisco Catalyst 2960-S switch (at least with certain images), *solicited-node* multicast addresses are still broadcasted. Maybe tools.ietf.org/html/draft-pashby-magma-simplify-mld-snooping-01 is implemented?

So, do we really need MLD for Neighbor Discovery?

Implementation Facts

- MLD is pre-enabled in Windows, Linux and FreeBSD Operating Systems. It is **NOT** in OpenBSD.
- MLD Reports are sent even before the Neighbor Discovery Process starts.
- To cover the possibility of the initial Report being lost or corrupted, it is recommended that it be resent once or twice after short delays.

Implementation Facts (II)

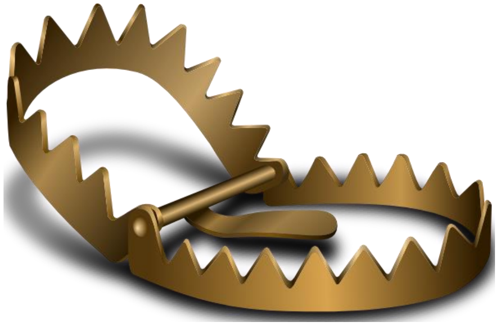
- All of them join several multicast groups:
  - Each OS joins the corresponding Solicited-Node Multicast Address.
  - Windows joins **FF02::1:3** (Link Local Multicast Name Resolution).
  - FreeBSD joins Node Information Queries multicast groups (**experimental** RFC 4620).

Global Unicast Address as Destination?

- All but FreeBSD accept the Queries and respond.
- This means that we can interact directly with nodes without the Router/Querier being involved.

Homework: MLD-Related Vulnerabilities

- Six MLD-related CVEs as of Nov 2014.
- Four related to Cisco products (2012,2013,2014)
  - Two of them when MLD Snooping is enabled, one related with VRFs and one with MLD tracking.
- One on NetBSD 4.0, FreeBSD, and KAME (2008).
- One on Windows XP, 2003 and Vista (2007).



Same Procedure Every ~~Year~~ Protocol

- Read the specs
- Build a lab
- Create a test plan
- Have fun ;-)

What To Look For

- Implementation problems
  - Yes, fuzzing.
  - We mean, what else ;-)
- RFC compliance issues
  - These may sound lame… but we´ll see that they can serve as a stepping stone for the next category.
- Design flaws & unwanted/-expected protocol behavior.

Devices Used in the Lab

- For routers: mainly Cisco 1921, IOS15.4(3)M, plus an ASR 1002.
- For switches: Cisco Catalyst 2960-S IOS 15.2(1)E3.
- As hosts: latest Windows (server, desktop), some Linuces, FreeBSD and OpenBSD.

Tools

Our approach

- Chiron
  - Abusing the protocol
  - Antonios added MLD capabilities → New version available: http://www.secfu.net/tools-scripts/

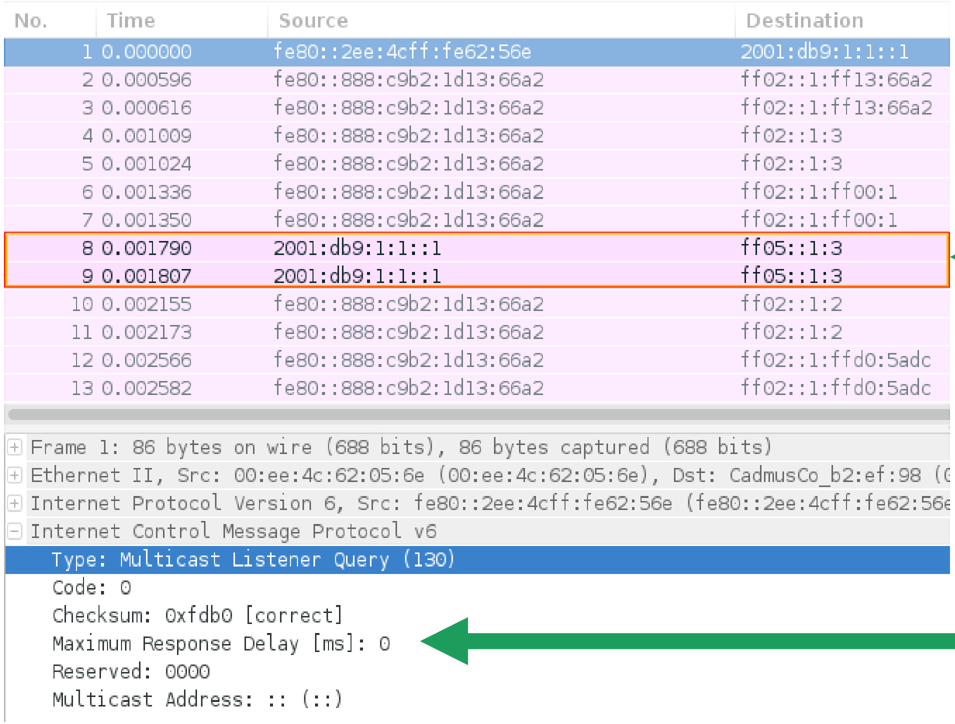
- Dizzy
  - Fuzzing
    - Latest version: http://www.insinuator.net/2014/02/fresh-meet-from-the-coding-front
  - New description files for MLD available → Released after the talk.

Results

RFC Compliance Issues, Linux

- Linux systems up to kernel v3.16 accept MLD messages with **Hop Limit > 1**.
- This is also the case for MLD messages with no Router Alert Option (but not that important).
- Centos 6.x accepts MLD messages when the source address is a **link-local multicast** one.

Windows



```
No.   Time        Source                     Destination
   1 0.000000    fe80::2ee:4cff:fe62:56e    2001:db9:1:1::1
   2 0.000596    fe80::888:c9b2:1d13:66a2   ff02::1:ff13:66a2
   3 0.000616    fe80::888:c9b2:1d13:66a2   ff02::1:ff13:66a2
   4 0.001009    fe80::888:c9b2:1d13:66a2   ff02::1:3
   5 0.001024    fe80::888:c9b2:1d13:66a2   ff02::1:3
   6 0.001336    fe80::888:c9b2:1d13:66a2   ff02::1:ff00:1
   7 0.001350    fe80::888:c9b2:1d13:66a2   ff02::1:ff00:1
   8 0.001790    2001:db9:1:1::1            ff05::1:3
   9 0.001807    2001:db9:1:1::1            ff05::1:3
  10 0.002155    fe80::888:c9b2:1d13:66a2   ff02::1:2
  11 0.002173    fe80::888:c9b2:1d13:66a2   ff02::1:2
  12 0.002566    fe80::888:c9b2:1d13:66a2   ff02::1:ffd0:5adc
  13 0.002582    fe80::888:c9b2:1d13:66a2   ff02::1:ffd0:5adc
```

```
⊞ Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
⊞ Ethernet II, Src: 00:ee:4c:62:05:6e (00:ee:4c:62:05:6e), Dst: CadmusCo_b2:ef:98 (
⊞ Internet Protocol Version 6, Src: fe80::2ee:4cff:fe62:56e (fe80::2ee:4cff:fe62:56e
⊟ Internet Control Message Protocol v6
    Type: Multicast Listener Query (130)
    Code: 0
    Checksum: 0xfdb0 [correct]
    Maximum Response Delay [ms]: 0
    Reserved: 0000
    Multicast Address: :: (::)
```

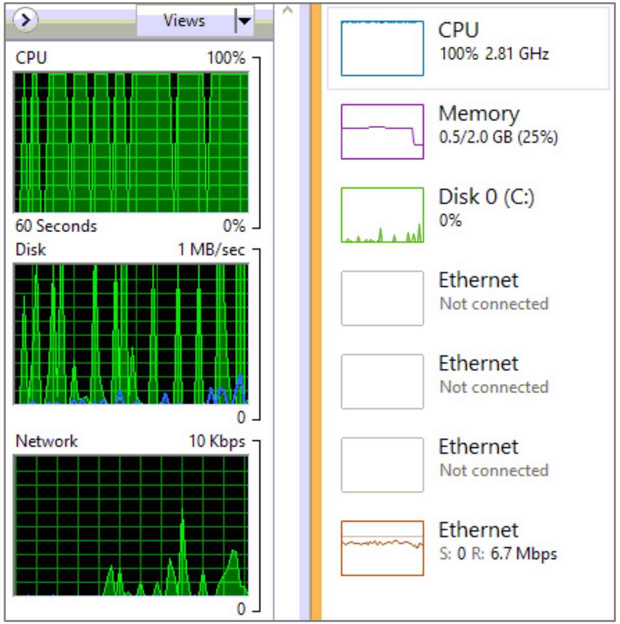
In certain cases Windows 8.1 responds with a report from its global unicast address.

MLDv1 query sent to the unicast address of a Windows 2012R2 DHCPv6 Server.

Why the Source Address Matters

- When an MLD Report with a non link-local address as source is received:
  - In MLDv2, it is strictly defined that it MUST be dropped.
  - In MLDv1 it is not strictly mentioned, but if accepted this would mean that we would be able to interact with routers remotely.

CPU Overload of Virtual Windows Guests



- Overload CPU of virtualized Windows guests
  - 100% utilization & some disk usage spikes.
  - Suspect: Interrupt handling.
- Maximum effect with MLDv1 generic queries.
  - MLDv2 messages have no effect.
- VProc : Intel Core TM(2) Quad Q9550@ 2.83GHz.

Router Ressource Depletion via Joins

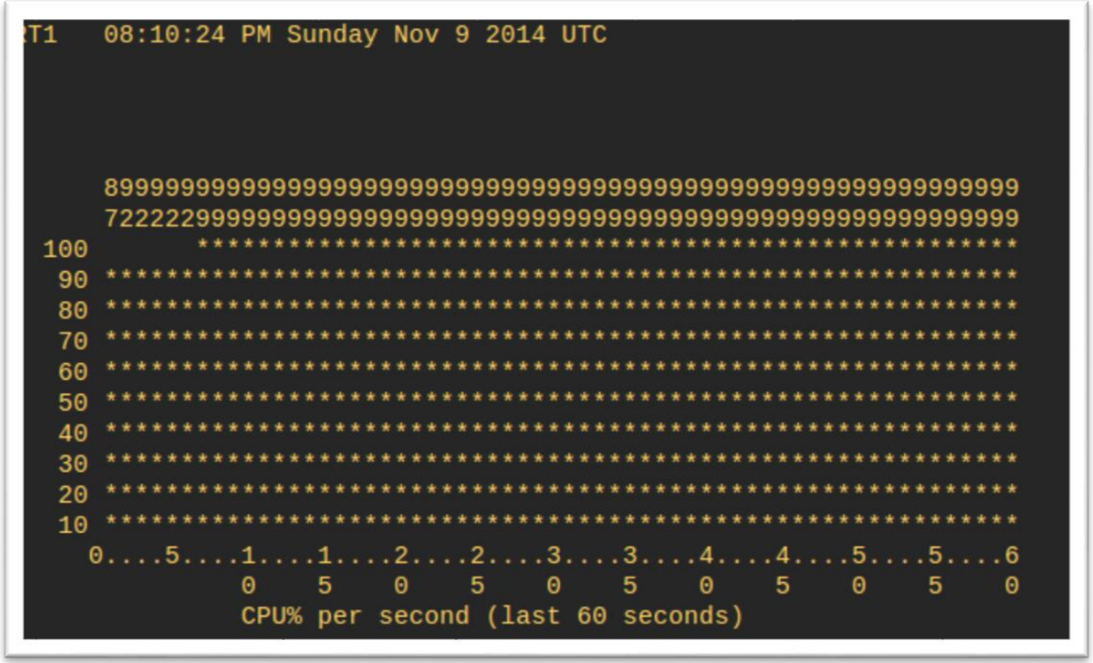
- **Results**: Memory depletion with about 100KB left.
- No further memory allocation possible for other processes.
  - E.g. 1.7Mbps of MLDv2 Reports can turn a Cisco 1921 into a brick.
- **Mitigation**: Set a state limit for the MLD process.
- **Drawback**: Once limited, it's easy to fill the MLD cache preventing new joins of multicast groups.
- * **FF02::** multicast groups are handled specially.

Flooding

Demo

Heavy Resource Consumption (II)

Here, the router is a Cisco ASR 1002, there´s only one attacker on *local-link*…



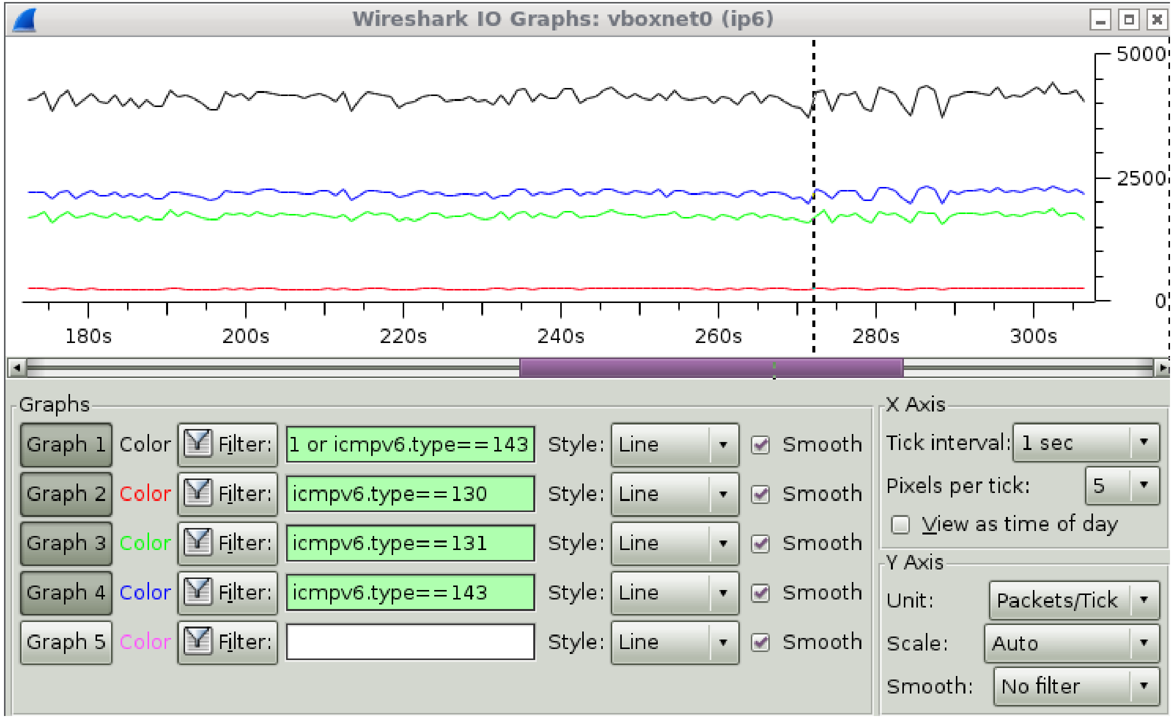
```
T1    08:10:24 PM Sunday Nov 9 2014 UTC




    899999999999999999999999999999999999999999999999999999999999
    722222999999999999999999999999999999999999999999999999999999
100       ******************************************************
 90 ************************************************************
 80 ************************************************************
 70 ************************************************************
 60 ************************************************************
 50 ************************************************************
 40 ************************************************************
 30 ************************************************************
 20 ************************************************************
 10 ************************************************************
   0....5....1....1....2....2....3....3....4....4....5....5....6
             0    5    0    5    0    5    0    5    0    5    0
             CPU% per second (last 60 seconds)
```

Amplification Attacks

Against the routers on the *local-link* using MLD Queries.

- Windows 8.1 hosts join at least four groups and send two Reports per group.
  - Amplification factor goes up to **8 x Number of machines** for Windows hosts.
    - For example, in a LAN with 200 hosts a single spoofed Query can trigger 1600 Reports all sent immediately to the router.
      - Did you get that? Amplification factor: 1.600!!
- What if we flood the link with such Queries?

Amplification Queries vs. Reports



Wireshark IO Graphs: vboxnet0 (ip6)

Graphs

Graph 1 Color Filter: 1 or icmpv6.type==143 Style: Line ☑ Smooth

Graph 2 Color Filter: icmpv6.type==130 Style: Line ☑ Smooth

Graph 3 Color Filter: icmpv6.type==131 Style: Line ☑ Smooth

Graph 4 Color Filter: icmpv6.type==143 Style: Line ☑ Smooth

Graph 5 Color Filter: Style: Line ☑ Smooth

X Axis

Tick interval: 1 sec

Pixels per tick: 5

☐ View as time of day

Y Axis

Unit: Packets/Tick

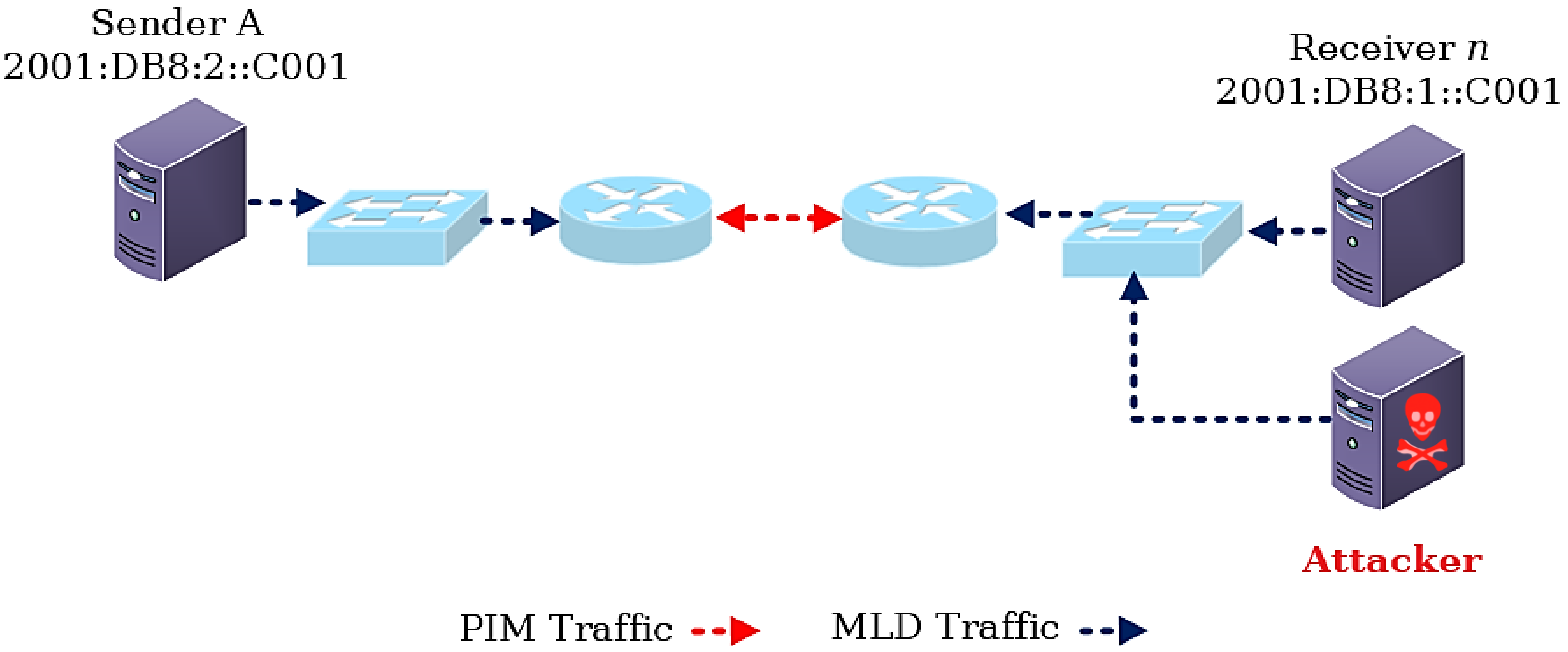
Scale: Auto

Smooth: No filter

How to Break MLD

- Cisco IOS15.4(3)M accepts:
  - MLDv1 and MLDv2 Queries sent to **FF02::2.**
  - MLDv2 Queries to **FF02::16** and its unicast address.
  - MLDv1 and MLDv2 Queries to its link-local address.
  - MLDv2 Reports sent to **FF02::2** and **FF02::16.**
  - MLDv1 Dones sent to the **FF02::2**, **FF02::16**, link-local and unicast addresses.
- **Result**: We have several ways to interact with the routers in a **one-to-one** manner.

Let's Have a Look at a Practical Attack



Attack Vector I - MLDv1 and MLDv2

- Take over the Querier Role
- Send spoofed MLDv1 Done or MLDv2 Reports to remove a listener from a multicast group.
- Send a spoofed Last Listener Query to the routers, they believe this to be a real Last Listener Query.
- Periodically send Generic Queries to the routers (**FF02::2, FF02::16** or their unicast addresses).

Attack Vector II – MLDv1

- Become Querier through MLDv1 Queries, forcing use of MLDv1. Same can be done by sending MLDv1 Reports.
- Send MLDv1 Done messages.
  The Querier (or you) sends a “last call” Query.
- Send MLDv1 Report to the unicast address of the legitimate listeners to trigger Report supression on their side.
- Legitimate routers do not receive any Reports and thus traffic to the group is no longer forwarded.

Real Life Call

- Cisco Catalyst 2960-S with IOS version 15.2(1)E3 **blocks Last Listener Queries** when MLD Snooping is **NOT** enabled (by default).
- If a router is in Querier mode, it sends periodical queries.
- When in Non-Querier state, it sets-up a timer for each group (~4 min).
- Groups are removed if no Reports are received before timer expiration.
- If Reports are received, the timer is reset to its initial value.

Non-Compliance Makes Things Easy

- Attacker takes over the Querier Role by sending Queries continuously to the **unicast address** of the legitimate querier (to avoid triggering Reports)
- Attacker sends an MLD Report (Include:None) for the desired multicast group
- We wait for the timeout to expire. But after that, it's removed for good

Non-Compliance Makes Things Even Easier

a) Attacker sends an MLD Report (**INCLUDE:NONE**) for the specific multicast group. Inter-domain multicast traffic is stopped immediately

b) Immediately after that, attacker takes over the Querier Role (to prevent legitimate router from sending periodic Queries) by sending Queries continuously to the unicast address of the legitimate querier

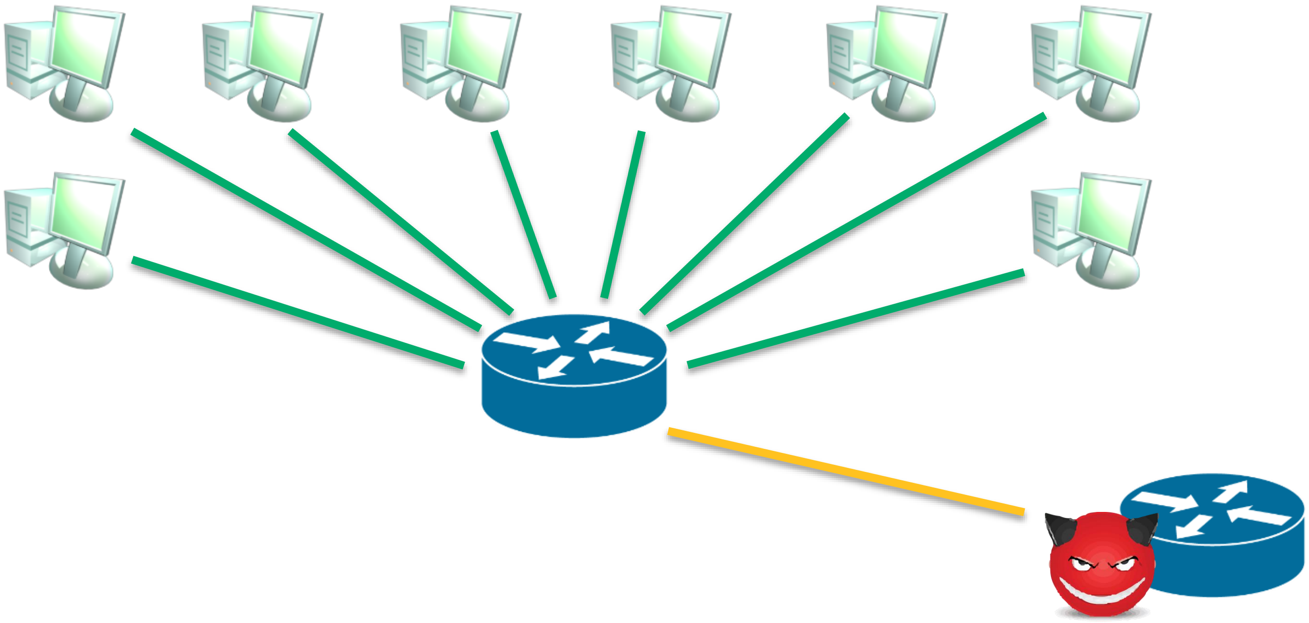
¬ Inter-domain traffic stops being forwarded and it doesn't recover

Real Life Scenario:

Shareholders' Meeting

Demo

The Show Must Go Offline



What if MLD Snooping Activated

¬ For the next demo we enable MLD-Snooping, this lets last listener queries pass

a) Take over the Querier role

b) Send an MLDv2 Report Include(None)

c) Send two last listener queries to the routers (unicast address or **FF02::16**)

¬ **Result**: The above procedure does not remove the group from the switch, it removes it from the router. This leads to desired multicast traffic being blocked

¬ **Reminder**: In Demo 1, steps **a** and **b** were swapped and we did not need step **c**

Let's Talk About MLD Snooping

- Our switch with MLD-Snooping forwards traffic going to:
  - **FF02::1** (all nodes at link-local) to all nodes, normal
  - Solicited-node multicast addresses also to all nodes (to avoid accidentally breaking ND and overloading the switch)
  - **FF02::2, FF02::16** only to dynamically discovered routers
- When a router stops sending queries, it does not receive any traffic going to **FF02::2** or **FF02::16**

Let’s Talk About MLD Snooping (II)

- Cisco’s MLD-Snooping is “smart”
  - For example, it uses PIM “Hello” messages to identify routers on the link, it seems to block multicast groups such as ff02::1:x.
  - That's really bad, services (like DHCPv6) that rely on such addresses (**FF02::1:2**) do not operate properly without intervention.
- Is this “RFC compliant”, should we call this a feature?
- Why are RFCs considering proprietary features before they even become a draft?

Mitigation

ERNW's *Seven Sisters* of Infrastructure Security

Access Control

Isolation

Restriction

Encryption

Entity Protection

Secure Management

Visibility

See also: http://www.insinuator.net/tag/seven-sisters/

General Use

- The building blocks can be “applied“ to all components / technologies / protocols.

Just ask yourselves:

- What is the “scope“? Can it be limited?
- Can (the traffic) be filtered / restricted?
- Are there authentication mechanisms?
- How‘s the stuff being managed?
- Any hardening (of a device or service) possible?
- What about logging / monitoring?

Mitigations for Admins

- Filter MLD Queries on the switch port level
  - Think "MLD Guard" (which does not exist).
  - = Port based ACL filtering ICMPv6 type 130
    - `deny icmp any any mld-query`
- Alternatively, in a MLD snooping scenario statically configure a port as an **mrouter** port.

Mitigations for Admins (III)

- At routers specify a limit on the rate that MLD Reports should be accepted from each host. MUST drop all the reports that exceed this limit.
- Consider “`no ipv6 mld router`“ if there‘s no inter-domain multicast routing in the environment.

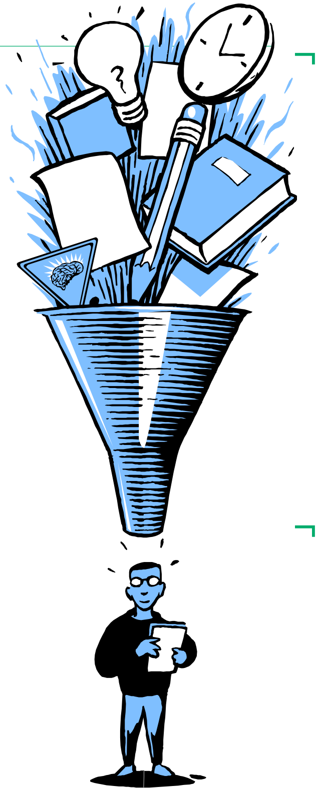
Mitigations for Admins (IIII)

- At switches with MLD-snooping enabled:
  - You might use *static-groups* to protect critical multicast based services (e.g. DHCPv6)
    - Keep operational impact/effort in mind ;-)
  - MLD snooping listener message suppression is enabled by default → forwards **only one** MLD report per response to multicast router queries.
  - If technically possible, limit the rate at which MLD messages are accepted by nodes.

In the Standards Space

- **MLDv2**: Routers shouldn't accept Queries destined to **FF02::2**, **FF02::16**, or unicast addresses (link-local or global).
- **MLDv1**: Nodes MUST not accept Reports to their unicast addresses (not even for debugging purposes).
- **Both**: Do not permit querier role take over by simply using a "lower" IPv6 address.

Conclusions



- In the IPv6 world there's a protocol called MLD.
  - It's complex & somewhat flawed, we think.
  - It's ubiquitous.
  - There's quite some potential for abuse
    - Huge local amplification attacks.
    - Disruption of network services.

- Security research in the IPv6 world is much needed.
  - And it's fun. Get your hands dirty.

There's never enough time...

THANK YOU...

...for yours!

Tool & Slides:

https://www.insinuator.net

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

Questions?

- You can reach us at: ✉
  - aatlasis@secfu.net, www.secfu.net
  - erey@ernw.de, www.insinuator.net
  - jsalazar@ernw.de, www.insinuator.net
- Follow us at:
  - @AntoniosAtlasis
  - @Enno_Insinuator

Guys, we would love to see you in Heidelberg!

March, 16-20 2015
Heidelberg, Germany
Make the world a safer place.

REGISTRATION OPEN: www.troopers.de

One more thing...

Some more stuff we couldn‘t include in the talk, for time reasons

Reconnaissance

- **Passive**:
  - Sniff the wire and identify routers, Windows and FreeBSD machines (at least).
- **Active**:
  - If you can't wait send MLD Queries and identify
    - ALL hosts → **FF02::1**
    - Routers-only → **FF02::2** or **FF02::16**
    - Most likely Windows-only → **FF02::1:3** or **FF02::C.**

Demo

Because Scanning a
Complete IP Range is so 1990

Non-link-local Address as Source

- In case of Windows, no reports are sent but the ND cache is poisoned.

- This happens when MLDv2 Queries or Reports are sent using:
  - A unicast address as a destination address but a multicast MAC as a destination MAC address.
  - A unicast address as a destination address and the target's MAC as a destination MAC address.

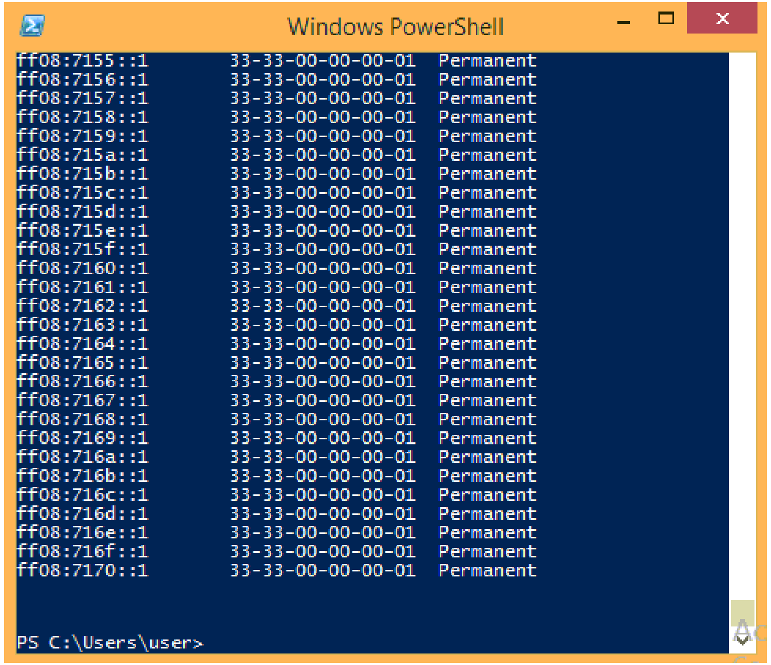
Non-link-local Address as Source

- If the destination address used is unicast a stale record with **00:00:00:00:00:00** layer-2 address is added to the ND Cache.

- If the destination address is a multicast one, a permanent record with the respective layer-2 multicast address is added instead.

- If the IPv6 address used as destination is registered in the windows ND cache its record state changes to stale and is resolved through ND once windows tries to transmit data to it.

Populating the Windows ND Cache



```
ff08:7155::1       33-33-00-00-00-01  Permanent
ff08:7156::1       33-33-00-00-00-01  Permanent
ff08:7157::1       33-33-00-00-00-01  Permanent
ff08:7158::1       33-33-00-00-00-01  Permanent
ff08:7159::1       33-33-00-00-00-01  Permanent
ff08:715a::1       33-33-00-00-00-01  Permanent
ff08:715b::1       33-33-00-00-00-01  Permanent
ff08:715c::1       33-33-00-00-00-01  Permanent
ff08:715d::1       33-33-00-00-00-01  Permanent
ff08:715e::1       33-33-00-00-00-01  Permanent
ff08:715f::1       33-33-00-00-00-01  Permanent
ff08:7160::1       33-33-00-00-00-01  Permanent
ff08:7161::1       33-33-00-00-00-01  Permanent
ff08:7162::1       33-33-00-00-00-01  Permanent
ff08:7163::1       33-33-00-00-00-01  Permanent
ff08:7164::1       33-33-00-00-00-01  Permanent
ff08:7165::1       33-33-00-00-00-01  Permanent
ff08:7166::1       33-33-00-00-00-01  Permanent
ff08:7167::1       33-33-00-00-00-01  Permanent
ff08:7168::1       33-33-00-00-00-01  Permanent
ff08:7169::1       33-33-00-00-00-01  Permanent
ff08:716a::1       33-33-00-00-00-01  Permanent
ff08:716b::1       33-33-00-00-00-01  Permanent
ff08:716c::1       33-33-00-00-00-01  Permanent
ff08:716d::1       33-33-00-00-00-01  Permanent
ff08:716e::1       33-33-00-00-00-01  Permanent
ff08:716f::1       33-33-00-00-00-01  Permanent
ff08:7170::1       33-33-00-00-00-01  Permanent



PS C:\Users\user>
```

- We haven’t managed to control the MAC address, hence not suitable for MiTM.

- Nonetheless, we can implicitly force ND by using an address which already is in the cache.