

# **Are you ready for IPv6 insecurities ?**

v1.1

George Kargiotakis

kargig@void.gr  
Berlin, RSS 2013

```
$ id
```

```
uid=1000(kargig) gid=1000(sysadmin)
```

```
groups=1(HELLUG),2(HTFv6),3(Hackerspace.gr),4(DLN.gr)
```

```
$ last kargig
```

```
GRNET - System Administration
```

```
Gennet - Linux-based broadband CPEs (IPv6 capable)
```

```
University of Ioannina - System Administration
```

```
$ apropos kargig
```

```
iloog - Greek gentoo-based livecd
```

```
GrRBL - Greek AntiSpam Blacklists
```

```
Greek Adblock Plus filter - self-explanatory
```

```
void.gr - My Blog
```

How Many of you

- ~~know what IPv6 is ?~~
- have used/are using IPv6 at work/home ?
- **know what SLAAC is ?**
- have used/are using transition mechanisms ?
- **have deployed services over IPv6 ?**
- are using native IPv6 ?
- **are using native IPv6 and have applied IPv6 specific security policies on servers/routers ?**

- Fast IPv6 crash course (still needed)
- Main Dish

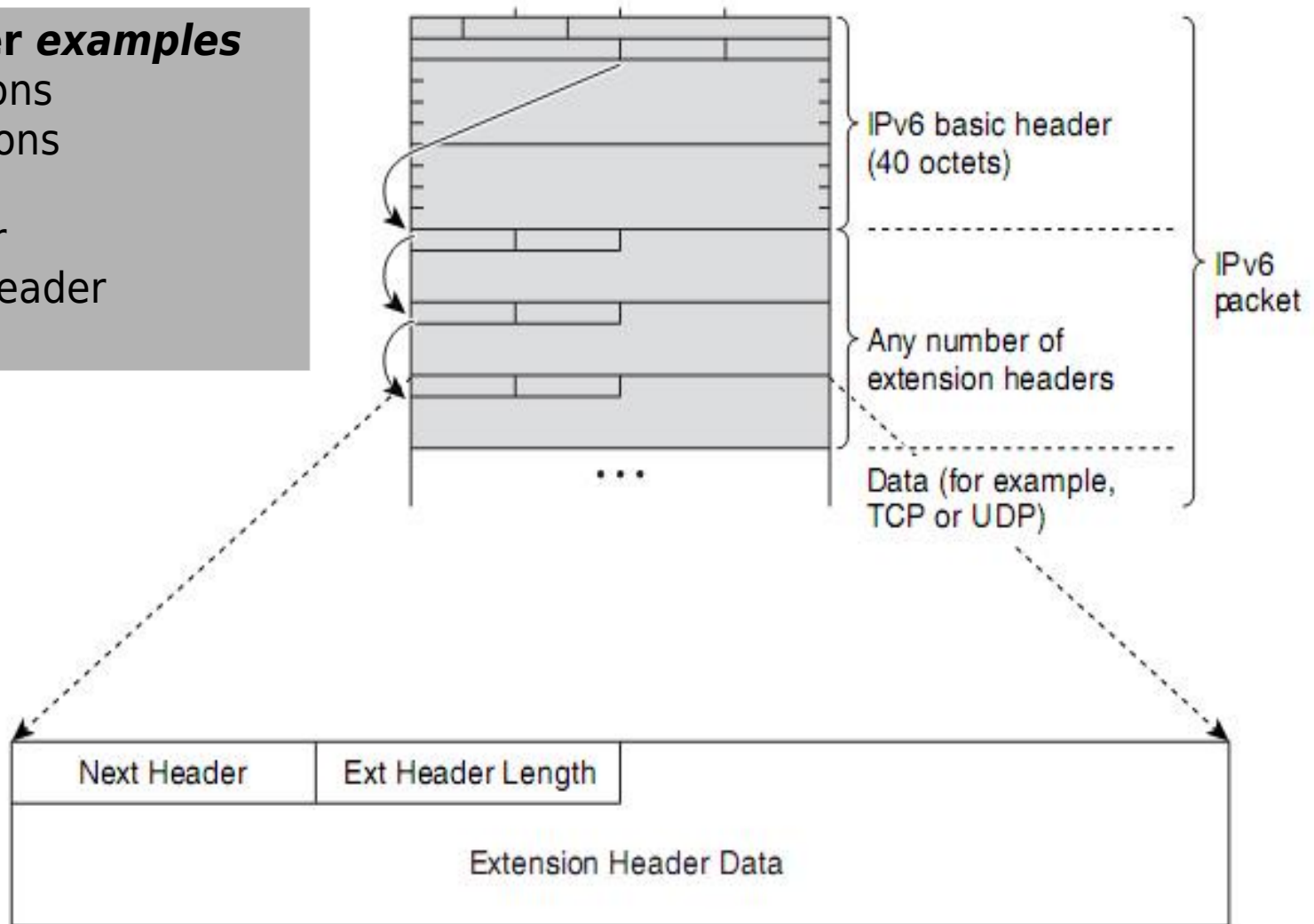
## Embrace the new

- 128-bit addr – 340.282.366.920.938.463.463.374.607.431.768.211.456 IPs
- Multiple IPs (w/ different scopes) per Interface
- Lots of Multicast (no more broadcast!)
- Network Discovery Protocol → Address Auto-configuration
- Simpler Header (definitely good!)
  - + Extension Headers (sounds promising!)
  - + Header Daisy Chaining (interesting!)
    - == a complete mess

## Extension Header Daisy Chaining

### Extension header examples

- Destination Options
- Hop-by-Hop Options
- Routing Header
- Fragment Header
- Authentication Header
- Mobility



## Commonly used Neighbor Discovery messages

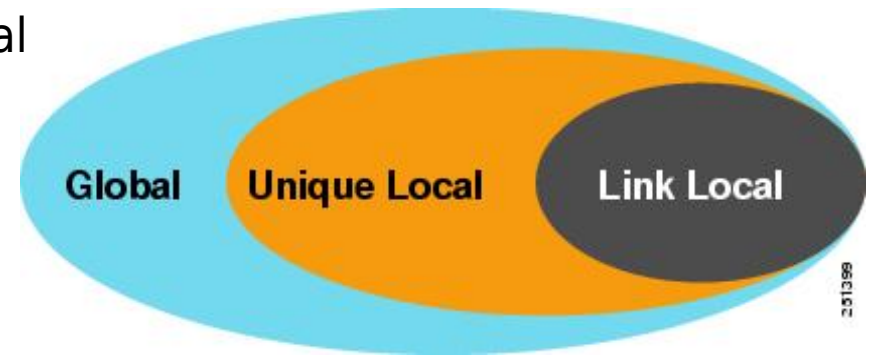
- Router **A**dvertisement (Type 134)
- Router **S**olicitation (Type 133)
- Neighbor **A**dvertisement (Type 136)
- Neighbor **S**olicitation (Type 135)

## Benefits of IPv6 ND

- Formalize Address Resolution + Router Discovery (Security at layer 3 independent of IPsec → SeND)
- Autoconfiguration
- Dynamic Router Selection
- Multicast

## Address Types

- Unicast: Link Local, Unique Local(\*), Global
- Multicast
- Anycast
- Reserved



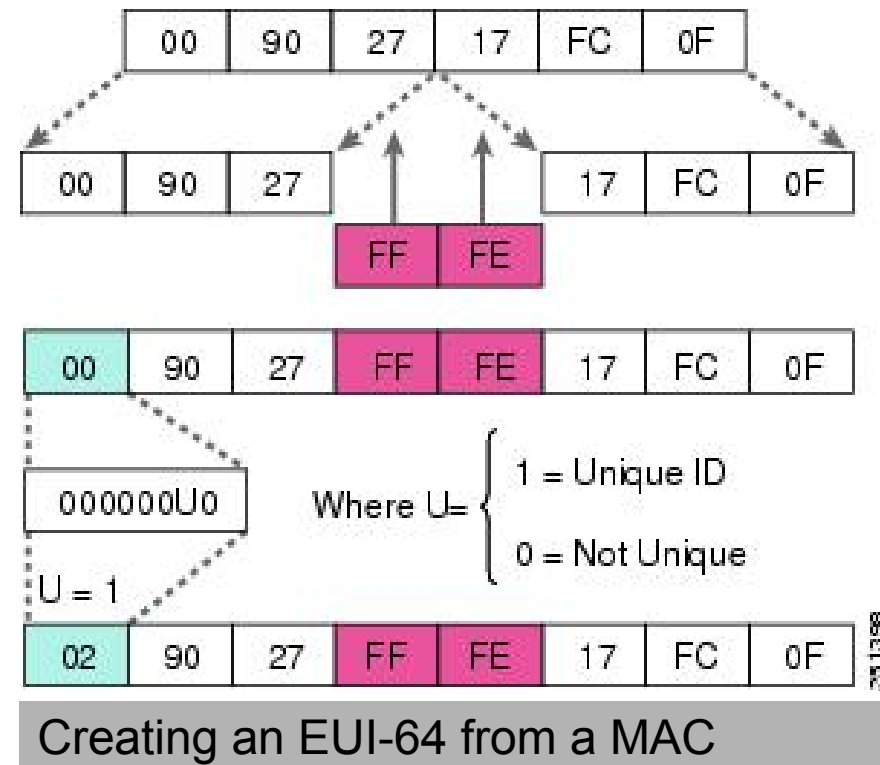
**\*ULA** → Global, but **non-internet-routable** addresses  
 → Unique Private address range per organization

Address Type	Prefix
Unspecified	:: (or ::/128)
Loopback	::1 (or ::1/128)
Multicast	FF00::/8
LL Unicast	FE80::/10
ULA Unicast	FC00::/7
Global Unicast	All the rest...



## SLAAC (Stateless Address Autoconfiguration) rfc4862

- Multicast ICMPv6
- EUI-64: Create last 64bits of address from 48bit MAC
- Router Advertisement gives:
  - IPv6 Prefix(es)
  - Default Router
  - MTU
  - Lifetime
  - DNS/DNS search list
  - Other Config (!)



**EUI-64:** 00:90:27:17:FC:0F → 0290:27**FF:FE**17:FC0F

## The SLAAC problem

- **MAC** gives us **EUI-64** → last 64bits of address are always the same → IPv6 “super cookie” → **Privacy issues!**

## Proposed solution

- **Privacy Extensions** (RFC4941) → Randomize last 64bits of address → **Temp. Addresses**

Temp. Addresses change every X secs (e.g. 600,1800,3600..)

- **New problem:** How to monitor local users with Temp. Addresses if they keep changing ? → *'FX' has an idea...*
- **New Proposal** (?): “Stable but random per Prefix” last 64bits (draft-gont-6man-stable-privacy-addresses-00 - Dec 2011)

## Stateful DHCPv6

- Model: Client/Server
- Transport: Multicast UDP
- Provides: Addressing, Routing, DNS, SIP, NTP, etc options
- Addresses: Temporary (IA\_TA) & non-temporary (IA\_NA)
- **(NEW) Prefix Delegation** (IA\_PD): Request prefix from ISP to provide addressing for LAN

## Stateless DHCPv6

- Get IP address by SLAAC + OtherConfig Flag "O"=1
- Extra configuration params (e.g. DNS) by DHCPv6

## DNS is extremely important!

**Browsers:** http://[2001:1af8:4100:a02c:1::16] ← **Global**

**Shell:** scp kargig@[fdbf:468f:aaa0:474d:ab01::ff]:file.ext localpath/ ← **ULA**

**Shell:** ssh kargig@fe80::a80c:eaff:feda:b0db%**eth0** ← **LL**

## AAAA forward record (hostname → IPv6 address)

```
void.gr.    IN    AAAA    2001:1af8:4100:a02c:1::16
```

## PTR reverse record (IPv6 address → hostname)

```
6.1.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa. IN PTR void.gr
```

## Main Dish

- IPv6 Security Hype
- Common Local Attacks & mitigation
- Fragmentation issues
- Network Scanning
- IDS/Firewalling - OS Support - IPv6 Migration Security
- Scanning IPv6 Internet
- Tools
- Food for thought - IPv6 Security Overview
- (+Bonus Slides)

# *RSS 2013* IPv6 Security Considerations



IPsec is mandatory!!111oneoneone

- NOT! IPsec **support** is mandatory, not usage!

No more ARP spoofing!!11eleveneleven

- Yeah, they are now called ND (local) attacks...  
and they are probably worse...

My fridge/toaster will be accessible from the Internet! Save meeeeeee!!

- It won't → Stateful firewalls/ACLs





## **Neighbor Cache Poisoning (DoS)**

- Attacker sends RA/RS/NA/NS and fakes source/target LL addr
- Victim can't connect to faked LL addr

## **Duplicate Address Detection (DoS)**

- Attacker replies to any victim's DAD requests
- Victim can't access the network

## Fake router (DoS)

- Attacker sends fake **R**outer **A**dvertisements with a certain prefix and blackholes traffic directed there

## Address Configuration (DoS)

- Attacker cancels previous default router prefix (lifetime=0) and sends new (fake) prefix to victim
- Victim can't access the network due to spoofed prefix filtering by default router.

## Spread ~~Misconfiguration~~Mayhem (DoS)

- Discard default router → set R Flag to 0 → RAs are there...but where did my router go ?
- Play with CurHopLimit, MTU, etc in RAs → keep admins busy trying to figure out network issues
- Force DHCPv6 → M/O Flag → can force request of DNS servers over DHCPv6 → ;)

## Address Resolution (MITM)

- Attacker claims victim's IP address and replies to **Neighbor Solicitation** requests. Get's all traffic.

## Fake Router Redirect (MITM)

- Attacker sends **Router Advertisement** and redirects traffic heading to an **off-link** host/prefix elsewhere (or himself)

## **First-Hop Router Attack (MITM)**

- Attacker sends **RA** and tricks victim into accepting himself as a default router by canceling the previous one (prefix lifetime=0) first. Steals all traffic.

## **DHCPv6 spoofing**

- Same way like DHCPv4

## Freezing/Crashing Operating Systems

- Abusing the limits of *Neighbor/Destination Cache* or *Routing table* entries can lead to freezes or crashes
- ➔ Victims include:
  - iOS/OS X
  - Windows XP → Server 2012 (semi patched)
  - Free/NetBSD
  - Linux/Android
- Use THC-IPv6 **flood\_router26** to freeze them

## This is **not** a Linux kernel bug!!

- Ubuntu 12.04/12.10 have Privacy extensions enabled by default

```
# tcpdump -ni eth0 host 2a00:1450:4002:800::100e
17:57:37.784658 IP6 2001:db8:f00:f00:ad1f:9166:93d4:fd6d > 2a00:1450:4002:800::100e: ICMP6, echo request
```

- Flood rogue RAs → **disabling** of Priv. Ext upon reaching 16addr/iface (default)

```
[ 1093.642065] IPv6: ipv6_create_tempaddr: retry temporary address regeneration
[ 1093.642067] IPv6: ipv6_create_tempaddr: regeneration time exceeded - disabled temporary address support
# tcpdump -ni eth0 host 2a00:1450:4002:800::100e
17:59:38.204173 IP6 2001:db8:f00:f00:5054:acff:fe8b:995d > 2a00:1450:4002:800::100e: ICMP6, echo request
```

- Dave Miller replies on netdev:

*“A malicious user who can emit random packets as root on your LAN can also corrupt your ARP cache with entries that point to the wrong MAC address. What's your point?” (!? #wtf)*

- But...taking down the interface and up again:

```
# tcpdump -ni eth0 host 2a00:1450:4002:800::100e
18:01:45.264194 IP6 ::1 > 2a00:1450:4002:800::100e: ICMP6, echo request,
```

- **Unfixable** without reboot. Sent to netdev again...still not fixed ;)

Specialized triggering with THC-IPv6 2.1 → flood\_router26 -A

## ND Attacks Mitigation Techniques

- **RAguard** (L2 Protection) - RFC 6105
  - Makes switching devices capable of identifying invalid RAs within packets and blocking them
  - Stateless & Stateful modes
- **SeND** - RFC 3971
  - Crypto approach to Secure ND
  - (Very) Hard to deploy - Needs PKI Infra + port security
- ~~“Smart switches doing ND snooping”~~
  - Discard rogue packets based on existing entries



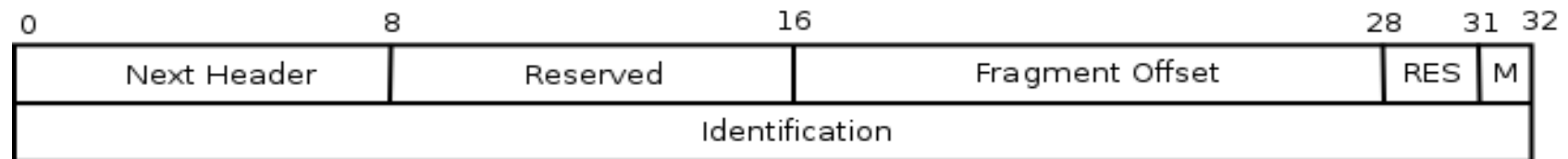
## Other Mitigation Techniques

- Firewall/ACL to block **specific** rogue ICMPv6 - RFC 4890
- DHCPv6 filtering/ACLs (UDP port 546/547)
- Monitor ND with NDPMon
- Monitor local attacks with **6Guard**
- **Disable SLAAC** (esp. in server environments)



**IP Fragmentation** in IPv6 only happens at hosts!  
**PMTU is mandatory!**

Fragmentation Header (Type 44)

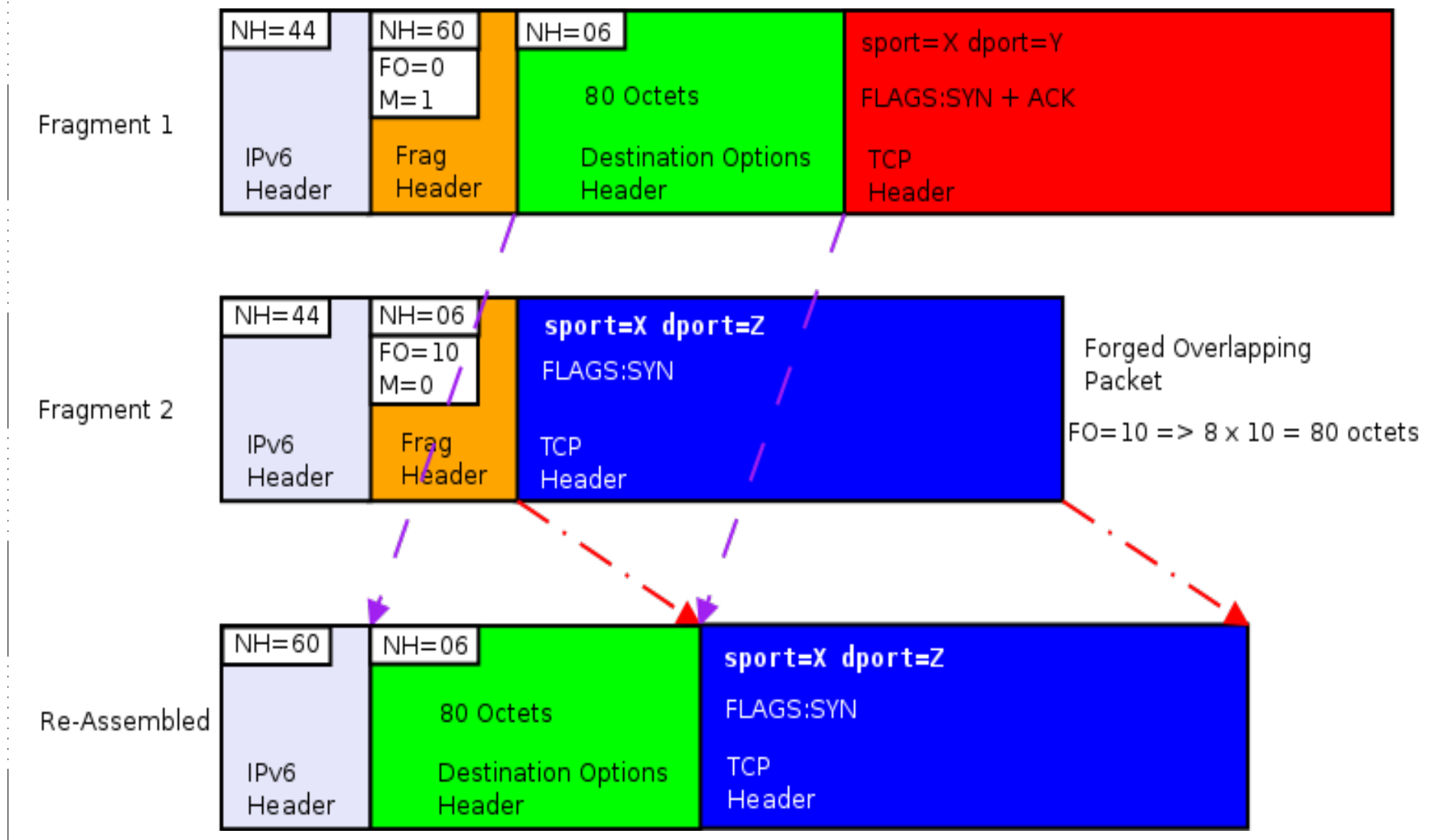


**Fragment Offset** is integer (x8 octets in size)  
**M** is boolean → More Fragments will follow  
**Identification** is a *unique* ID per session

## Major issues

- Overlapping Fragments (RFC 5722)
- Atomic Fragments + Predictable Fragment Identification values

## (Over-Simplified) Overlapping Fragments example





- Many OSs use(d) to “merge” overlapping fragments(\*)  
Any **recent OS** should completely **drop sessions** (not just packets) with overlapping fragments (Linux semi-patched at 11/2010)
- If you can **predict** fragment Identification values, you can send fake packets with overlap. fragments → **DoS** (Linux patched at 07/2011)
- If Frag.Id values → predictable then **Atomic** fragments, single packets with a fragm. Header (offset=0 & M=1), can cause **DoS** to **non-fragmented** traffic

(\*) “Attacking IPv6 Implementation Using Fragmentation” BH-EU-12 A.Atlasis

- (some) RAGuard implementations can be evaded using fragments by adding loooong or multiple DOH (\*)

- Scan a host using fragmented packets

(some Router) ACLs filter by port and allow fragments, else they have to reassembly (ipv6 virtual-reassembly), so you can send a fragmented packet and add multiple ext. headers: eg FRAG + DOH + TCP SYN

src → (FRAG + DOH) + (FRAG + DOH) + ... + (FRAG + TCP SYN)

dst → Can't reassemble, it's a fragment anyway, let it in!

Example tool: **synfrag** <https://github.com/yahoo/synfrag>

- It **is** possible to open/keep a session using frag. packets

(\*) More on this issue at:

<http://blog.si6networks.com/2012/02/ipv6-nids-evasion-and-improvements-in.html>

- RFCs are vague on how an OS should respond to “weird” IPv6 packets → Every OS behaves differently → perform (coarse) Fingerprinting
- Predictable fragment values → **Idle Scanning** (*Hello the 90s called!*)
- Some IDSs can be evaded using multiple Frag + DOH packets (\*)  
9\*(FRAG+DOH) + (FRAG+ICMPv6/TCP/UDP) **can** evade Snort

(\*) “Security Impacts of Abusing IPv6 Extension Headers” BH-AD-12 A.Atlasis

## DNSSEC + EDNS0 + IPv6 fragmentation

- DNSSEC answers can be as long as 4-5kb + EDNS0 supports large UDP replies → the server needs to fragment the UDP reply on IPv6
- Intermediate firewalls/routers can't tell that the fragments ( $> 1^{\text{st}}$ ) are part of a DNS reply, can't find udp header → can't match on "udp.sport==53" → Drop the packets → :( client
- Bad (?) solution, set EDNS0 size to  $< 1400$ bytes → use tcp segments instead of fragments





## Do you remember TCP evil bit ? (rfc3514)

- Original idea was definitely “IPv6 Hop-by-Hop Router Alert Option” rfc2711

The described solution is to define a new IPv6 Hop-by-Hop Header option having the semantic "routers should examine this datagram more closely" and require protocols such as RSVP to use this option.

- Wonder what happens to routers when you flood them with packets containing H-b-H EH triggering “Router Alert” option and random data as “value”...



## Server LANs (!SLAAC)

- Rely on DNS for prefix extraction
- Start with first 64bits of prefix + ::1 (or ffff::1)
- Try 1-255 as last 16bits of address (or 1-ff)
- Try some common last 16bits of addresses:
  - 100,1000,1111,2000,666,etc
  - f00d,cafe,dead,beaf,aaaa,ffff,b0ff,b00b,b00c
  - e.g. FB → 2620:0:1cfe:face:b00c::3
- Common router addresses → ::1 or ::2
- Tools: dnsdict6 + alive6

Recommended Presentation: "Recent Advances In IPv6 Insecurity" by Marc "Van Hauser" Heuse 27C3

## Home/SMB LANs (SLAAC)

- Use Vendor IDs (used in MAC → EUI-64 transformation)
- Example: scan for vuln HP printers when you know prefix  
128-(64+24+16)=24 →  $2^{24}$  possible targets → like scanning /8 in IPv4 (certainly not impossible!)

*2001:db8:foo:b00b:XXXX:XXFF:FEYY:YYYY*

## Indirect Methods

- Email Headers/web logs → watch for 'ff:fe' of SLAAC
- Client side “attacks”: email to clients → payload on IPv6 only host, parse logs

*Temp. Addresses make it really hard to remotely scan Home/SMB LANs (but currently add huge administrative/monitoring cost)*

## Rev DNS Trick - efficiently mapping ip6.arpa

- Reduce queries (from gazzillions to thousands) to find hosts within a /32 (→ that's  $2^{96}$  IP addresses)
- Can find hosts within a /32 in minutes-hours
- Go one nibble at a time “backwards” according to responses (does not work with PowerDNS, works with Bind!)
  - Add 0 in front of x.x....x.x.ip6.arpa
  - Do PTR for 0.x.x....x.x.ip6.arpa
  - If response is NXDOMAIN (→ nothing here)
    - Add +1 to that nibble and do a PTR query again
  - If response is NOERROR → a record follows → continue with this nibble and add 0 in front of it

## Example: finding host(s) within 2001:1af8::/32

6.1.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa PTR void.gr

- I. 0.8.f.a.1.1.0.0.2.ip6.arpa → NXDOMAIN → add +1 ... **4**.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- II. 0.4.8.f.a.1.1.0.0.2.ip6.arpa → NXDOMAIN → add +1 ... **1**.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- III. **0**.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- IV. **0**.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- V. 0.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NXDOMAIN → add +1 ... **a**.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- VI. **0**.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- VII. ....
- VIII. ....
- IX. 0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NXDOMAIN ...  
**1**.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR
- X. 0.1.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NXDOMAIN ...  
**6**.1.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.c.2.0.a.0.0.1.4.8.f.a.1.1.0.0.2.ip6.arpa → NOERROR

## **Rev DNS Trick** - efficiently mapping ip6.arpa

### Tools of the trade

- thc-ipv6: dnsrevenueum6 (included since v2.0)
- ip6-arpa-scan.py
- dns-ip6-arpa-scan.nse



## Reveal LL addresses

- **ping6 ff02::1%eth0** - All-Nodes Multicast Address
  - Does not work on Windows Vista/7
- **ping6 ff02::2%eth0** - All-Routers Multicast Address
- thc-ipv6 toolkit:
  - **alive6** (-l): ping + ping with error header (Hop-by-Hop opt) works even on Windows Vista/7 (#wtf)
  - **fake\_mld6**: mld discovery (listen for joins)

## Get Global Addresses from these

- Sniff RAs for advertised prefix & combine LL addresses.

## Discover Hosts on a LAN using IPv6 tricks

### Rogue RA (control hosts)

- Send RA (lifetime=1) with new prefix & listen for solicitations  
Tool: nmap targets-ipv6-multicast-slaac NSE

### Information Leakage from hosts

- mDNS
- scapy + ICMPv6 139/140 - Node Information Query/Response
- dig any void.gr @ff02::1%eth0 (use tcpdump/wireshark and look at `_all_` the replies)



## **IDS support is still very immature**

Bad signs include among others:

- Poor logging of IPv6 Addresses (LL!)
- Even poorer actions against attackers
- Few specific IPv6 Rules (apart from dedicated tools)
- Fragmentation attacks
- SLAAC enabled on networking interfaces (!?)

## **Firewalling**

- Use dual stack enabled tools (ferm on Linux)
- Don't forget LL, ULA addresses

## **Don't Forget!**

Every Major OS has been IPv6 enabled for years now

But most of them haven't been extensively tested ;)

## **Block transition techniques not in use**

- Deny IPv4 protocol 41 forwarding unless that is exactly what is intended (also block 192.88.99.0/24 → 6to4 tunnels)
- Deny UDP 3544 forwarding → Teredo tunneling
- Deny TCP/UDP 3653 → TSP (tunnel setup protocol)

## **Avoid Dynamic Tunnels** (6to4, Teredo, TSP, etc)

- Turn them off if unneeded (Windows 7)
- **Bitch** at your ISP for native IPv6 connectivity



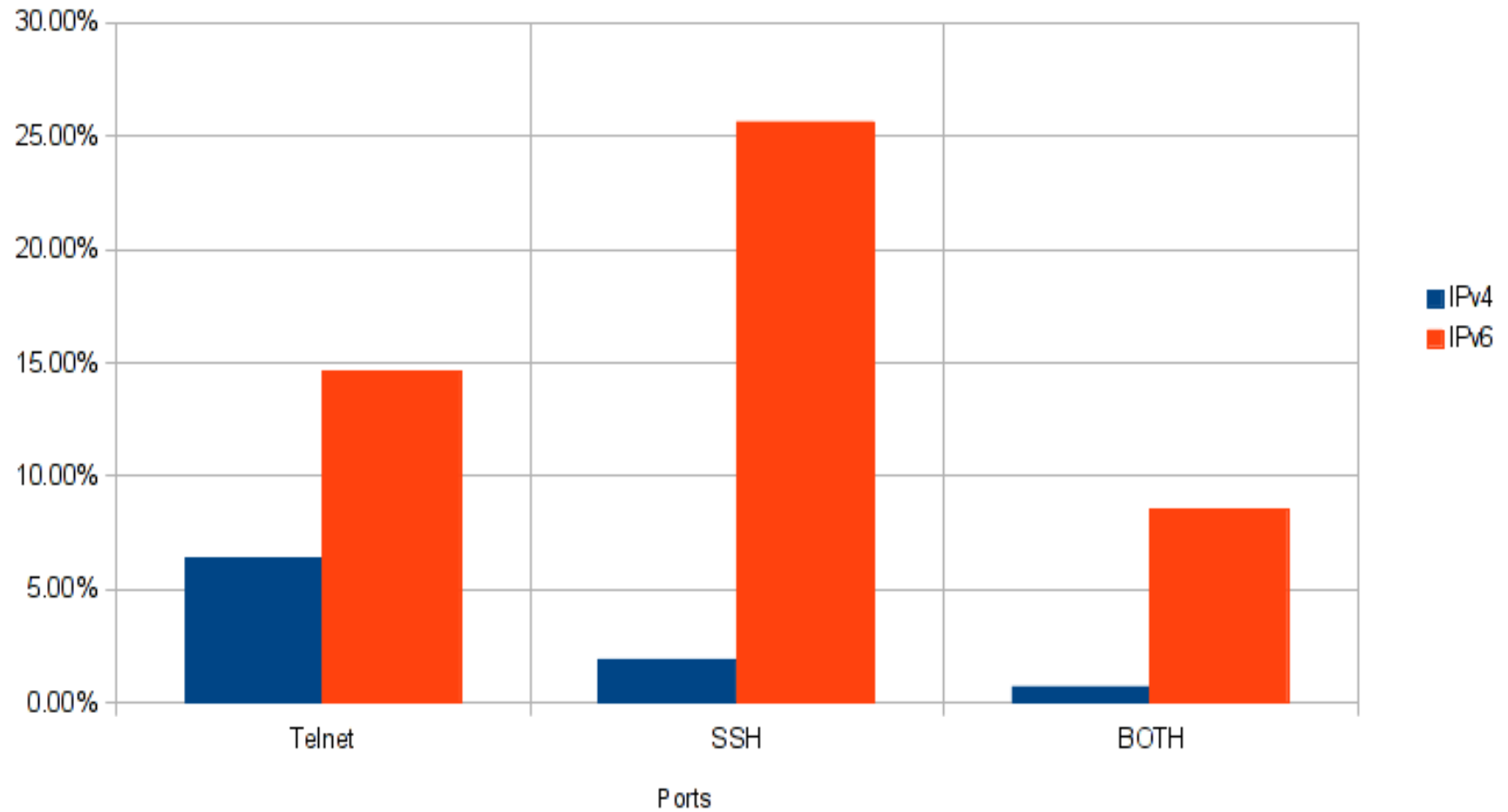
## Comparing IPv4 and IPv6 security policies

- Wanted to check status of firewall policies
- Got a few thousand hosts from apache logs on dual stack hosts
- If IPv6-IP { rev DNS → hostname → A query → IPv4-IP }
- Traceroute each IP(v4/v6)
- Connect to SSH/Telnet of every unique intermediate router of the path to IP(v4/v6)
- Log && parse results
- **Surprise!**



## IPv4 vs IPv6 ACLs

>2000 Routers/Hosts per protocol sample



Major Internet Carriers/ISPs have either telnet or ssh access over IPv6 unprotected

- Same Router/Host (almost) always has IPv4 ACLs
- If ssh/telnet are open...what about other ports ?
- Only one (Greek) ISP noticed (?) the scans and terminated access over IPv6

```

!!!!!!!!!!!!!!
          [REDACTED]
!!!!!!!!!!!!!!

[REDACTED] Network
UNAUTHORISED ACCESS IS PROHIBITED
  !!! ALL ACCESS IS LOGGED !!!
For comments or BUG reports please notify noc@[REDACTED]

[REDACTED] (tty0)

login:
telnet> quit
```



- THC-IPv6
- scapy
- ndisc6
- tcpdump/wireshark (ORLY?)
- nmap (-6) + NSE scripts
- nc6/socat
- 6tunnel
- Ndpmon/6guard
- isic

## Things you can play with

- Data exfiltration using multiple Ext. Headers (DOH)
- OS fingerprinting according to malformed Ext. Header replies
- Ext. header fuzzing against OSs
- ND flooding/fuzzing
- Crashing routers with malformed packets
- Play with IDS/Firewalls + Fragmentation
- IPv6 over 3G/Mobile Devices
- Mobile IPv6 ?

## **IPv6 is no more or less secure than IPv4**

- Experience is the issue - currently there aren't enough experienced engineers, more training needed
- Fewer tools in the wild
- ...but also fewer exploits
- Many topics are now being standardized/changed
- Many current implementations don't follow standards

## **IPv6 is ATM a playground for creative Hackers ☺**

IPv6 has problems that IPv4 “solved” in the 90s...

IPv6 will change traffic patterns (p2p, MIPv6)

IPv6 larger address spaces makes worms and scanning less effective but there are still ways to find hosts (be creative!)

Apply IPsec wherever possible (like you did on IPv4...)

LAN based attacks will become **far** more popular

- Disable SLAAC on IPv4-only networks or when unneeded
- Apply stronger Physical Security, Ethernet-Port Security, NAC, 802.1X, SeND

## Interesting Links

- <http://lists.si6networks.com/listinfo/ipv6hackers/>
- <https://www.ietf.org/mailman/listinfo/v6ops>
- <http://www.packetlevel.ch/html/scapy/scapyipv6.html>
- <https://www.void.gr/kargig/ipv6/>
- <http://ipvsix.me/>
- <http://www.stindustries.net/ipv6-security/>

## Interesting People

- Marc Hause → <http://mh-sec.de/>
- Fernando Gont → <http://www.si6networks.com/>



## Thanks fly to

- **Census Labs**
- **@faidonl**
- **@apoikos**

For the endless discussions regarding IPv6 security

**Go and deploy secure IPv6 Services!**

**Thanks for listening!**

(No Angelina Jolies were harmed during the making of this presentation)

**Any Questions ?**

**For any questions or consulting regarding  
IPv6, you can contact me at**

**kargig [at] void [dot] gr**

**pgp 0xE4F4FFE6**

**<https://twitter.com/kargig>**

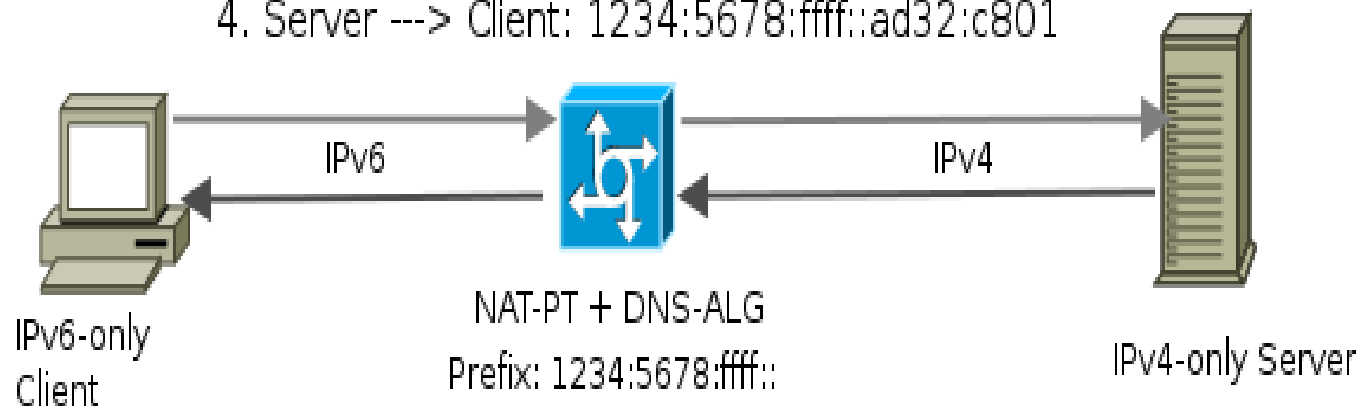
## **Bonus Slides**

GOAL	IPv4	IPv6
Simple Gateway between Internet and Private Network	DHCP	DHCPv6-PD + SLAAC
Simple Security	Filtering side-effect due to lack of translation state	ACL/Firewall
Local Usage Tracking	NAT State Table	Address uniqueness
End-System Privacy	NAT transforms device ID bit in the address	Privacy Extensions
Topology Hiding	NAT transforms subnet bits in the address	Untraceable addresses (IGP host routes/MIPv6 Tunnels)
Addressing Autonomy	Private Address Space	Large Address Space + ULA
Global Address Pool Reservation	Private Address Space	<b>WHAT ?</b>
Renumbering/Multihoming	Address translation at border	Lifetime per prefix / Multiple addresses per interface

## NAT-PT (Network Address Translation/Protocol Translation)

- Allows IPv6 hosts to connect to IPv4 hosts through DNS-ALG and NAT manipulation (**deprecated!**)
- Advertise a /96 IPv6 prefix to LAN
- Translates DNS requests and connections

1. Client ---> Server: AAAA host.com
2. NAT-PT/DNS-ALG ---> Public DNS: A host.com
3. Public DNS ---> NAT-PT/DNS-ALG: 173.50.200.1
4. Server ---> Client: 1234:5678:ffff::ad32:c801



## NAT-PT Router inside an IPv4-only network

- Windows Defaults: SLAAC on + DHCPv6 on
- NAT-PT/DNS-ALG Daemon → DNS/traffic manipulation
- SLAAC + OtherConfig Flag "O" → Clients get IPv6 prefix served by NAT-PT router + start DHCPv6 Client
- DHCPv6 Client → DNS server pointing to NAT-PT router
- Add them to the mix → Clients can be deceived to use IPv6 to talk to IPv4 Internet hosts

### **But Luckily...**

- NAT-PT daemons don't work that well → NAT64 (?)
- Not clean → Can Break Dual-Stack and v6-only sites