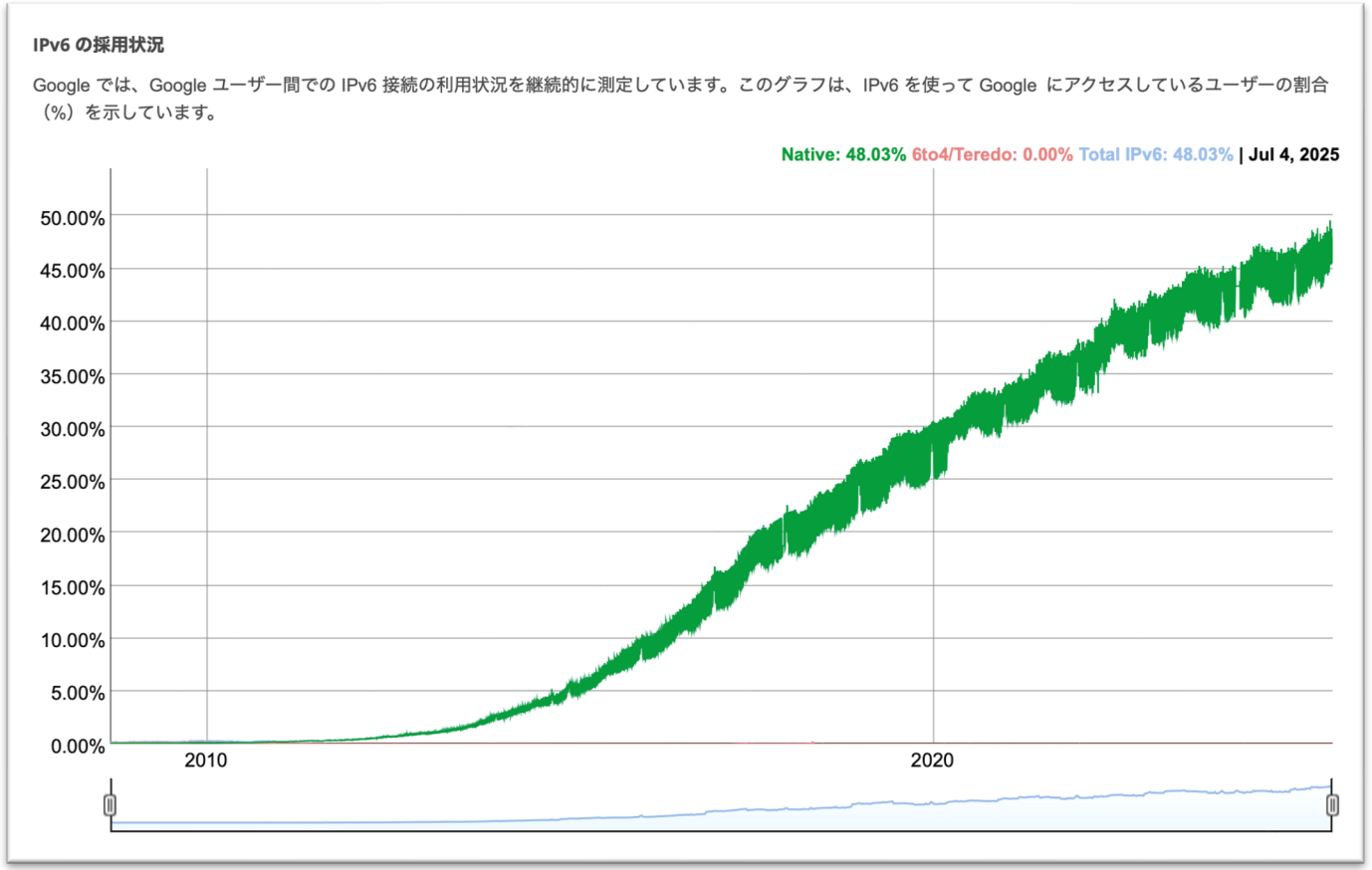
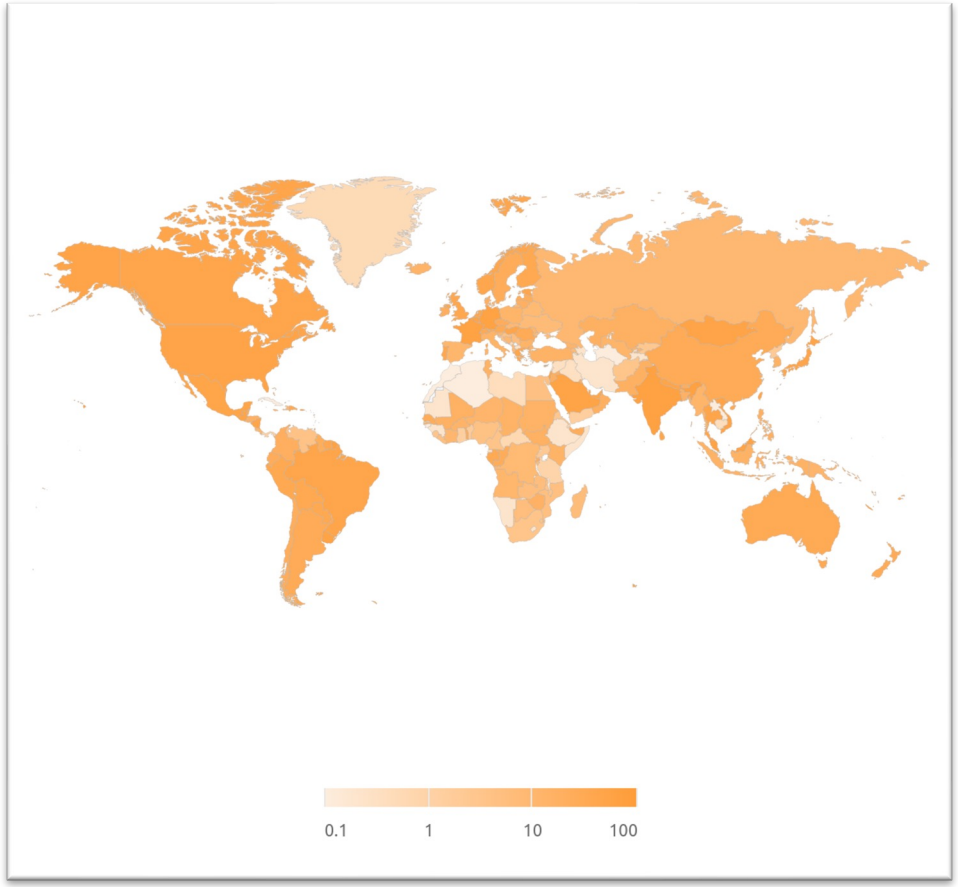
A Day in the Life of IPv6 Scanning

Matsuzaki ‘maz’ Yoshinobu
<maz@iij.ad.jp>

IPv6 adoption



https://www.google.com/intl/ja/ipv6/statistics.html



https://www.akamai.com/ja/security-research/ipv6-adoption-visualization

IPv6 Address

- 2001:0db0:0000:0000:0000:0000:0000:0000
  - 128bit long
- 2001:db8:cafe:beef::/64
  - Each segment is generally configured as /64
  - Terminals use the remaining 64 bits as Interface ID (IID) to configure their IP address

- Example:
  - IPv4: 192.168.0.0/24 → 192.168.0.0 – 192.168.0.255
  - IPv6: 2001:db8:cafe:beef::/64
    → 2001:db8:cafe:beef:0000:0000:0000:0000
    - 2001:db8:cafe:beef:ffff:ffff:ffff:ffff

IPv6 Address Auto-Configuration

- DHCPv6 (DHCP for IPv6)
  - DHCPv6 Server assigns IP address based on request from clients
- SLAAC (Stateless Address Autoconfiguration)
  - Prefix information is advertised via IPv6 RA on-link
  - Terminal generates remaining 64 bits (IID) by itself to construct IPv6 address
  - Recommended IID methods are Semantically Opaque (RFC7217) and Temporary (RFC8981)
- RFC7217: Generates stable, random IID per subnet
- RFC8981: Generates temporary, random IID

Various IID Generation Methods

Method	Tolerance for Tracing	Tolerance for Scanning
• RFC2464 Stable, EUI-64 base	weak	weak
• RFC4941 Stable, random	weak	strong
• RFC3972 Crypto base	strong	strong
• RFC7217 Stable, random	strong	strong
• RFC8981 Temporary, random	strong	strong

- EUI-64 based IID may reveal product-specific scannings

Too broad to scan all

- Need to find IPv6 prefixes (64bit) in use
  - from 18 quintillion possibilities
- Need to find IIDs (64bit) in use
  - from 18 quintillion possibilities

- Helpful hints
  - DNS / TLS certificates -> IP address from the hostname
  - Public measurements like RIPE Atlas probes

TGA (Target Generation Algorithm)

- Estimates other IPv6 hosts from hint information
- Generates candidate target list for scanning
- Various studies and proposals exist
  - 6Tree, DET, 6GAN, 6Forest, 6Scan

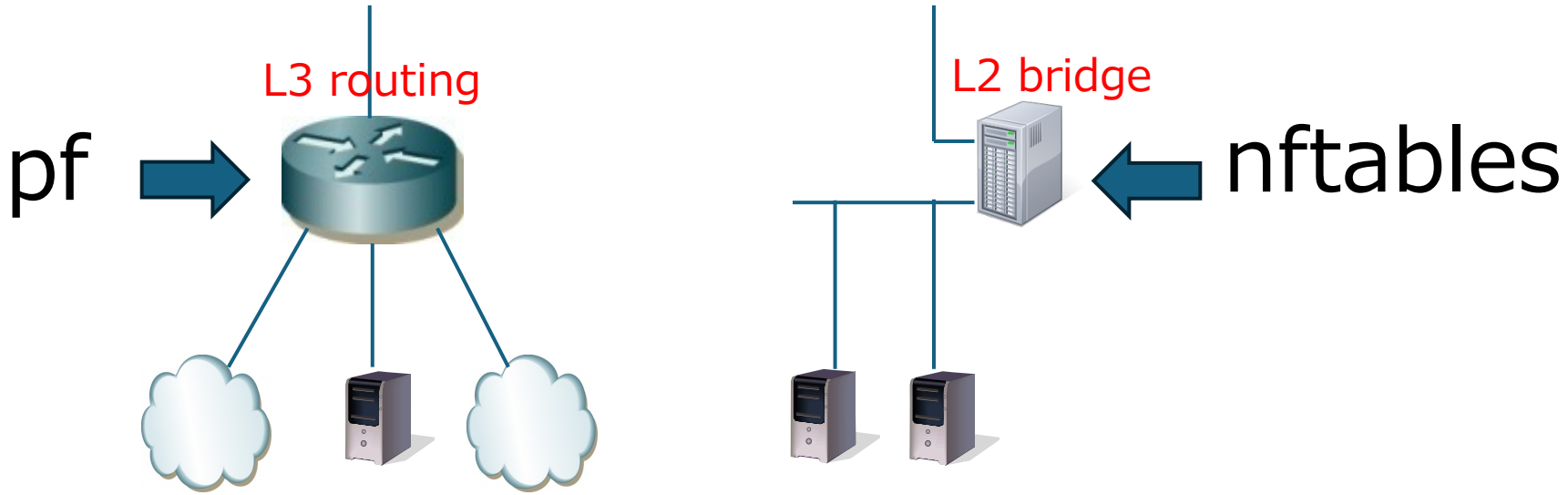
- **IPv6 Hitlist** seems the prominent hint
  - https://ipv6hitlist.github.io/
  - Maintains lists of reachable IPv6 hosts
  - Currently listing about 3.6 billion addresses

Well, let’s see what’s coming in

- pf
  - Solid packet filter for OpenBSD, and ported to other systems
  - Supports stateful tracking for tcp, udp, icmp, etc.
  - Readable rules using quick syntax, tables, macros
  - pflog provides tcpdump-friendly logging
- nftables
  - Linux packet filter framework
  - Supports IPv4/IPv6/arp/bridge
  - Rule * Chain
  - Counter and Trace capabilities

My configuration setup

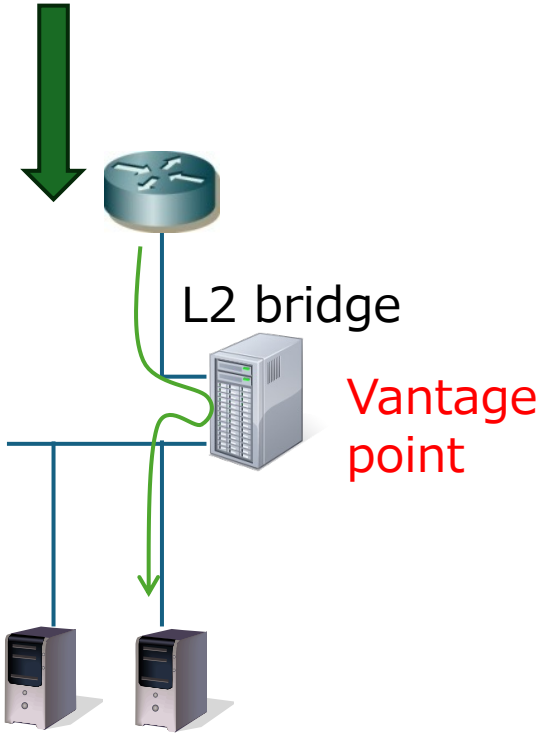
pf for L3 segments; nftables for L2 end segments



This L2 setup can’t detect packets sent to unused addresses
Need a device inside that responds to any NDP NS for detection

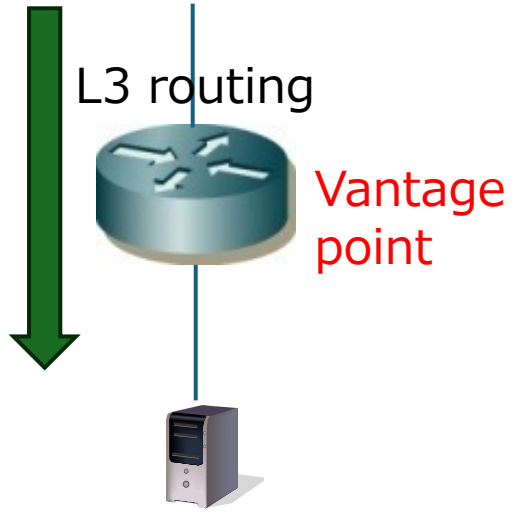
Visible and invisible aspects – L2

- Packets to existing hosts are visible
  - Once a packet reaches the upstream router, it triggers NDP (Neighbor Solicitation), and if MAC address is resolved, the packet is sent
- Packets to addresses not responding to NDP are dropped by the router
  - By recording NDP, we can observe which destination packets were headed to



Visible and invisible aspects – L3

- All incoming packets can be observed
  - Depending on routing
- Even packets to unused networks can be observed if routed



nftables.conf example

```
#!/usr/sbin/nft -f

flush ruleset
define EXT = ens18
define INT = ens19
table bridge filter {
        chain forward {
                type filter hook forward priority filter; policy drop;
                ether type arp drop;
                ether type ip drop;
                ct state established,related accept;
                icmpv6 type { nd-neighbor-solicit, nd-neighbor-advert, nd-router-advert} accept;
                iif { $INT } oif { $EXT } ct state new accept;
                iif { $EXT } oif { $INT } ct state new tcp dport { 22 } accept;
                log;
        }
}
```

pf.conf example

```
if="vio0"
icmp6_allows="{ 128, 133, 134, 135 }"
set skip on lo
antispoof quick for { lo }
scrub in all
pass in quick on $if inet6 proto ipv6-icmp icmp6-type toobig keep state
pass in quick on $if inet6 proto ipv6-icmp from any to ($if) icmp6-type $icmp6_allows keep state
pass in quick on $if inet6 from any to ff02::/16 keep state
pass out quick on $if proto tcp all modulate state flags S/SA
pass out quick on $if proto { udp, icmp, ipv6-icmp } all keep state
block in log on $if
```

Relation to IPv6 Hitlist

- L2 observation (/64 with 1 host listed)
  - 4 packet destinations (0 listed)
  - 1 Atlas probe, 2 TLS cert hosts, 1 new Ubuntu host
- L3 observation (/56 with 9 hosts listed)
  - 500 packet destinations (8 listed)
  - 1 Atlas probe, 2 TLS certs, various others

- To be scanned, must respond to probe packets beforehand

Port trends seen via nftables (L2)

- TCP 94% -- Mostly HTTPS/443, HTTP/80
  8080,8443,8081,8888 and such
- UDP 4% -- SNMP/161, DNS/53, ISAKMP/500, NTP/123
  STUN/3478 and etc.
- Others 2% -- Tunnels like IP-ENCAP, GRE, IPv6

- ICMPv6 excluded here due to anomaly (124x more than others, explained next page)
- 325packet/host/day

Frequent ICMPv6 Echo Requests

- Source: Huawei-Cloud-SG
- Sends to specific destinations every 2s
  - Atlas probe, new Ubuntu host
- Same ICMPv6 ID
- ICMPv6 SEQUENCE increments

-> likely left a ping6 running

Port trends seen via pf (L3)

- TCP 88% -- Many on 8080,8081,8001,8888,8090
  Also 80,443,21,1723,3306,3389,445,,,
- UDP 6% -- Many on SNMP/161, DNS/53, ISAKMP/500,
  NTP/123, STUN/3478. And other ports
- ICMPv6 3% -- Echo Request, Toobig
- Others 3% -- Tunnels like IP-ENCAP, GRE, IPv6

- 127packet/host/day

Types of Destination IID Patterns

- EUI-64 base -- 366 destinations
- Lower 24 bits patterns -- 101 destinations
  - ::a ::[0-9a-f]
  - ::a:2 ::[0-9a-f]:2
  - ::a0:2 ::[0-9a-f]0:2
  - ::a2 ::[0-9a-f]2
  - ::a02 ::[0-9a-f]02
  - ::a002 ::[0-9a-f]002
- Random-looking lower 64 bits -- 33 destinations
  - 3931:828f:a5f8:789f

Destination Types per Prefix

- Prefixes with no reachability history receive almost no packets
  - Not listed in Hitlist
  - **Traditional IPv4 darknet method does not work**
- In case of using sequential IID, others do not get scanned (for now)
  - I use 100,101,102 for IID, but no probes to 103,104
- When EUI-64 based IID is detected, similar devices are scanned based on MAC address assignments
  - About 60% of scans follow this
  - Also observed modified versions of known IIDs

Interesting Incoming Packets

- ICMPv6 Packet Too Big
  - ICMPv6 Error message for non-communicating packets
  - Contains original packet as UDP with src/dst port 22
  - New MTU: 1300
- Tunnel endpoint probing
  - Payload is ICMPv6 echo reply from the destination (target) host
  - Simple tunnel protocols like GRE, IPv6
- Some distinctions like Flowlabel
  - Lack of Flowlabel
  - Judging by this alone is risky though

Distribution of the Scanning Source

- Often from measurement-related networks
  - Entities also scanning over IPv4
  - Universities, research institutions
  - Cloud providers
- Source IP usage
  - A few hosts
  - A few /64s for spreading traffic
    - Some load-sharing implementations?

IPv6 Hitlist update

- Hosts that respond are likely added to the hitlist
  - The specific probing methods used are not yet identified
- Entries are removed after a period of no response to probing packets
  - Removal typically takes about two weeks
- Some entries for non-existent hosts may persist longer
  - If the entire /64 prefix becomes unresponsive, lingering entries are also removed

Getting Noticed by IPv6 Scanners

- Make the /64 prefix appear in use
  - Being listed in the **IPv6 Hitlist** is a key objective
  - Some scanners may maintain their own reachability lists
  - Issue **TLS certificates** using hostnames
  - Publish **AAAA records** in DNS to be discovered
    - Simply adding an AAAA/NS record is not enough
    - Consider placing **URI links** in locations where they can be crawled
- Preferably use **EUI-64-based IIDs**
  - Existing research and knowledge help guide scanning efforts
  - Nearby address space is likely to be explored proactively

How to Avoid IPv6 Scanning

- Avoid listing sensitive hosts in public reachability lists
  - Segment prefixes by purpose
  - Place scan-exposed systems in dedicated segments
    - Including ATLAS probes
- Do not respond to unsolicited probing from external
  - Silence helps trigger automatic delisting
- Aim to be removed from the IPv6 Hitlist
  - Filtering probe packets typically causes delisting in ~2 weeks
  - Some scanners persist beyond normal thresholds
- Prevent host identification via TLS certificates
  - Use self-signed or wildcard certificates when feasible

Summary

- IPv6 Scanning is already happening in the wild
  - Various IPv6 scanning techniques are actively being researched
  - Often by academic and commercial entities
- Reachable IPv6 host lists are maintained
  - e.g., the widely used IPv6 Hitlist
  - Appearing in such lists may trigger scanning of neighboring addresses
- Multiple services and ports are being scanned
  - Not just HTTP or SSH - scanning is broad
  - Access controls are your friends
- Traditional IPv4 darknet methods are ineffective for IPv6
  - Prefixes with no reachability history receive almost no packets