Network 1\textsuperscript{st} Hop Security

Mitigating the security risks of ARP, DHCP and IPv6 Autoconfiguration

Klaus Möller

WP8-T1

Webinar, 11\textsuperscript{th} of August 2020

Public

www.geant.org
The First Hop

• The way from the end-system (PC, Laptop, Server, Tablet, etc.) to the default router
• Aka the local (W)LAN segment
  – One collision domain for all systems on the local net: Hubs, shared coaxial cabling (very old), access point (WLAN)
  – One collision domain per end-system: VLAN with one or more switches
• Additionally: Locally active servers for network infrastructure
  – DHCP
  – Optionally: TFTP, DNS, others
Local network: Attack Surface

- Detection of other hosts on the subnet: ARP, IPv6 NDP
  - Obtaining the MAC address for a given IP address to communicate locally
  - Without the MAC address of the default gateway, no communication beyond local network

- Automatic configuration of IP addresses: DHCP, IPv6 SLAAC
  - This usually includes the IP address of the default gateway (router)

- Other end-system configuration: DHCP (IPv4 & IPv6)
  - DNS server, NTP server, (Windows) Domain Controllers

- Not covered
  - Directly accessible services on switches, access points or routers (SSH, Web, etc.)
  - Other servers on the local network
ARP Basics

• **ARP: Address Resolution Protocol (RFC 826)**
  - Host wants to find the link-layer (MAC) address for a (destination) IP-address

• **How:**
  - Host broadcasts (MAC address ff:ff:ff:ff:ff) ARP request
  - If a host with this IP address is on the local link, it responds with its IP address in an Ethernet frame (unicast to the querying host)
  - Learned address pairs (IP, MAC) are stored locally in the ARP cache
  - Cache will be updated when a host receives ARP responses, even if already present
  - Hosts may send unsolicited ARP responses (i.e. in case of address changes)

• **ARP can be used for duplicate address detection (RFC 5227)**
  - Requests with empty source address and (tentative) IP address
ARP Attacks

• ARP Spoofing: Sending of fake ARP responses
• ARP (Cache) Poisoning: Planting false entries into a victim's ARP cache through ARP Spoofing
• Allows Man-in-the-Middle (MitM) attacks: reading or altering (unencrypted) traffic through impersonating the legitimate communication partner
  - Usually the default gateway, proxy, etc.
  - Impersonating DNS server or DHCP server allows further MitM attacks
ARP Attack Detection

- Watch for deviating entries in the ARP cache
- Deviating: IP-MAC address pair is not what it is supposed to be
  - Esp. default gateway, DNS/DHCP servers, proxies, etc.
- CLI: `arp -an` (old) or `ip neighbor show` (newer)
- Better: Monitoring tools: addrwatch, ArpON, arpwatch
  - One host per LAN segment is enough due to ARP ethernet broadcasts
  - The segment may contain several IP subnets
  - Alerting through Syslog or E-mail
- Monitoring switch table of (MAC, port, VLAN) mappings (CAM)
  - Gives also a hint to where the attack might come from (switch ports)
ARP Attack Mitigation

• Static ARP cache entries
  – I.e. manually configuring IP-MAC pairs on each host
  – Impractical on larger networks

• Locking down MAC addresses on switch ports
  – Works against ARP spoofing only when triplet (MAC, IP, port) is monitored/protected
  – Manual configuration or auto-learning through DHCP traffic monitoring
  – Reconfiguration required when systems move, HW changes, etc.

• Encrypting/authenticating traffic end-to-end with certificates
  – Alerts/aborts communication when certificates do not match
  – How do users know a certificate is faked?
DHCP: Basics

- **DHCP:** Dynamic Host Configuration Protocol (RFC 2131)
- How it works: “DORA”
  - Client sends (DHCP) Discover Message
  - Server answers with (DHCP) Offer Message
  - Client chooses from the offers, sends (DHCP) Request message
  - Server sends (DHCP) Acknowledgement message with configuration parameters
    - IP address, MTU, TTL, DNS server, NTP server, Domain controller, etc.
- Implemented on top of (older) BootP protocol
  - Server: Port 67/udp
  - Client: Port 68/udp
DHCP: Attacks

- Rogue DHCP client
  - Attacker client assigns all IP addresses to himself → Denial-of-Service by address pool exhaustion
  - Not such a big problem with IPv6 – really?

- Rogue DHCP server
  - Attacker masquerades as the (legitimate) DHCP server
  - Can send his own configuration parameters to clients
  - Means for further attacks: MitM, eavesdropping, modifying traffic

- DHCP Spoofing
  - Attacker sends forged DHCP responses
  - Consequences similar to rogue DHCP server
DHCP Attack Detection

- Monitor DHCP messages (DHCP snooping)
  - Unusually large number of Discover or Request packets
    ⇒ perhaps a misconfigured or rogue DHCP client
  - DHCP Offer/Acknowledgements not coming from legitimate (IP, MAC) address pairs
    ⇒ sign of rogue DHCP servers or DHCP spoofing
- Monitor DHCP server logs
  - Handing out unusually large number of leases
  - Requests for strange options (those not usually asked for by clients)
DHCP Attack Mitigation

- Filtering of DHCP Offer/Acknowledgements (DHCP Guard)
  - Those that come from switch ports other than that of the legitimate server
  - The other part of DHCP Snooping
  - Filter DHCP at the network boundary firewall
  - If not using DHCP relays, filter at the subnet boundary

- DHCP authentication (RFC 3118)
  - Servers with support available (Cisco, Juniper, ISC, ...)
  - Client support not yet there?
  - **All** clients on the (local) net have to support it
DHCPv6

- Different protocol (RFC 8415)
  - Now implemented directly on UDP(v6)
  - Client port: 546/udp
  - Server port: 547/udp

- Operational principle is the same as in DHCP on IPv4
  - DISCOVER, OFFER, REQUEST, ACKNOWLEDGE messages

- Security software has to work with different protocol and port numbers
IPv6

- Yes, you have it on your local (sub)network - although it might not work beyond the first hop
- It’s build-in and enabled by default on all operating systems
  - Linux, *BSD, Windows, MacOSX, etc. since 15 years at least!

```bash
> ip addr show dev wlan0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether dc:8b:28:5b:79:a3 brd ff:ff:ff:ff:ff:ff
   inet 192.168.93.99/24 brd 192.168.178.255 scope global noprefixroute dynamic wlan0
      valid_lft 841369sec preferred_lft 841369sec
   inet6 fe80::f742:11d7:4008:159a/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
> ping ff02::1%wlan0
PING ff02::1%wlan0(ff02::1%wlan0) 56 data bytes
64 bytes from fe80::f742:11d7:4008:159a%wlan0: icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from fe80::9ec7:a6ff:fe25:eff1%wlan0: icmp_seq=1 ttl=64 time=2.63 ms (DUP!)
64 bytes from fe80::4240:a7ff:feb6:49fc%wlan0: icmp_seq=1 ttl=255 time=163 ms (DUP!)
```
IPv6 Address configuration

- Three ways to configure IPv6 addresses
  - Static (manually)
  - Stateless Address Autoconfiguration (SLAAC)
  - Stateful (DHCPv6)

- Really just choosing the Interface ID (last 64 bit)

- Prefix (first 64 bit) usually given
  - Statically by network admin, automatically by router/DHCPv6 server

- Problems
  - Differences between and co-existence of IPv4 and IPv6
  - Privacy issues with IPv6 interface IDs
IPv6 Stateless Address Auto Configuration (SLAAC)

- Process, by which IPv6 hosts obtain
  - (Global) IPv6 Prefix
  - Interface ID
  - Router (default gateway) IP address
  - DNS server IP address
  - Check if their (choosen) IP address is not already in use on the subnet

- IPv6 Neighbor Discovery Protocol (RFC 4861)
- Implemented in five ICMPv6 message types
  - Router Solicitation (Type 133)
  - Router Advertisement (Type 134)
  - Neighbor Solicitation (Type 135)
  - Neighbor Advertisement (Type 136)
  - Redirect (Type 137)
IPv6 Neighbor Discovery

• Neighbor Solicitation/Advertisements
  − Host A wants to send data to host B (in the local subnet)
  − A knows B’s IPv6 address but not the link-layer address
  − A sends Neighbor Solicitation packet the multicast MAC address B is registered at (can be deducted from B’s IPv6 address)
  − B answers with Neighbor Advertisement Packet to A’s Unicast MAC address (knows it from the Neighbor Solicitation message)
  − This packet includes B’s MAC-Address
  − A and B communicate via Unicast from here on

• This mechanism replaces (IPv4) ARP!
IPv6 Duplicate Address Detection

- Host A wants to know if the IPv6 address chosen is already in use in the subnet
- A sends Neighbor Solicitation Packet to the multicast address of the chosen IPv6 address (source address is ::)
- If the address is already in use, the using host sends a Neighbor Advertisement packet to the link-local all-nodes multicast address (ff02::1)
- If A receives no answer, the address can be used
IPv6 Router Discovery

- Router periodically sends Router Advertisements (RA)
  - Unsolicited: semi-periodically from router to link-local all-nodes address (ff02::1)
  - Solicited: as answer to Router Solicitation (RS) packet from a host

- Host extracts
  - List of valid (subnet)prefixes for this subnet
  - If a prefix can be used for SLAAC or not
  - List of routers (default gateways)
  - Lifetime of the RA
  - Hop Limit
  - Optional parameters like MTU or DNS servers
IPv6 SLAAC Process

• Host creates a link-local IPv6 address (State: tentative)

• Hosts uses DAD to check if the address is unique
  - If already in use: STOP!
  - Else: IP address enters state valid

• Host sends Router Solicitation

• Host received Router Advertisement(s)
  - If no RA received, continue with DHCPv6
  - If prefix can be used for SLAAC, create global IPv6 address

• Check global address with DAD
  - If unique (no answer), enter state Valid
  - If not unique, continue with next prefix
  - No more prefixes? STOP!
ND Attacks

- ND spoofing
  - For each Neighbor Solicitation packet, send a (fake) Neighbor Advertisement
    - DAD/SLAAC stops → Denial of Service
  - Send out lots of fake Neighbor Advertisements
    - Overflow of Neighbor tables in systems on the local network
RA Attacks

• RA spoofing (aka Rogue IPv6 Router)
  - Attacker sends fake RA packets
    • Removes existing routers from routing table
    • Can assign fake prefixes to hosts on the subnet
    • Makes his/her router the default gateway → MitM attacks
  - Can turn an IPv4 only net into a dual stack network!

• IPv6 is preferred by default over IPv4!
NDP Attack Detection

- Monitoring of IPv6 address – MAC address pairs: addrwatch
  - Like arpwatch, but includes IPv6
- Monitor routing/neighbor tables at local hosts
  - Sudden increase is likely a sign of attack → HIDS
- Snooping of ICMPv6 at the switch
  - Catches both ND and RA attacks and probably more
NDP Attack Mitigations

• Filtering of RAs on the switch ports (Router Guard)
  – Can be bypassed with IPv6 Extension Headers if BCP not followed

• Static configuration of IPv6 addresses (disable RA processing)
  – Ignore ICMPv6 types 133, 134, 137 completely
  – Resolves most RA issues at the price of considerable more work
  – And shuts down DHCPv6 also:(

• Set router preference flag (RFC 4191) to "high" on your routers
  – Only a minor help

• Authentication of ND and RA packets (SEND)
  – Not used/supported
DHCPv6 and SLAAC

- Usage of DHCPv6 is dependant on flags in RAs
  - Managed Configuration Flag (M): Use a *Configuration Protocol* (i.e. DHCP) to obtain an IPv6 address (stateful)
  - Other Stateful Configuration Flag (O): Use a *Configuration Protocol* to get further informations (i.e. DNS servers, etc.)

<table>
<thead>
<tr>
<th>M</th>
<th>O</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Don’t use DHCPv6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Use DHCPv6 only for further information (DHCP Stateless)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Use DHCPv6 only to obtain an IPv6-Adress</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Use DHCPv6 for both IPv6 Adress and further information</td>
</tr>
</tbody>
</table>
## Mitigation: One Size doesn’t fit all

<table>
<thead>
<tr>
<th>Zone</th>
<th>Rogue RA Mitigation Measure</th>
<th>cost (+ o -)</th>
<th>feasibility</th>
<th>effect (+ o -)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Network</td>
<td>Router-Preference=high / Monitor NDP Managed Switch (RA Guard, Port ACLs)</td>
<td>+/-</td>
<td>+</td>
<td>0/+</td>
</tr>
<tr>
<td>Internal Server-Zone</td>
<td>Router-Preference=high / Monitor NDP Disable RA processing</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DMZ</td>
<td>Router-Preference=high / Monitor NDP Disable RA processing</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Guestnet Wired</td>
<td>Router-Preference=high Managed Switch with RA Guard or Port ACLs</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Guestnet Wireless</td>
<td>Router-Preference=high Partitioning</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Original slide(s): https://www.first.org/education/ipv6_security.zip
**IPv6 Local interface ID**

- **Two step process**
  - Derive EUI-64 from 48 Bit MAC address
  - Complement bit 7

**MAC address**
say: 00:25:64:df:5d:c7

**EUI-64 address**
say: 00:25:64:ff:fe:df:5d:c7

**IPv6 Interface ID**
say: 0225:64ff:fedf:5dc7

---

Company ID:
http://standards.ieee.org/regauth/oui/oui.txt
IPv6 Interface ID Privacy Problem

- 64 bit is enough to uniquely identify a system (48-bit actually)
- Tracking possible across different subnets (work, home, cafe, ...)
- Solution: Privacy Enhancements (RFC 4941)
- Randomly chosen Interface ID with regular changes aka *temporary address*
- However
  - Default interval is often too long → allows tracking again
    - Router Advertising and Kernel variables
  - And it’s not changed when the prefix changes
  - An EUI-64 (public) address may be assigned additionally, preference?
    - Controlled by kernel variables on Linux/*BSD
What you have learned?

- Basic Overview of attacks on ARP, IPv6 NDP and DHCP(v6)
- Basic Mitigation measures

What has been left out?

- DNS/mDNS Security: Will be covered in an upcoming course on DNS Security
- LAN protocols: STP, VTP, LLDP, CDP, etc.: Turn it off on links to end-systems
- Router failover protocols: HSRP, VRRP, etc.
Thank you

Any questions?

Next module: *Authentication*, 13\textsuperscript{th} of August 2020

www.geant.org
References

Standards

Tools: Offense

- **ARP**
  - Arp-scan: https://github.com/royhills/arp-scan
  - Ettercap: https://www.ettercap-project.org/
- **DHCP**
  - DHCPig: https://github.com/kamorin/DHCPig
  - Dhcpstarv: http://dhcpstarv.sourceforge.net/
  - Yersinia: https://sourceforge.net/projects/yersinia/ (old homepage, new homepage broken?)
- **IPv6**
  - SI6 Networks’ IPv6 toolkit: https://www.si6networks.com/research/tools/ipv6toolkit/
  - Chiron: https://github.com/aatlasis/Chiron
Tools: Defense

- **ARP**
  - Addrwatch: https://github.com/fln/addrwatch
  - TuxCut: https://a-atalla.github.io/tuxcut/
  - ArpON: http://arpon.sourceforge.net/

- **DHCP**
  - Open DHCP Locate: http://odhcploc.sourceforge.net/
  - Univ. Princeton dhcp_probe: https://www.net.princeton.edu/software/dhcp_probe/
  - Microsoft DHCPOLOC Utility: https://gallery.technet.microsoft.com/DHCPOLOC-Utility-34262d82

- **IPv6**
  - Addrwatch: https://github.com/fln/addrwatch
Backup material

Stuff that didn’t make it due to time constraints

www.geant.org
IPv6 Address configuration

- **Static, like IPv4**
  - Linux: `ifconfig ... or ip addr ...`
    - Persistent configurations vary (/etc/...?)
  - Windows: GUI or `netsh interface ...`

- **Use for**
  - Central servers that better have fixed addresses
  - Administrators responsibility that addresses are unique

- **High Security, if**
  - Neighbor Advertisements are ignored
  - Router Advertisements are ignored
    - What to do when router fails (VRRP?)
**IPv6 Address states**

- **Tentative**: Address is still being checked for uniqueness
- **Valid**: Address can be used to send and receive
  - Includes Preferred and Deprecated states
- **Preferred**: Address is preferred for new connections
- **Deprecated**: Address should not be used for new connections
- **Invalid**: Address can’t be used anymore
How to enable/disable Privacy Extensions

- **Linux**
  - `sysctl -w net.ipv6.conf.all.use_tempaddr=2` (Default=0)
  - `Sysctl -w net.ipv6.conf.all.temp_preferred_lft=14400` (Default 86400)
  - Entries in `/etc/sysctl.conf`
- **FreeBSD/Mac OS X**
  - `sysctl -w net.inet6.ip6.use_tempaddr=1`
  - `sysctl -w net.inet6.ip6.prefer_tempaddr=1`
- **MS Windows**
  - `netsh interface ipv6 set global randomizeidentifiers=disabled`
  - Default: enabled
IPv6 Neighbor Solicitation

IPv6: fe80::2ab:cdff:fe12:3456
MAC: 00:ab:cd:12:34:56

Ethernet Header
- Dest MAC: 33:33:ff:78:90:ab
IPv6 Header
- Src Addr: fe80::2ab:cdff:fe12:3456
- Dst Addr: ff01::1:ff78:90ab
- Hop Limit: 255
Neighbor Solicitation Header
- Trgt Addr: fe80::2ab:cdff:fe78:90ab
Neighbor Discovery Option
- Src LL Addr: 00:ab:cd:12:34:56
IPv6 Neighbor Advertisement

Ethernet Header
- Dest MAC: 00:ab:cd:12:34:56
IPv6 Header
- Src Addr: fe80::2ab:cdff:fe78:90ab
- Dst Addr: fe80::2ab:cdff:fe12:3456
- Hop Limit: 255

Neighbor Advertisement Header
- Target Addr: fe80::2ab:cdff:fe78:90ab

Neighbor Discovery Option
- Target MAC Addr: 00:ab:cd:78:90:ab
IPv6 Router Solicitation

IPv6: fe80:2ab:cdff:fe12:3456
MAC: 00:ab:cd:12:34:56

Ethernet Header
- Dest MAC: 33:33:00:00:00:02
IPv6 Header
- Src Addr: ::
- Dst Addr: ff02::2
- Hop Limit: 255

Router Solicitation Header

IPv6: --
MAC: 00:ab:cd:78:90:ab
IPv6 Router Advertisement

- **IPv6 Router Advertisement**
- **Ethernet Header**
  - Dest MAC: 33:33:ff:78:90:ab
- **IPv6 Header**
  - Src Addr: fe80::2ab:cdff:fe12:3456
  - Dst Addr: ff02::1
  - Hop Limit: 255
- **Router Advertisement Header**
  - Cur Hop Limit, Flags, Router / Reachable / Retrans
- **Neighbor Discovery Option**
  - Src LL Addr: 00:ab:cd:12:34:56
  - MTU: 1500
  - Prefix Information

IPv6: fe80::2ab:cdff:fe12:3456
MAC: 00:ab:cd:12:34:56

IPv6: --
MAC: 00:ab:cd:78:90:ab
IPv6 Secure Neighbor Discovery (SEND, RFC 3971)

- Host has to prove that an IPv6 address really belongs to it
- Host creates RSA key pair
- Used to generate Cryptographically Generated Address (CGA)
  - Modifier: Random numbers

RSA Keys
- Private
- Public

CGA Parameters
- Modifier
- Public Key
- Subnet Prefix

SHA-1

Subnet Prefix
Interface Identifier

Cryptographically Generated Address
IPv6 Secure Neighbor Discovery

- Host adds SEND messages as options to Neighbor Advertisements
- “Proof”
  - The CGA address really belongs to the public key
  - The host has this accompanying private key
- For a successful attack, the attacker has to find (in time) a key pair that generates the same signature
IPv6 SEND Problems

• Lack of support in operating systems:
  – None: Windows, MacOS X, BSD
  – Linux kernel patches, but not in mainline
  – Cisco IOS 12.2, Juniper?

• Susceptible to Denial-of-Service attacks
  – Attacker floods local network with SEND NS packets
  – Victims have to to many cryptographical operations (RSA is slow)

• Layer 2 attacks (MAC address spoofing) can still be done