DYNAMIC HOST CONFIGURATION, PLEASE

AsiaBSDCon 2023

Florian Obser - florian @ openbsd.org

WHO AM I

- OpenBSD developer for 10+ years
  - ~ 2k commits
  - `many files changed, 412k insertions(+), 511k deletions (-)`
  - Priv'sep Network Daemons / ping / traceroute / dig
- Senior Systems Engineer @ RIPE NCC
  - k.root-servers.net
  - Routing Information Service (RIS)

WHAT IS...

- overview of network configuration on OpenBSD laptops
  - Wi-Fi
  - IPv4 & IPv6 auto-configuration
  - Cellular networks
  - DNS resolution
  - IPv6-only networks
- Previous in-depth presentations
  - BSDCan 2018: slaacd(8)
  - BSDCan 2019: unwind(8)

  see openbsd.org/events.html

JOIN THE WI-FI (I)

- find Wi-Fi: `ifconfig iwm0 scan`
- network interface configuration: `ifconfig(8)` / `hostname.if(5)` / `netstart(8)`

```
$ cat /etc/hostname.iwm0
nwid home wpakey "trivial password"
inet autoconf
inet6 autoconf
up
```

JOIN THE WI-FI (II)

```
$ cat /etc/hostname.iwm0
join home wpakey "trivial password"
join work wpakey zUDciIezevfySqam
join "Airport Wi-Fi"
join ""
inet autoconf
inet6 autoconf
up
```

STOP SLACKING (I)

- slaacd(8)
    - IPv6 stateless address auto-configuration daemon
    - Forms Semantically Opaque Interface Identifiers & Temporary Interface Identifiers
    - handles nameservers
    - enabled per default

STOP SLACKING (II)

- slaacd(8)
  - handles multiple network interfaces
    - `ifconfig iwm0 inet6 autoconf`
    - `ifconfig em0 inet6 autoconf`
  - handles multiple default routers on a link
    - Hic sunt dracones

STOP SLACKING (III)

- slaacd(8)
  - privilege separated & pledged

    parent

    ```
    pledge("stdio inet sendfd wroute")
    ```

    frontend

    ```
    pledge("stdio unix recvfd route")
    ```

    engine

    ```
    pledge("stdio")
    ```

STOP SLACKING (IV)

- slaacd(8)
    - monitors network state
    - re-configures interfaces as needed
    - withdraws nameservers and proposes new ones as needed using a route(4) socket

DYNAMIC HOST CONFIGURATION, PLEASE (I)

- ~~dhclient(8)~~
- dhcpleased(8)
  - DHCP client
  - transmogrified slaacd(8)
  - enabled per default

DYNAMIC HOST CONFIGURATION, PLEASE (II)

- dhcpleased(8)
  - handles multiple network interfaces
    - `ifconfig iwm0 inet autoconf`
    - `ifconfig em0 inet autoconf`

DYNAMIC HOST CONFIGURATION, PLEASE (III)

- dhcpleased(8)
  - privilege separated & pledged

    parent

    can't pledge because of `bpf(4)`, unveils `/dev/bpf`, `/etc/dhcpleased.conf` and `/var/db/dhcpleased/`

    frontend

    `pledge("stdio unix recvfd route")`

    engine

    `pledge("stdio")`

DYNAMIC HOST CONFIGURATION, PLEASE (IV)

- dhcpleased(8)
  - monitors network state
  - re-configures interfaces as needed
  - withdraws nameservers and proposes new ones as needed using a route(4) socket

ROUTE PRIORITIES

- dhcpleased(8) & slaacd(8) handle multiple interfaces

```
Internet:
Destination        Gateway            Flags   Refs      Use   Mtu  Prio Iface
default            192.168.178.1      UGS        4      110     -     8 em0
default            192.168.178.1      UGS        0        0     -    12 iwm0
```

CELLULAR NETWORKS

- umb(4) for UMTS & LTE connectivity
- handled completely by the kernel
- nameservers are proposed via route(4) messages

IT IS ALWAYS DNS

- resolvd(8)
  - then: dhclient(8) owned `/etc/resolv.conf`
  - now: dhcpleased(8), slaacd(8), iked(8), and umb(4)
  - solution: resolvd(8) collects nameserver proposals
  - integrates manual edits of `/etc/resolv.conf`
  - enabled per default

LET US UNWIND A BIT (I)

- plain DNS is not secure
    - exposes every network tool to spoof-able untrusted data
- libc stub cannot do DoT / DoH / DoQ
- running unbound(8) puts us at the mercy of the local network

LET US UNWIND A BIT (II)

- unwind(8)
  - privilege separated recursive nameserver
    - libunbound for heavy lifting
  - resolvd(8) will automatically use it
    - `rcctl enable unwind && rcctl start unwind`
  - learns nameservers from dhcpleased(8) / slaacd(8)

LET US UNWIND A BIT (III)

- unwind(8)
  - DNSSEC validation
  - handles captive portals
  - monitors network conditions
  - DoT, recursion, or network nameservers
  - last resort: can behave exactly like libc stub
  - is pragmatic, no fanatical devotion to privacy

TIME FOR GELATO (I)

- Scenario: IPv6-only network with NAT64
    - maybe DNS64 as well
- unwind(8) can detect DNS64 & perform synthesis
- does not work with IPv4 literals or ping(8)

TIME FOR GELATO (II)

- gelatod(8)
    - half of 464XLAT
        - NAT64 gateway
        - Customer-side transLATor (CLAT) using pf(4)
    - detects NAT64 prefix from DNS64 or Router Advertisements

TIME FOR GELATO (III)

- complicated configuration: two pair(4) interfaces, one rdomain, and one pf(4) anchor

```
ifconfig pair1 inet 192.0.0.4/29
ifconfig pair2 rdomain 1
ifconfig pair2 inet 192.0.0.1/29
ifconfig pair1 patch pair2
route add -host -inet default 192.0.0.1 -priority 48
```

  - only in ports because of this

- generates pf(4) rule based on NAT64 prefix & our IPv6 address

```
pass in log quick on pair2 inet af-to inet6 \
    from 2001:db8::da68:f613:4573:4ed0 to 64:ff9b::/96 \
    rtable 0
```

QUESTIONS?

florian @ openbsd.org / @florian@bsd.network