A Method for Generating Semantically Opaque IPv6 Interface Identifiers (IIDs) with DHCPv6

(draft-gont-dhcwg-dhcpv6-iids-00)

F. Gont, SI6 Networks

dhc WG. IETF 122
Bangkok, Thailand. March 15th-21st, 2025

Motivation

- Security & privacy implications of predictable transient numeric identifiers have been discussed at length in RFC{9414, 9415, 9416}
    - Lack of formal recommendations -> flawed implementations
- RFC 9416 / BCP 72 introduces **requirements for these identifiers**:

    Protocol specifications SHOULD recommend an algorithm for generating their transient numeric identifiers

- DHCPv6-assigned IPv6 addresses/IIDs are one of such identifiers
- This document aims at producing a Std Track version of RFC 7943
    - That can be formally recommended in future DHCPv6-specs without a downref

Generation of IPv6 Addresses / IIDs

- Flawed DHCPv6-assigned addresses/IIDs can lead to:
  - Security & privacy issues [RFC7721] resulting from:
    - Flawed algorithms
    - Reduced address pool size
  - Interoperability/operational issues:
    - Unstable addresses (e.g. device reboot & lack of stable storage)
    - Artificial limits on address assignment (at odds with RFC 7934)
- A number of DHCPv6-server implementation are known to do this
- Similar considerations apply for prefixes generated for DHCPv6-PD

draft-gont-dhcwg-dhcpv6-iids-00

- Std Track version of RFC 7943 (Informational) -- a DHCPv6 version of RFC 7217 (employed with SLAAC)
- Proposes an algorithm to select IPv6 IIDs that:
  - Do not result in address patterns
  - Are stable within each network, but change across networks
  - Do not require stable storage (for the lease database) across reboots
- Multiple DHCPv6 servers can generate the same IIDs, without lease database syncronization (just set the same secret_key)

Algorithm: RID computation

`RID = F(Prefix | Client_DUID | IAID | Counter | secret_key)`

Where:

- RID: Randomized identifier (will be the basis for the leased address)
- F(): Hash function (we recommend SHA-256)
- Prefix: IPv6 prefix employed for the address pool
- Client_DUID: DUID contained in the received Client Identifier Option
- IAID: IAID contained in the received IA_NA option
- Counter: Counter value that can be employed to resolve address conflicts
- secret_key: Secret key (unknown to the attacker)

Algorithm: Steps

1) Compute RID
2) Select candidate address as:

```
   IPV6_ADDR = IPV6_ADDR_LOW +
                     RID % (IPV6_ADDR_HI - IPV6_ADDR_LOW + 1)
```

3) Compare IPv6 IID to reserved IIDs: if unacceptable, increment **`Counter`** and go back to step #1
4) If the address is unavailable or marked as “declined”, increment **`Counter`** and go back to step #1
5) Use the computed address

Next steps

- Comments/questions?
- Adopt as WG document?