Automatic DNSSEC Bootstrapping with Authentication

CENTR Jamboree 2023
June 1, 2023

Peter Thomassen <peter@desec.io>

draft-ietf-dnsop-dnssec-bootstrapping
DNSSEC validation rate vs. secure delegation rate

31% vs. 7%

○ Germany 70%
○ Scandinavia 90%
○ Russia 63%
○ Saudi Arabia 99%

○ 50–70% in some places
○ even for signed zones: < 50%

Problem Statement

Securing a delegation = Ry creates DS record with child’s DNSSEC parameters.

Problems:

1. **How does the Registry get these parameters?**
   - Largely the same problem as “DS Automation” for rollovers

2. **How are those parameters authenticated?**
   - NB: for key *rollovers*, existing chain of trust can be used. Not here!

3. **What else is there to consider?**
Approaches to DNSSEC Bootstrapping

1. manual submission
   ○ Generally supported, but cumbersome
Approaches: Manual Submission

- Involves the Child DNS Operator (origin) and Parent Registry (recipient)
  - ... typically with the Registrar as the messenger
  - ... typically facilitated through the Registrant

- Slow
- Error-prone
- Out of band
- Not properly authenticated
Approaches to DNSSEC Bootstrapping

1. **manual submission**
   - Generally supported, but cumbersome

2. **trust on first use (TOFU):** query DNSKEY, compute DS, and hope for the best
   - Used by notable Registrar in Germany
Approaches: Trust on First Use (various interfaces)

- No manual dealing with cryptographic parameters
- Known timing
- No authentication!
Approaches to DNSSEC Bootstrapping

1. **manual submission**
   - Generally supported, but cumbersome

2. **trust on first use (TOFU)**: query DNSKEY, compute DS, and hope for the best
   - Used by notable Registrar in Germany

3. **Several attempts on REST interfaces** or REST-DNS hybrids, driven by CIRA
   - No known deployments

   “Need to redesign around the DNS Operator”
   ― Jacques Latour, Tech Day at ICANN 53
Approaches to DNSSEC Bootstrapping

1. manual submission
   ○ Generally supported, but cumbersome
2. trust on first use (TOFU): query DNSKEY, compute DS, and hope for the best
   ○ Used by notable Registrar in Germany
3. Several attempts on REST interfaces or REST-DNS hybrids, driven by CIRA
   ○ No known deployments
4. CDS/CDNSKEY from insecure child (RFC 8078)
   ○ Requires stateful monitoring
Approaches: CDS/CDNSKEY from Insecure Child

- Example 
  - example IN DS 
  - example.com.
  - @ IN CDS 
  - @ IN CDNSKEY 

Registry/Registrar for example.com.
Approaches to DNSSEC Bootstrapping

1. **manual submission**
   - Generally supported, but cumbersome

2. **trust on first use (TOFU):** query DNSKEY, compute DS, and hope for the best
   - Used by notable Registrar in Germany

3. **Several attempts on REST interfaces or REST-DNS hybrids, driven by CIRA**
   - No known deployments

4. **CDS/CDNSKEY from insecure child (RFC 8078)**
   - Requires stateful monitoring

5. **CDS/CDNSKEY with authentication** by child operator (IETF DNSOP draft)
   - Used by .ch/.li (parent) and Cloudflare/deSEC/Glauca HexDNS (child)
   - Implementations exist for PowerDNS Auth and Knot DNS (upstream PRs coming up)
Approaches: CDS/CDNSKEY with Authentication

1. Define a signaling mechanism for DNS operators
   ○ allow publishing arbitrary information about the zones under management, on a per-zone basis
   ○ do so using namespace under each nameserver hostname with zone-specific subdomains
   ○ require DNSSEC for authentication (requires nameserver domains to be secure)

2. Ask DNS Operators to publish authentication signal for CDS/CDNSKEY
   ○ start with conventional CDS/CDNSKEY records at the apex of the target zone (RFC 8078)
   ○ co-publish these records via signaling mechanism (signed with NS zone's keys)

3. Validate target domain’s CDS/CDNSKEY records against this signal
   ○ if successful: “transfer trust to the target domain”
     → provision DS records at parent
Approaches: CDS/CDNSKEY with Authentication

💡 Use an established chain of trust (left) to take a detour
● identically co-published
● authenticated, immediate
● no active on-wire attacker

Extends RFC 8078 to add authentication for initial DS

Registry/Registrar for example.com.

\_signal.ns1.provider.net.

provider.net.

net.

com.

eexample IN DS

example.com.

_registry

_dsboot.example.com  IN CDS
_dsboot.example.com  IN CDNSKEY

Registry/Registrar for example.com.
It’s already in Production

---

Child:

- **3 DNS operators**, for all DNSSEC-enabled domains
  - deSEC
  - Cloudflare (manages **23% of Top 1M domains**)
  - GlaucHexDNS

Parent:

- **2 ccTLDs**: .ch/.li (+.cl testing)
- gTLDs in ICANN process to ensure consistent behavior
- GoDaddy to introduce automatic **DNSSEC bootstrapping** as a Registrar

Sources: Cloudflare coverage measurement: Nils Wisol via Tranco (06/2022); [https://github.com/oskar456/cds-updates](https://github.com/oskar456/cds-updates)
CDS & CDNSKEY (and CSYNC): Things to Think about …

● Who’s in charge of scanning? Registry vs. Registrar
  ○ What if not done?
● CDS/CDNSKEY dichotomy: which to publish in the child?
● Acceptance checks: validation breakage? CDS ~ CDNSKEY?
  ○ draft-thomassen-dnsop-cds-consistency
● Registry lock: suspend scanning during EPP locks?
● Error reporting: to whom? How? How frequently?
● Competing submissions: e.g. by the registrar or via GUI
● Efficiency improvements: notification trigger instead of scanning
  ○ draft-thomassen-dnsop-generalized-dns-notify
● …
You are invited!

- draft-ietf-dnsop-dnssec-bootstrapping on the way to IETF DNSOP Last Call
  - Vocal support on the mailing list always helps (dnsop@ietf.org)

- Child-side implementations
  - deployed at DNS operators
  - being developed for open source auth nameservers (close to done for PowerDNS & Knot DNS)

- Now: need parent-side implementations 😢
  - add authentication to existing CDS/CDNSKEY scanning implementations (~6 ccTLDs)
  - others: start scanning for CDS/CDNSKEY under more TLDs

- Let’s make DNSSEC easy.
Thank you!

... also to our supporters:

- SSE
- nlnet
- RIPE NCC

Questions?
Backup
Protocol Details

Algorithm

- Co-publish CDS/CDNSKEY records under a subdomain of the NS hostnames:
  \[\text{CDS/CDNSKEY IN } \_dsboot.example.com\_signal.ns1.provider.net\]
- Use DNSSEC to validate these records, under each NS hostname

Technical Considerations

- Naming scheme with \_signal label allows delegating to separate zone
  - removes risk of accidentally modifying the nameserver’s A/AAAA records
  - reduces churn on nameserver zone
  - allows splitting off DNS operations (e.g. online-signing with different key; delegate by parent)
- Prefix allows different types of signals (e.g. for multi-signer p2p key exchange)
# Who’s in Charge of Polling?

<table>
<thead>
<tr>
<th>DS Flow</th>
<th>Registrar</th>
<th>Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Operator → Registrar → Registry (no EPP backchannel needed)</td>
<td>DNS Operator → Registry → Registrar (requires EPP backchannel, RFC 9167)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deployment (today)</th>
<th>Registrar</th>
<th>Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 1 (Domainnameshop)</td>
<td>- 10 (9 ccTLDs + RIPE)</td>
<td></td>
</tr>
<tr>
<td>- 1 planned (GoDaddy, since 2020)</td>
<td>- Several gTLDs ready (CentralNIC, CORE)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope</th>
<th>Registrar</th>
<th>Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Covers gTLD and ccTLD names</td>
<td>- Covers only gTLD names</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pros</th>
<th>Registrar</th>
<th>Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Preserves customary flow</td>
<td>- Adoption appears easier in Ry space</td>
<td></td>
</tr>
<tr>
<td>- Fewer steps to DS (EPP notify is async)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cons</th>
<th>Registrar</th>
<th>Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>- TLD query and/or EPP rate limit</td>
<td>- No ccTLD coverage in Ry agreements, potentially limiting recommendation scope</td>
<td></td>
</tr>
<tr>
<td>- Adoption difficult, many Registrars</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Some even lack DS interface today</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Some charge for setting DS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- NOTIFY target discovery unclear</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Locks

<table>
<thead>
<tr>
<th>Lock</th>
<th>Update</th>
<th>Delete</th>
<th>Transfer</th>
<th>Renew</th>
<th>DS automation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Registry</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>Transfer Lock</code></td>
<td></td>
<td></td>
<td>prohibited</td>
<td></td>
<td>allowed</td>
</tr>
<tr>
<td><code>serverUpdateProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>serverDeleteProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>serverTransferProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>serverRenewProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>URS Lock</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ccTLD-specific Lock</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Registrar</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>clientUpdateProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>allowed</td>
</tr>
<tr>
<td><code>clientDeleteProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>allowed</td>
</tr>
<tr>
<td><code>clientTransferProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>allowed</td>
</tr>
<tr>
<td><code>clientRenewProhibited</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>allowed</td>
</tr>
</tbody>
</table>
Security Model

- We use an established chain of trust to take a detour
  - authenticated, immediate
  - no active on-wire attacker

- Actors in the chain of trust can undermine the protocol
  - can also undermine CDS / CDNSKEY from insecure

- Mitigations exist, e.g:
  - monitor delegation
  - diversify NS TLDs
  - multiple vantage points
<table>
<thead>
<tr>
<th><strong>BOOTSTRAPPING INVOLVES</strong></th>
<th><strong>MANUAL</strong></th>
<th><strong>BOOTSTRAPPING METHOD</strong></th>
<th><strong>PROPOSED</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>zone operator Z</td>
<td>✓¹</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>domain owner</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>registrar</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>registry</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ACTORS WHO CAN INITIALIZE KEYS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required parties (trusted)</strong></td>
</tr>
<tr>
<td>registrar</td>
</tr>
<tr>
<td>NS zone operator</td>
</tr>
<tr>
<td>NS zone ancestors</td>
</tr>
<tr>
<td>NS zone owner</td>
</tr>
<tr>
<td><strong>Others parties (untrusted)</strong></td>
</tr>
<tr>
<td>active on-wire attacker</td>
</tr>
<tr>
<td>social engineering attacker [1]</td>
</tr>
</tbody>
</table>

| **PROPERTIES** |
|----------------|-----------------|-----------------|----------------|
| Prerequisites  | out-of-band channel | MITM attack mitigation | suitable NS zone configuration |
| Authentication | bad in practice [1] | none days | cryptographically minutes |
| Duration       | varies           |     |                           |

Table 1: Comparison of methods for establishing a new secure delegation, displaying a) entities involved in the bootstrapping of an individual insecure zone, b) attack surface towards trusted and untrusted third parties, and c) prerequisites, key material authentication, and bootstrapping duration. Key initialization within parentheses (✓) requires collusion across all NS zones. ¹ For offline signing, only the signing key holder is involved. ² Registry could refuse deployment through registrar. ³ Requires knowledge of private key. ⁴ Several vantage points and long time must be covered.