ATHENE
National Research Center
for Applied Cybersecurity

Protocol Fixes for KeyTrap Vulnerabilities

Elias Heftrig, Haya Schulmann, Niklas Vogel, Michael Waidner

Outline

- Recap on KeyTrap Attacks
- Problems with Short-Term Mitigations
- Suggested Protocol Fixes
- Discussion

Recap on KeyTrap Attacks

DoS by DNSSEC Validation

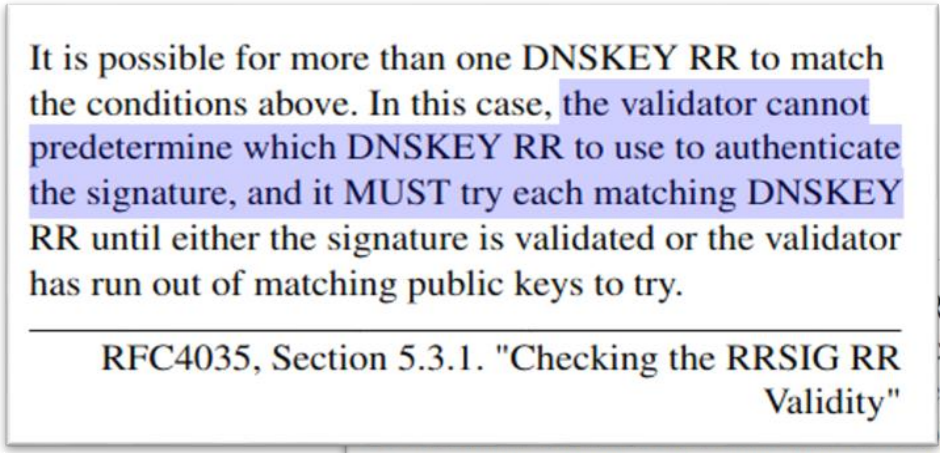
„A potentially Internet-killing vulnerability“

- Internet Pioneer during Disclosure

- CPU resource exhaustion attacks on DNSSEC validators
- High impact
  - Resolvers could be stalled up to 16h with just a single response
- Low resources
  - Host a malicious domain and get a victim resolver to resolve a name
- All tested implementations found vulnerable
  - Resolvers, Libraries, Debugging Tools, …
- Abundance of vulnerable networks

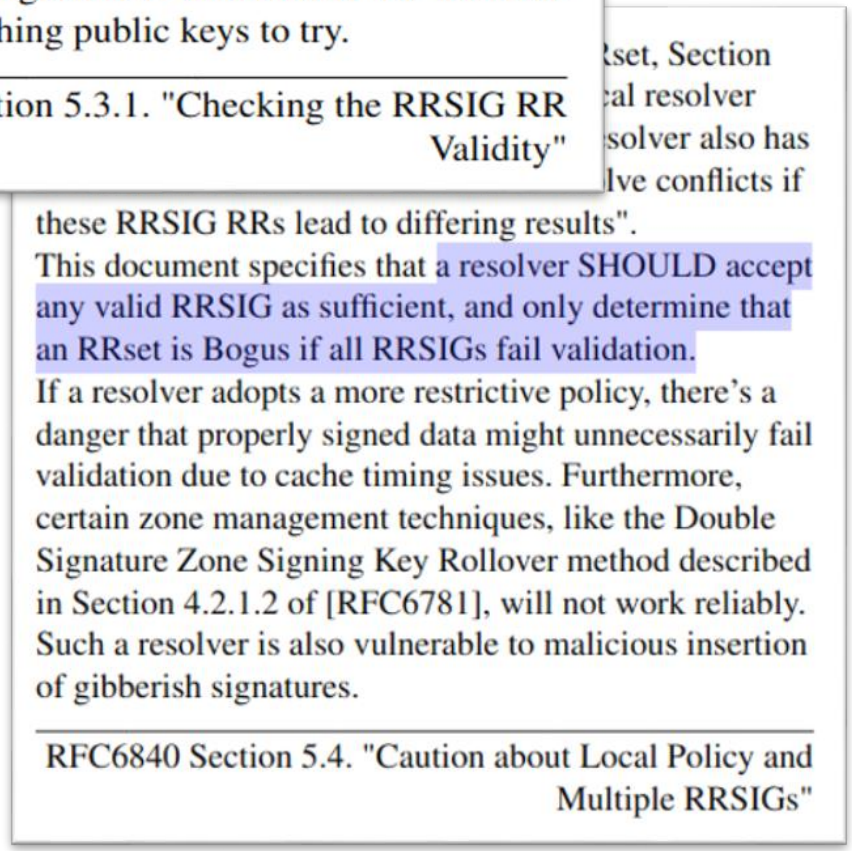
  → appx. 1/3 of web clients worldwide use validating resolvers
- Patching against KeyTrap required tight coordination with a multi-vendor, >30 heads task force

Exploitation of DNSSEC Protocol Design



It is possible for more than one DNSKEY RR to match the conditions above. In this case, the validator cannot predetermine which DNSKEY RR to use to authenticate the signature, and it MUST try each matching DNSKEY RR until either the signature is validated or the validator has run out of matching public keys to try.

RFC4035, Section 5.3.1. "Checking the RRSIG RR Validity"



these RRSIG RRs lead to differing results".
This document specifies that a resolver SHOULD accept any valid RRSIG as sufficient, and only determine that an RRset is Bogus if all RRSIGs fail validation.
If a resolver adopts a more restrictive policy, there's a danger that properly signed data might unnecessarily fail validation due to cache timing issues. Furthermore, certain zone management techniques, like the Double Signature Zone Signing Key Rollover method described in Section 4.2.1.2 of [RFC6781], will not work reliably. Such a resolver is also vulnerable to malicious insertion of gibberish signatures.

RFC6840 Section 5.4. "Caution about Local Policy and Multiple RRSIGs"

- "Eager validation" approach to ensure robustness against validation errors
  - Try all possible DNSKEYs for an RRSIG until one works
  - Try all possible RRSIGs for an RRset until one works
- Specification implies complex algorithms over expensive public-key crypto operations
  - First vulnerable requirements date back to 1998
- Patches necessarily break core specification requirements

Fundamental Problem Exposed by KeyTrap

CPU resource exhaustion has never been properly addressed in DNSSEC until KeyTrap

- RFC4033 and RFC4035 generally warned about resource exhaustion attacks
- NSEC3 specification initially addressed resource requirements from SHA1 iteration counts
- Ideation of a DNSKEY-only attack by Dutch Bachelor‘s student (not weaponized)
- RRSIG-based CPU resource exhaustion attack exploiting RRSIGs over NSEC RRs in previous work

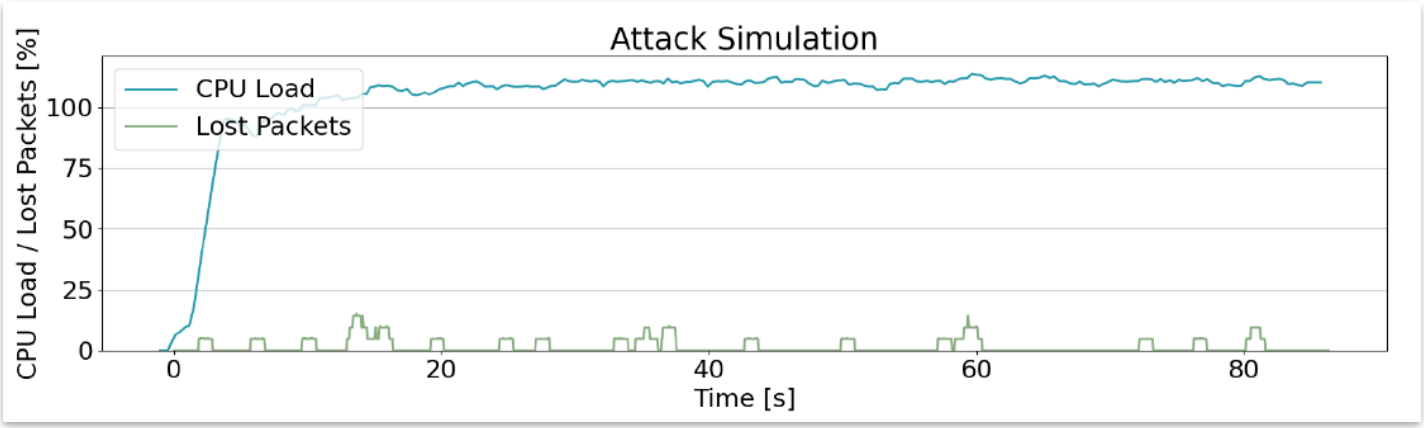
Openness of DNS(SEC) protocol semantics allows for a plethora of KeyTrap-like attack vectors

- Exploitation of DS hashing and RRSIG validation
- Exploitation of valid and invalid signatures
- Attacks covering different RRsets (cnf. Protocol semantics)

The *Short-term* Fixes against KeyTrap address the attack vectors but introduce (so-far) unmanaged complexity

Problems with Short-Term Mitigations

Architectural Containment



➢ All-valid RRSIGs attack on patched BIND9

Scheduling-based countermeasures

- Intermitting long-running validation tasks to allow other tasks in the pipeline to proceed
  → Still allows waste of (low-priority) CPU cycles – economic attacks?

Limiting Cryptographic Operations

Limits encompass the numbers of …

- RRSIGs tried to validate a given RRset
- DNSKEYs tried to validate a given RRSIG
- DS RRs tried to validate a given DNSKEY
- Failed or attempted validations per message
- RRSIG and DS validations per resolution

> The amount of work which a resolver will do in response to a client request must be limited to guard against errors in the database, such as circular CNAME references, and operational problems, such as network partition which prevents the resolver from accessing the name servers it needs.
>
> RFC1035, Section 7.1 "Transforming a user request into a query"

→ Per-resolution limits extend general DNS resolver instructions from RFC1035 to DNSSEC

Problems with current per-resolution Limits

Inconsistent selection of limits

- Limits and their values are hardcoded or set by configuration file
- Desirable and (arguably necessary) to adapt to individual resolver requirements
- Problematic in absence of a mechanism to signal and adapt name server responses to these limits

→ factor of unreliability, disincentivizing domain-side use of DNSSEC

Limits to DNSSEC imply limits to DNS → Layer Violation

- Add complexity to the already complex DNS(SEC)
- Restrict scalability of DNS (“DNS Security Restrictions”?)
- Hamper future DNS protocol development
- Managing validation complexity in face of (per-resolution) limited validation budgets is challenging

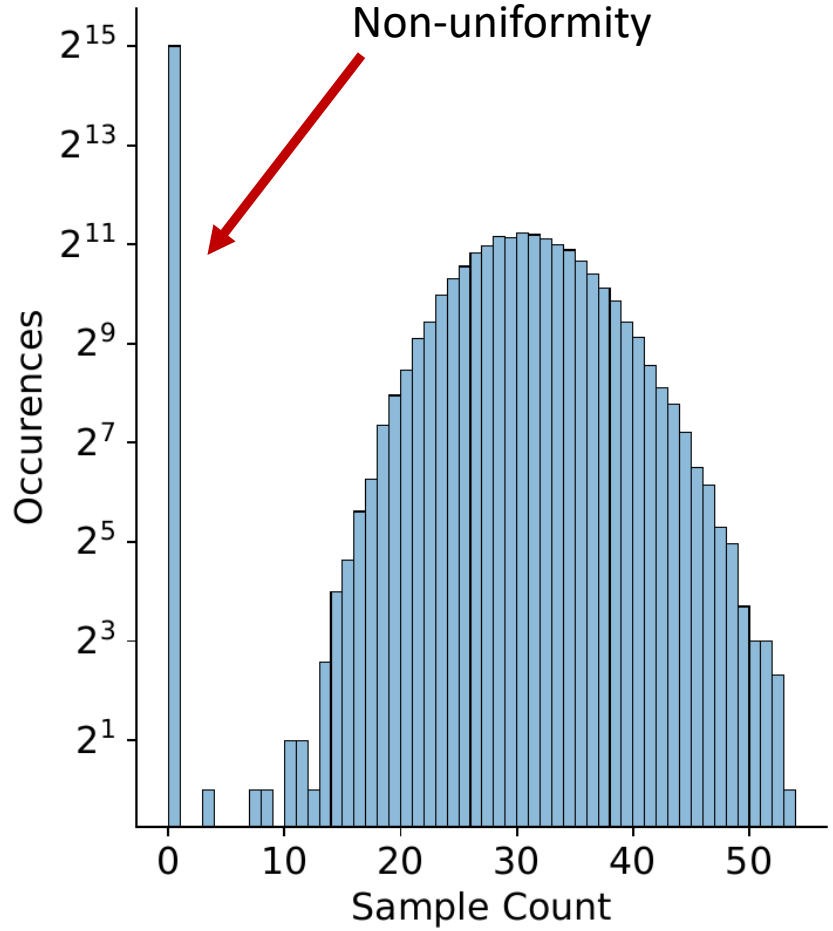
Factors Driving Complexity of Validation

Number of RRsets requiring validation in responses

- Introduction of new record types (e.g. DELEG)
- Elective validation (of, e.g., infrastructure RRsets)
- Openness to future DNS use cases

KeyTag collisions

- Induce ‘natural’ validation failures
- Make validation complexity a matter of probability
- Don’t necessarily follow a uniform random distribution
  - Collision probability depends on DNSSEC algorithm



➢ Frequencies of KeyTag observations in 1M dice-rolled RSASHA256 keys

Factors Driving Complexity of Validation

Crypto-agility

- Future algorithms that increase CPU load may require global revision of local validation limits
- Different crypto libraries varying in CPU requirements

Additional promoters of complexity

- Varying depth of delegation
  - Domains may require multiple resolutions to get resolved (corner case bugs)
- Cross-zone coordination
- Depth of recursion (esp. CNAMEs)

Suggested Protocol Fixes

Managing Validation Budgets

Set a global minimum per-resolution validation budget in the specification

- Not considering elective validations or cache
- Reflecting current operational insights and updated over time

→ allows inter-zone budget alignment – caveat: needs to consider aliasing

Introduce EDNS0 options to signal…

- Total and current validation budgets from resolvers to name servers
- Validation budget depletion error from resolvers to clients

→ supports global monitoring of validation budgets at domains and Internet nodes

Outlawing KeyTag collisions

- Require KeyTag to uniquely identify a DNSKEY in a zone
  - Blunt confrontation to RFC4034, requiring the opposite
- Just changing semantics of current records would require worldwide coordination

→ hard to enforce without breaking things

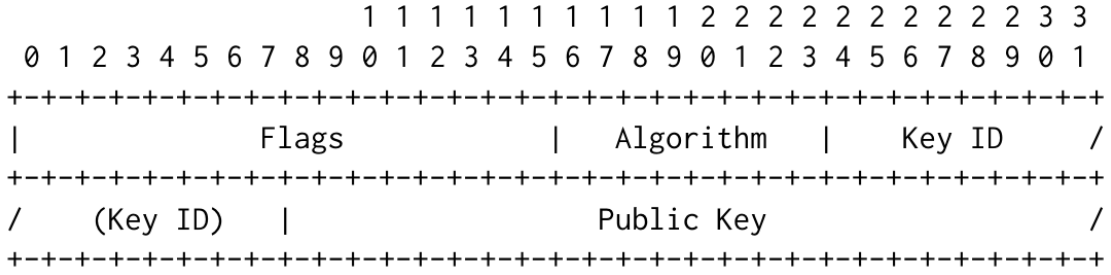
Solution: RFC3755-style introduction of new key record

> However, it is essential to note that the key tag is not a unique identifier. It is theoretically possible for two distinct DNSKEY RRs to have the same owner name, the same algorithm, and the same key tag. The key tag is used to limit the possible candidate keys, but it does not uniquely identify a DNSKEY record. Implementations MUST NOT assume that the key tag uniquely identifies a DNSKEY RR.
>
> RFC4034, Appendix B "Key Tag Calculation"

Introducing IDKEY

Replacement DNSKEY Record

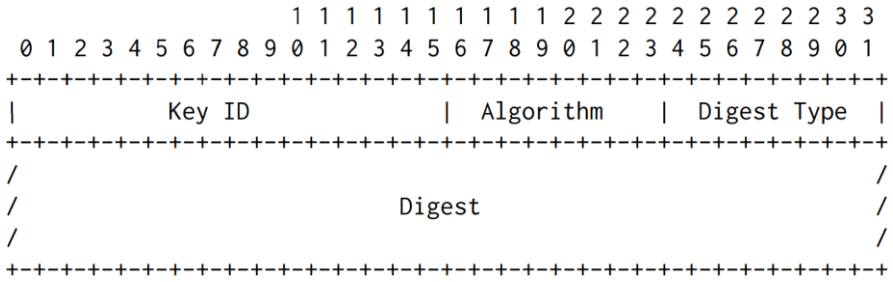
- Features 16-bit Key ID, required to be unique per zone
  - Validators are required to insist on uniqueness
- RRSIGs are re-purposed (Key ID ~ Key Tag)
  - Allows to keep responses small during transition
- DS replaced by analogue IDDS
  - Required for secure fallback to insecure in transition



➢ IDKEY Record Format

Transitioning to IDKEY

- Resolvers supporting IDKEY query for both DNSKEY and IDKEY during transitioning, either may be used
- Domains set Key ID := Key Tag during transitioning and provide both DNSKEY and IDKEY sets



➢ IDDS Record Format

Relax Absolute Validation Requirements

MUST [...], or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

RFC2119 "Key words for use in RFCs to Indicate Requirement Levels"

It is possible for more than one DNSKEY RR to match the conditions above. In this case, the validator cannot predetermine which DNSKEY RR to use to authenticate the signature, and it MUST try each matching DNSKEY RR until either the signature is validated or the validator has run out of matching public keys to try.

RFC4035, Section 5.3.1. "Checking the RRSIG RR Validity"

When multiple RRSIGs cover a given RRset, Section 5.3.3 of [RFC4035] suggests that "the local resolver security policy determines whether the resolver also has to test these RRSIG RRs and how to resolve conflicts if these RRSIG RRs lead to differing results".
This document specifies that a resolver SHOULD accept any valid RRSIG as sufficient, and only determine that an RRset is Bogus if all RRSIGs fail validation.
If a resolver adopts a more restrictive policy, there's a danger that properly signed data might unnecessarily fail validation due to cache timing issues. Furthermore, certain zone management techniques, like the Double Signature Zone Signing Key Rollover method described in Section 4.2.1.2 of [RFC6781], will not work reliably. Such a resolver is also vulnerable to malicious insertion of gibberish signatures.

RFC6840 Section 5.4. "Caution about Local Policy and Multiple RRSIGs"

When validating a response to QTYPE=*, all received RRsets that match QNAME and QCLASS MUST be validated. If any of those RRsets fail validation, the answer is considered Bogus.

RFC6840, Section 4.2. "Validating Responses to an ANY Query"

- Degrade MUST-requirements to SHOULD in RFCs 4035 and 6840
- Warn about taking the SHOULD requirements literally

Discussion

Questions raised for future DNSSEC

- Can we tolerate waste of low-priority CPU cycles or do we need to apply limits to DNSSEC validation?
- Can we manage the complexity induced by the local per-resolution validation limits?
  - Are communicated validation budgets the way to go?
- Do we need to outlaw non-unique DNSKEY identification or can we just bare the probability of collisions?
- Should we opt for more radical approaches?
  - Proof of work from clients?
  - Validating DNSSEC only at the client?
- Tighter revision of standards? Increased use of formal methods?