DNS security basics

The basics

Karst Koymans

Informatics Institute
University of Amsterdam
(version 19.4, 2019/09/25 13:38:41)

Tuesday, September 24, 2019

Table of Contents

In what sense is DNS insecure?

- DNS data can be subject to forgery
  - Non-authoritative servers can try to inject false information
  - Records can be changed in transit to point to evil information
    - Like "A" records pointing to a phisher's IP address
- DNS data traverses the network in clear text
  - Anyone with network path access can eavesdrop on your DNS traffic
  - Gleaning information useful in spoofing attacks
- DNS data can even be spoofed by blind guesses
  - Guess query transaction ID and source port (Kaminsky attack)
  - You can try as often as you like

What DNSSEC has to offer

- Protects against forgery
  - Uses public key cryptography
  - Cryptographically signs the resource record sets in responses
  - Builds a chain of trust from the root down...
    - ...or from some domain up
- Does not prevent eavesdropping
  - Data still traverses the network in clear text
  - Gleaning data is still possible
  - But spoofing is not possible any more
    - Except for things like flags

DNSSEC specification

- Original specification from January 1997
  - RFC 2065
- Revised specification from March 1999
  - RFC 2535
  - Incorporated feedback from early users
  - Had deployment problems, especially scaling issues
- "Final" specification from March 2005
  - DNSSEC-bis (RFC 4033, 4034 and 4035)
- "Final" addition from February 2008
  - NSEC3 (RFC 5155)

Alternative DNS security mechanism

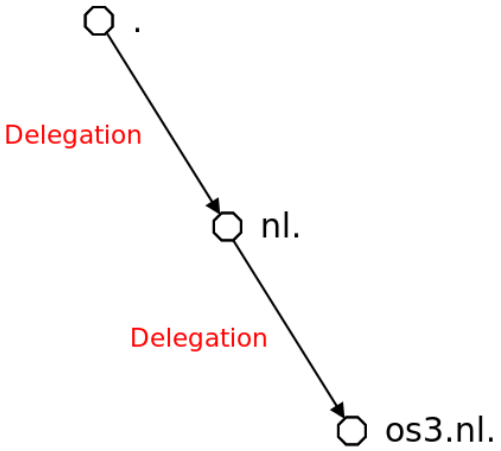
- DNSCurve
  - Idea by Daniel J. Bernstein
  - Proposed in August 2008 (after NSEC3 spec)
  - Encrypts and authenticates on the transport layer
  - Signs communication packets, not resource records
  - Uses labels of name server domain names to distribute public keys
  - Uses state of the art elliptic curve cryptography for speed
  - Worth a read at `https://dnscurve.org/`
    - Also see `https://curvecp.org/`

Basic tree walking mechanism

- Validating an RRset from the root (".") down
- Suppose we ask for the type A RRset for "www.os3.nl."
  - Start at the root, which is a trust anchor[1]
  - Verify the authenticity of the delegation to "nl." zone
  - Verify the authenticity of the delegation to "os3.nl." zone
  - Verify the authenticity of the RRset for "www.os3.nl."

[1] Other trust anchors can be given by explicit configuration or DLV (DNSSEC Lookaside Validation), both not needed any more

Delegations of authority



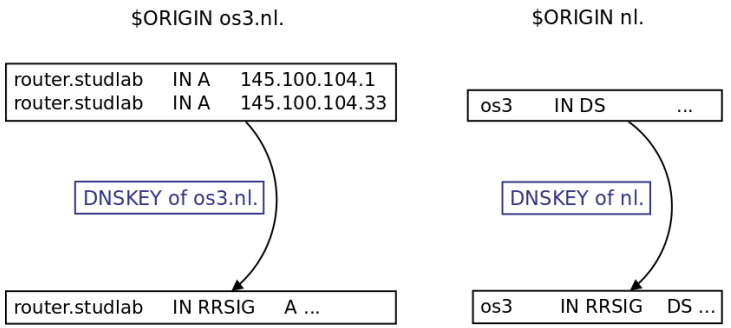
Source: Niels Sijm, CIA lecture 2012-2013

DNSSEC-bis resource records

- Resource Records used to build the chain of trust
  - DNSKEY
    - DNS (public) KEYs belonging to a zone, published at the apex
    - Used to verify signatures that have been added to the zone
    - Root DNSKEYs[2] are well known
  - DS
    - Delegation Signer
    - Contains a hash of the DNSKEY of a delegation (child) zone
    - A DNSKEY hashed in the parent is called a secure entry point (SEP)
  - RRSIG
    - Resource Record (set) SIGnature
    - Contains the signature of an RRset of a given type

[2] To be more precise these are known as the root Key Signing Keys (KSKs)

Signing Resource Record sets



Adapted from: Niels Sijm, CIA lecture 2012-2013

What RRsets should be signed using RRSIG?

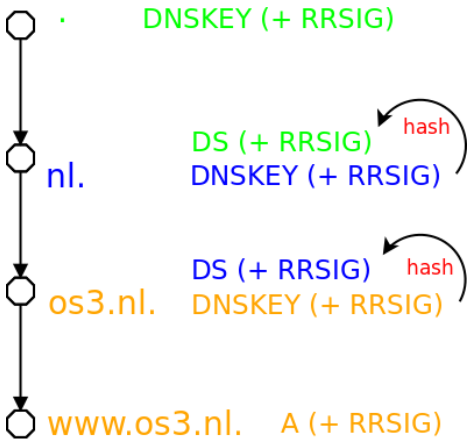
- All authoritative data
- Including the NS RRset at the apex in the child zone (by the child)
- But not including the NS RRsets at delegation points
  - Instead the DS RRset is signed, if it exists, which is therefore also considered authoritative in the parent zone
    - This authority issue can be solved by better naming in graphs
- And also not including RRsets of all other glue

Tree walking algorithm

- Validating delegation from "." (root) to "nl." zone (top down)[3]
  1. Ask "." zone for DS of "nl." zone
  2. Ask "nl." zone for DNSKEY of "nl." zone
  3. Verify that DS contains a valid hash of DNSKEY
- Three steps are missing...
  5. Get the public part of the root zone's signing key
  6. Get the signatures of the DS record
  7. Check that the DS record for "nl." is authentic

[3] Ignoring KSKs and ZSKs for simplicity, for now

Resource record types used



Adapted from: Niels Sijm, CIA lecture 2012-2013

Detailed tree walking algorithm (1)

1. Retrieve DNSKEY for root zone[4]
   - Stored in resolver as a secure starting point[5]
   - Can also be retrieved from root nameservers (TOFU?)
     - `dig @a.root-servers.net. . dnskey`
2. Ask root zone for the DS of the "nl." zone
   - Contains a hash of the DNSKEY of the "nl." zone
   - Stored in the root zone
     - `dig [@a.root-servers.net.] nl. ds`

[4] Still ignoring difference between KSK and ZSK
[5] Alternatively get the root trust anchor in DS form from IANA

Detailed tree walking algorithm (2)

3. Ask "nl." zone for DNSKEY of "nl." zone
   - `dig [@ns1.dns.nl.] nl. dnskey`
4. Verify that DS contains a valid hash of DNSKEY
   - Create the hash of the DNSKEY RRset yourself
   - Compare the result to the hash in the DS record

Detailed tree walking algorithm (3)

- Don't forget the following steps
  5. Retrieve signature of "nl." DS RRset
     - Stored in root zone
     - `dig +dnssec [@a.root-servers.net] nl. ds`
     - Without the `+dnssec` flag you don't get the RRSIG info needed
  6. Verify this signature using the root zone DNSKEY, thereby proving the authenticity of the DS record

Multiple DNSKEY records

- Zones tend to contain multiple DNSKEY records in the apex
  - Usually there are at least two DNSKEYs
    - A KSK (Key Signing Key) as secure entry point for chain of trust
    - A ZSK (Zone Signing Key) for RRSIG creation
  - A ZSK may change much more often than a KSK
- This is not necessary for DNSSEC operation
- It simplifies DNS key management
  - For instance key rollover should be easier
    - First root KSK key rollover planned on 20171011, but delayed
    - The year after it was for real on 20181011, 16:00 UTC

DNSSEC-bis chain of trust (1)

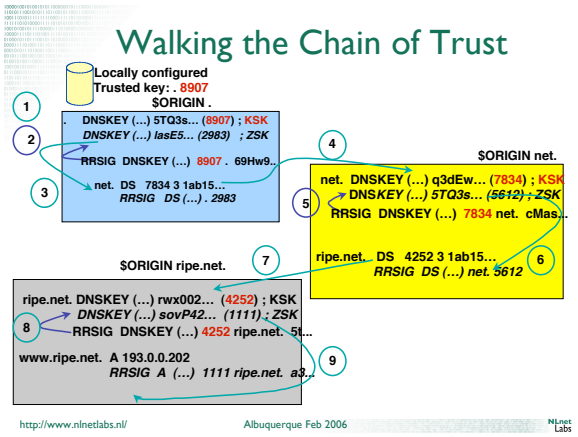
This time we present the algorithm more abstractly and bottom up

- To validate a resource record set RRset
  - Validate RRSIG(RRset)
    - by using a ZSK (zone signing key) from the DNSKEY RRset
  - Validate RRSIG(DNSKEYset)
    - by using a KSK (key signing key) from that same DNSKEY RRset
  - Validate the KSK
    - by using a DS (present in the parent zone) which contains a hash of the KSK
    - the KSK used is called a SEP (Secure Entry Point)

DNSSEC-bis chain of trust (2)

- Continue validating one level higher up in the hierarchy
  - Use DS as RRset and iterate
- Use trust anchors or DLV for (DNSKEYs or) DSs to end the checks
  - for instance by checking root zone keys
- Authority around a cut is now as follows
  - NS and DNSKEY (plus their RRSIGs) are authoritative on the child side of the cut (zone apex)
  - DS (and its RRSIG) is authoritative on the parent side of the cut (delegation point)

Chain of trust illustration (top down using KSK/ZSK)



Source: Olaf Kolkman, RIPE NCC, NLnet Labs

RRSIG record example from RFC 4034

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                20030220173103 2642 example.com.
                                oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t
                                GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

This is the text representation format
In wire format, dates are in seconds since January 1, 1970
and the signature is sent as is (binary)

RRSIG record content

Part	Meaning
Type covered	Record type this RRSIG is about
Algorithm	Signature algorithm used; 5 is RSA/SHA-1
Labels	Number of labels of owner[6]
TTL	Original time to live
Expiration and Inception	Signature validity date bounds
Key Tag	To help find[7] (not identify) the signing key
Signer's Name	Owner name of zone and key to use
Signature	Signature (in Base64)

[6] Not counting the root label and a (possible) asterisk ("*") label (wildcard) in front

[7] In fact it is a non-cryptographic checksum on the RDATA of the DNSKEY RR

DNS Security Algorithm Numbers

Number	Meaning	RFC
5	RSA/SHA-1	3110/4034
8	RSA/SHA-256	5702
10	RSA/SHA-512	5702
15	Ed25519	8080
16	Ed448	8080

Source: https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml

DNSKEY record example from RFC 4034

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w== )
```

This is the text representation format
In wire format the key is sent as is (binary)

DNSKEY record content

Part	Meaning
Flags	Zone key (KSK+ZSK); Secure entry point (KSK)
Protocol	Always 3 (for backward compatibility with KEY RR)
Algorithm	Signature algorithm used; 5 is RSA/SHA-1
Public key	Key used for signing (in Base64)

Note that the key tag can be calculated from this information
by a simple checksum (16-bit modular sum) calculation

DS record content

Part	Meaning
Key Tag	To help find (not identify) the signing key
Algorithm	Signature algorithm of the signing key
Digest Type	Hashing algorithm used; 1 is SHA-1, 2 is SHA-256
Digest	Sequence of case-insensitive hexadecimal digits

This is the text representation format
In wire format the digest is sent as is (binary)

DS record example from RFC 4034 and 4509

```
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiROGOMYkDshWoSKz9Xz
                                             fwJr1AYtsmx3TGkJaNXVbfi/
                                             2pHm822aJ5iI9BMzNXxeYCmZ
                                             DRD99WYwYqUSdjMmmAphXdvx
                                             egXd/M5+X7OrzKBaMbCVdFLU
                                             Uh6DhweJBjEVv5f2wwjM9Xzc
                                             nOf+EPbtG9DMBmADjFDc2w/r
                                             ljwvFw==
                                           ) ; key tag = 60485 ; in child zone
; RFC 4034 and 4509 mistakenly use "key id" instead of "key tag" here

dskey.example.com. 86400 IN DS 60485 5 1 (
               2BB183AF5F22588179A53B0A98631FAD1A292118 ) ; SHA-1 ; in parent zone

dskey.example.com. 86400 IN DS 60485 5 2 (
               D4B7D520E7BB5F0F67674A0C
               CEB1E3E0614B93C4F9E99B83
               83F6A1E4469DA50A ) ; SHA-256 ; in parent zone
```