DNSSEC

Geoff Huston AM
APNIC

What's DNSSEC?

It’s the ability to add digital signatures to DNS responses.

```
$ dig +dnssec www.potaroo.net AAAA @127.0.0.1

; <<>> DiG 9.18.2 <<>> +dnssec www.potaroo.net AAAA @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36348
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: 037e7ff2970bd29801000000628b47b76d96ecc2d227fae5 (good)
;; QUESTION SECTION:
;www.potaroo.net.                    IN          AAAA

;; ANSWER SECTION:
www.potaroo.net.        6394         IN          AAAA          2401:2000:6660::108
www.potaroo.net.        6394         IN          RRSIG         AAAA 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.
W9CDfQ3nCl35ZuFCIxgz+Rl4f+L8O/RRpJLwpPVq6wMgP5CPpP8sSiQc ySCB5scLFBN5aeqG1/jOBeywVYfp0g==

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Mon May 23 18:37:11 AEST 2022
;; MSG SIZE  rcvd: 207
```

response

digital signature

So what?

- If the client can *validate* this digital signature, then it can be assured that:
  - The response the client received is **authentic** and **complete**
  - The response is **current**
- It doesn’t matter how the DNS client learned this response: if the response correctly validates then the data is a genuine and complete extract from the authoritative zone file
- If it wasn’t signed and validated then you really can’t tell if the data has been altered by some intermediary

What about non-existence?

```
$ dig +dnssec www.potaroo.net TXT @127.0.0.1
```

Query for a non-existent record type

```
; <<>> DiG 9.18.2 <<>> +dnssec www.potaroo.net TXT @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32884
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: 660df460a7f4ed0101000000628b49a2d9c0f21dc394dd81 (good)
;; QUESTION SECTION:
;www.potaroo.net.                IN          TXT

;; AUTHORITY SECTION:
www.potaroo.net.        6400     IN          NSEC        _443._tcp.www.potaroo.net. A AAAA RRSIG NSEC
www.potaroo.net.        6400     IN          RRSIG       NSEC 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.
lhP13N+YR6m3dBYLUxfgv8fGsuiF4f14UcpznpyqlevlJyEumLgHtzUV Y6k6MXpiygGqI70KzZidqzAhglVCcQ==
potaroo.net.            6400     IN          SOA         ns1.potaroo.net. gih.potaroo.net. 2022032501 10800 3600 3600000 6400
potaroo.net.            6400     IN          RRSIG       SOA 13 2 6400 20320331235230 20220324225230 41284 potaroo.net.
oQZTmjoMBb8r8FUiHbp+62ZjSV1aXU9Gl6K28ngh6RXHFPWmzTJIilEA dCkf7fzA3d9ANqm5I5UiMikBRPceFw==

;; Query time: 143 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Mon May 23 18:45:22 AEST 2022
;; MSG SIZE  rcvd: 377
```

NODATA flags

Defined RRtypes for this label

NODATA response

NODATA signature

What about non-existence?

```
$ dig +dnssec _80_tcp.www.potaroo.net A @127.0.0.1
```

Query for a non-existent name

```
; <<>> DiG 9.18.2 <<>> +dnssec _80_tcp.www.potaroo.net A @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 47442
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: b6c04852ce47b40901000000628b4b20b3104cdb1bfb9e90 (good)
;; QUESTION SECTION:
;_80_tcp.www.potaroo.net.        IN        A

;; AUTHORITY SECTION:
potaroo.net.        6018    IN    SOA      ns1.potaroo.net. gih.potaroo.net. 2022032501 10800 3600 3600000 6400
potaroo.net.        6018    IN    RRSIG    SOA 13 2 6400 20320331235230 20220324225230 41284 potaroo.net.
oQZTmjoMBb8r8FUiHbp+62ZjSV1aXU9Gl6K28ngh6RXHFPWmzTJIilEA dCkf7fzA3d9ANqm5I5UiMikBRPceFw==
www.potaroo.net.    6018    IN    NSEC     _443._tcp.www.potaroo.net. A AAAA RRSIG NSEC
www.potaroo.net.    6018    IN    RRSIG    NSEC 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.
lhP13N+YR6m3dBYLUxfgv8fGsuiF4f14UcpznpyqIevlJyEumLgHtzUV Y6k6MXpiygGqI70KzZidqzAhglVCcQ==

;; Query time: 6 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Mon May 23 18:51:44 AEST 2022
;; MSG SIZE  rcvd: 396
```

There are no names between these two labels

NXDOMAIN digital signature

DNSSEC Design Basics

- DNSSEC does not alter the DNS in any way, nor does it alter the basic query/response protocol
- DNSSEC adds 5 new Resource Record Types:
  - RRSIG – the digital signature of a zone resource record
  - DNSKEY – the public key(s) used to ”sign” the zone
  - DS – the hash of the zones entry key, placed in the parent zone
  - NSEC – a spanning record used to sign across the “gaps” in a zone
  - NSEC3 – a variant of the NSEC spanning record used to sign across the “gaps” in a zone
- If a query sets the DNSSEC OK flag then the signature (RRSIG record) is added to the response (if one exists in the zone)

"Signing" a zone

- Generate a key pair
- Add the public key to the zone as a DNSKEY record
- Use the private key to generate signatures for all records in the zone (RRSIG records)
- Publish the signed zone file
- Pass the hash of the public key to the zone’s parent to publish as a DS record alongside the NS delegation records

For larger zones…

- Alternatively, the zone’s records can be signed when they are served
- This means that you don’t need to assemble the entire zone in one place to generate signature records in advance
- The signer sits after the ‘normal’ server and adds DNSSEC records to the response on the fly
- This approach is useful for large zones with constantly changing content

Why DNSSEC?

Because the DNS is a mess!

- DNS over UDP is a open (unencrypted) protocol that anyone on the wire can mess with!
- It’s unaccountable: Nobody really knows where your queries go
- It’s opaque: Nobody really knows where your answers come from
- It’s unverifiable: You can’t tell is the answer is “true” or not!
- It’s a massive privacy leak: Everything you do do starts with a DNS query which ties together your IP address and a query name

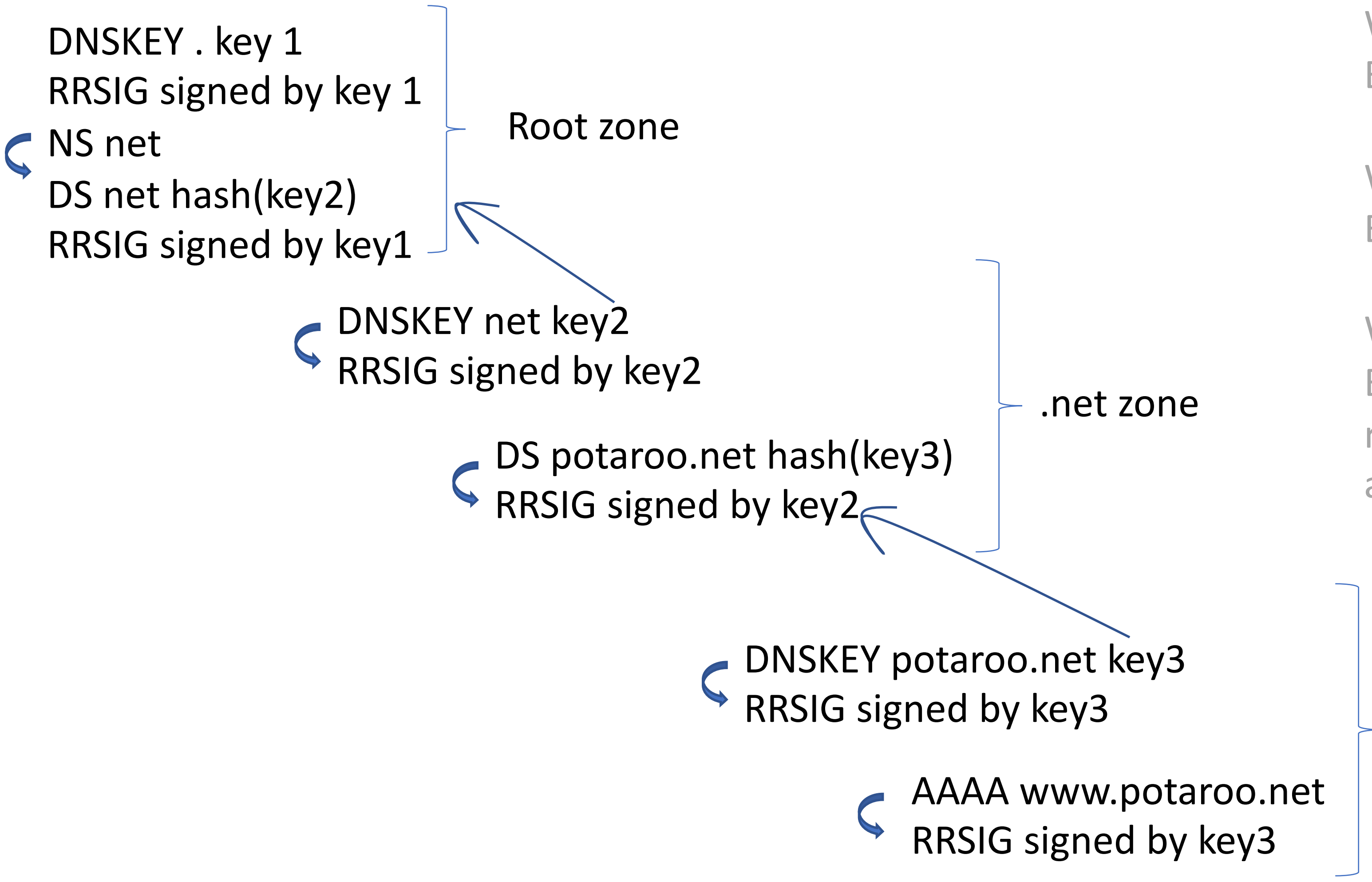
Exploiting DNS weakness

- DNS queries are intercepted and altered on a routine basis
- DNS rewriting is commonly used for various forms of censorship
- DNSSEC can’t stop such efforts to intercept and manipulate DNS queries and response
    - But it can withhold responses that fail to pass DNSSEC validation

How does DNSSEC do "Trust"?

- If we are talking “trust” when should we be talking X.509 public key certificates as well?
  - No, no X.509 certificate is needed or used in DNSSEC
- This entire process is based on the keys themselves
- Its strength lies in the transitive trust model of interlocking keys…

DNSSEC "key chains"



DNSKEY . key 1
RRSIG signed by key 1
NS net
DS net hash(key2)
RRSIG signed by key1

Root zone

DNSKEY net key2
RRSIG signed by key2

DS potaroo.net hash(key3)
RRSIG signed by key2

.net zone

DNSKEY potaroo.net key3
RRSIG signed by key3

AAAA www.potaroo.net
RRSIG signed by key3

potaroo.net zone

Why should I trust key 3?
Because key2 has signed over key3

Why should I trust key2?
Because key1 has signed over key2

Why should I trust key1?
Because I have been configured to regard the root key (key1) as a trust anchor

DNSSEC Validation

- Retrieve the zone signing key(s) for this zone (DNSKEY)
- Check that the signature matches the couplet of the RRdata and and the zone key
- Check that the signature of the DNSKEY record matches the couplet of the RRdata and and the zone key
- So if I trust the zone key, then I can trust this record
- Why should I trust the zone key (DNSKEY)?
- Query the zone parent for the Delegation Signer (DS) record
- Validate the signature of the DS record in the parent zone
- Repeat for the parent zone

- Once you get to the root zone check that the key you have retrieved from the DNS matches the root zone key that you have pre-loaded as your single trust point

DNSSEC Validation

- This is like Authoritative Server discovery in the DNS, but in reverse
- At each level the client retrieves the DS and DNSKEY resource records and then moves UP a level to the parent zone
- Until it reaches the root zone

- Then if performs the sequence of crypto operations to validate the chain of signatures

DNSSEC Validation can be slow

- For each level the validating client needs to retrieve the DNSSEC-signed DS and DNSKEY records
- For each record the validating client needs to perform a crypto validation operation
- E.g. for www.potaroo.net that’s 5 additional DNS queries and 6 crypto operations:
  - DNSKEY potaroo.net @ns1.potaroo.net
  - DS potaroo.net @a.gtld-servers.net
  - DNSKEY net @a.gtld-servers.net
  - DS net @a.root-servers.net
  - DNSKEY . @a.root-servers.net

DNSSEC Validation can be slow

```
$ dig +dnssec +bufsize=1232 DNSKEY au @2a01:8840:bf::1
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.16.27 <<>> +dnssec +bufsize DNSKEY au @2a01:8840:bf::1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22246
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
;; QUESTION SECTION:
;au.                                           IN          DNSKEY

;; ANSWER SECTION:
au.                                  43200     IN          DNSKEY     256 3 8 AwEAAbFKG7+4ErwxorDty/DvZbdzQ4/jVPqvSCKTr4oAOwJ+xFy747Bb …
au.                                  43200     IN          DNSKEY     257 3 8 AwEAAZvV7K54lJPnZUPiOxhY7nNiQ8/V0xSgCHyRxXLWTZGr56TF9gYJ …
au.                                  43200     IN          RRSIG      DNSKEY 8 1 43200 20220714000000 20220601223000 34882 au. mg/xwPs …


;; Query time: 36 msec
;; SERVER: 2a01:8840:bf::1#53(2a01:8840:bf::1)
;; WHEN: Sun Jun 05 23:28:51 UTC 2022
;; MSG SIZE  rcvd: 1385
```

Query using iPv6 with the UDP buffer size set to the current recommended value of 1232

The large response cannot fit in UDP and the query is retried using TCP, adding 2 additional round trip intervals to the time to complete the DNS response

How "good" is DNSSEC?

- Like all crypto, the choice of crypto algorithms to use to generate keys and signatures is crucial
- RSA is fast to use, but it has a low crypto strength, so crypto strength is achieved by using longer RSA keys
- Elliptical Curves are ”denser” – slower to use, but have a higher crypto strength for a given key size
- DNS over UDP prefers smaller keys!

Crypto Strength

Algorithm	Private Key size	Public Key Size	Signature Size	Strength Equivalence
RSA -1024	1,102	438	259	80
RSA-2048	1,776	620	403	112
RSA-4096	3,3112	967	744	140
ECDSA P-256	187	353	146	128
Ed25519	179	300	146	128

Care and Feeding Instructions

- No key can be used forever, so you need to “roll” your keys on a regular basis
  - We have to use the ‘old’ key to ‘introduce’ the ‘new’ key by getting the old key to sign across the new key in the DNSKEY records
  - Then we can discard the ‘old’ key
- There is no key revocation mechanism in DNSSEC
  - Compromised keys must be vacated quickly
  - Reverting to an unsigned state then loading the new key is an option here

Is DNSSEC worth the effort?

The case for “**Yes**”

- Too Much Blind Trust. We are trusting that the DNS mapping of the name to an IP address is genuine, trusting that the routing system is passing the IP packets to the ‘correct’ endpoint, trusting that the representation of the name on your screen is actually the name of the service you intended to go to, trusting that the TLS connection is genuine, and trusting that the WEB PKI is not corrupted, to name but a few critical points of trust
- We really have no alternatives – we have no other way of securing the DNS content
- The DNS is central – if an attacker can corrupt the DNS at will, then many other kinds of attacks are possible as a consequence

Is DNSSEC Worth the effort?

The case for “**No**!”

- Its One More Thing to go wrong
    - It adds the tasks of secure key management, regular key rotation, synchronisation with the parent zone
- DNS Responses are larger
    - All responses include a digital signature
    - DNSKEY responses include the entire key set plus the digital signature
    - DNS over UDP has reliability issues with large responses
        - UDP fragmentation for large responses is unreliable
        - TCP failover on truncated responses is unreliable
- Validation takes additional time
    - The validator must separately query for DS and DNSKEY records up the delegation chain and then perform a sequence of crypto operations

Is DNSSEC Worth the effort?

The case for “**No**!”

- Sub resolvers generally don’t validate responses anyway!
  - They rely on the AD bit being set in the response from the recursive resolver
  - Which defeats the entire purpose of DNSSEC!! Its crazy!
- Signalling DNSSEC validation failure is extremely badly handled in the DNS
  - There was no defined DNSSEC validation error code, so the standard reused the SERVFAIL error code
  - SERVFAIL as a response code triggers an exhaustive search across **all** servers
- What’s the realistic assessment of threat?
  - As a result, the only threat that DNSSEC protects the stub against is tampering with the response sent from the Authoritative server to the Recursive resolver, which is a pretty abstract threat model

If not DNSSEC, then what?

- Nobody “important” seems to be signing here in .au
    - No gov.au records
        - Not even AFP or ASD!
        - Not the federal shop front (my.gov.au), nor the ATO
    - No major retail banks
    - More generally, few folk DNSSEC-sign their DNS names in .au
- Instead, they are trusting that TLS is robust
    - TLS relies on the certificate infrastructure of the web PKI
    - So they are trusting that the web PKI is robust
- And this is a problem

PKIs have problems too!

https://www.feistyduck.com/ssl-tls-and-pki-history/

- The problem here is that with so many points of trust and no easy way of limiting the trust domain each client is forced to trust every single CA that all of its actions are absolutely correct all of the time
- Every CA in the PKI simply must never lie
- Which is an impossible objective
- And we have no robust *certificate revocation* mechanism to “unsay” dud certificates

Certificates are a Failure?

- We persist with long-lived certificates and non-functional revocation mechanisms, because it’s the path of least resistance
- The problem with certificates that provide a trust window of a few hours, is that the existing CA infrastructure and the use models of locally stashed certificates just can’t cope with such an increased intensity of certificate re-issuance.
- If certificates are incapable of informing a client that they are about to be drawn into misplaced trust then what exactly are they good for anyway?
- The entire objective here was to answer the simple question: “**Is the service that I am about to connect to the service that I intended to connect to?**” And the problem is that this entire certificate structure can only answer a question that relates to the past, not the present!

Where to from here?

- We’ve been trying to patch up the PKI system for some decades, and the result is a system that is not much more robust, but now has a greater level of external dependencies/vulnerabilities
- DNSSEC could be a more robust approach here but adoption resistance and operational immaturity count heavily against it
  - It’s not a clear and useful “solution” to a current set of opsec issues
- But doing nothing seems to be irresponsible as well!

Where to from here?

I really don’t know!

- I don’t think we can “fix” the certificate system
  - Too many points of trust create vulnerabilities for the entire system
  - Revocation is broken so mis-issuance and key compromise create persistent vulnerabilities
- But there is a lot of resistance to DNSSEC
  - The single point of all trust is not at all reassuring
  - Validation is too slow and too fragile
  - Key management is fragile