CLOUDFLARE®

DNSSEC at Internet scale

Ólafur Guðmundsson
olafur at cloudflare.com
@Oaudm

We're on a mission to build a better web

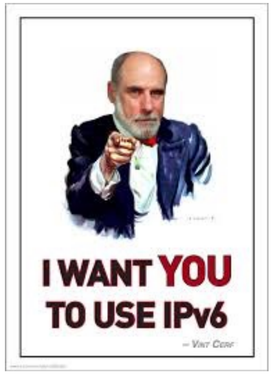
- Deliver the Internet faster, better and safer
- Take every aspect our mission seriously
- Challenge standard practices
- write blogs about what we are doing/discovering
- Use lots of open source and contribute back

- We are hiring

CloudFlare Scale

- Speed (Gb/s)
- Sites: 74+ and growing
- Servers in production: <secret>
- Domains: Millions and growing
- People: 250+
- Traffic: Almost 10% of http/https connections
- Routing: Anycast everything

Innovative



- IPv6 for all
- Universal SSL
- Universal DNSSEC
- HTTP2
- Inexpensive

DNSSEC overview

DNSSEC: Signing

- What is signed ?
  - The name, type, class, TTL and content along with two timers and by what key
    - Signer is always the zone the record resides in
    - Timers set when the signature becomes valid and when it expires: common case signatures valid for 2-4 weeks
- When is it signed
  - Signatures need to be refreshed periodically
- Signature expiration only affects the RRset not any RRset that is subservient of that RRset, i.e. if things are valid at the point of validation it is ok to use and cache for TTL.

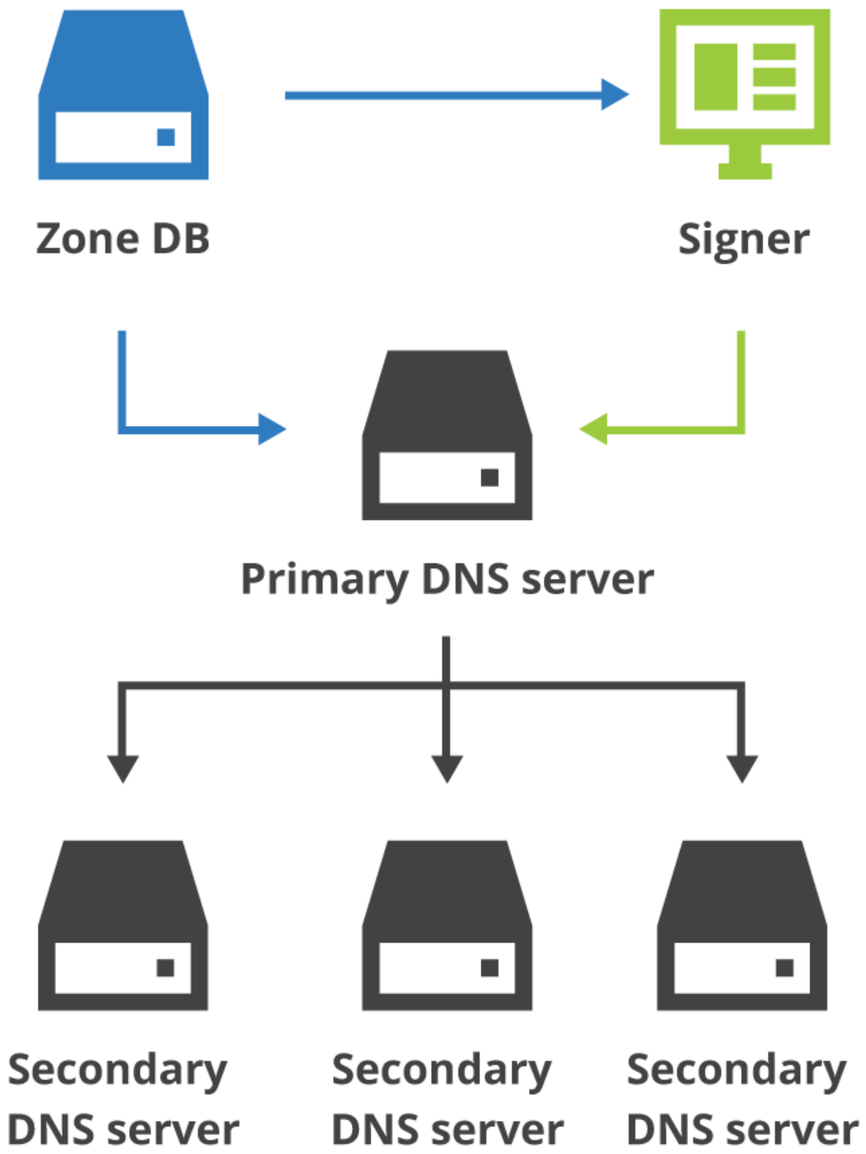
DNSSEC: Why?

- Answers are protected against forgery

- Enables new things like DANE, and other keying protocols

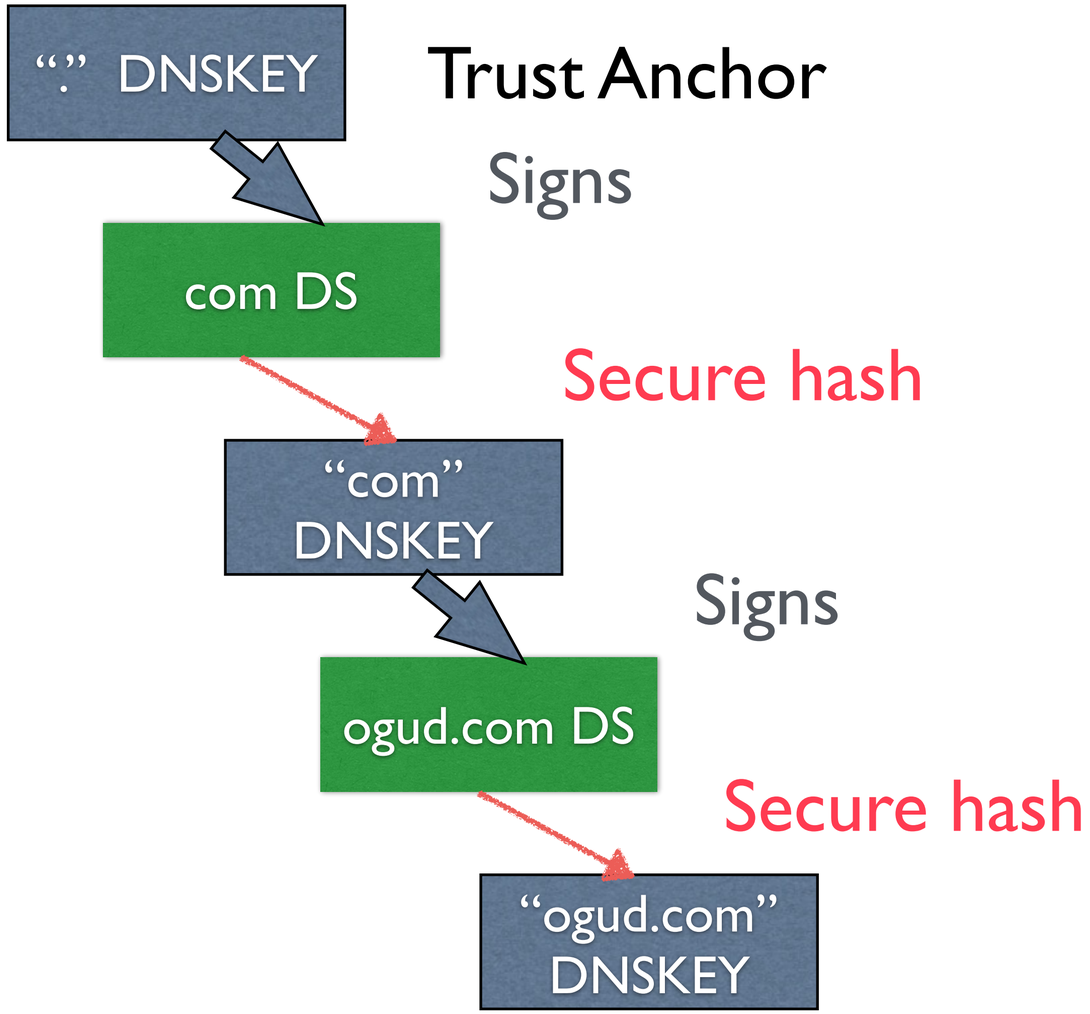
DNSSEC: How

- Digital signing
  - RSA and ECDSA keys
  - zones are signed in central location and distributed to DNS servers
- Answers
  - There is always a signed answer or a signed proof that answer does not exist
  - Negative Proof can either prove type does not exist or name does not exist



DNSSEC ==> DNS-Authentication

- Goal: Deliver tamper evident answers
- Digital Signatures added to RRSet
- Trust hierarchy established by series of DNSKEY and DS records
  - DNSKEY == Public KEY
  - DS == Hash (domain name I DNSKEY)
- Explicit denial of existence

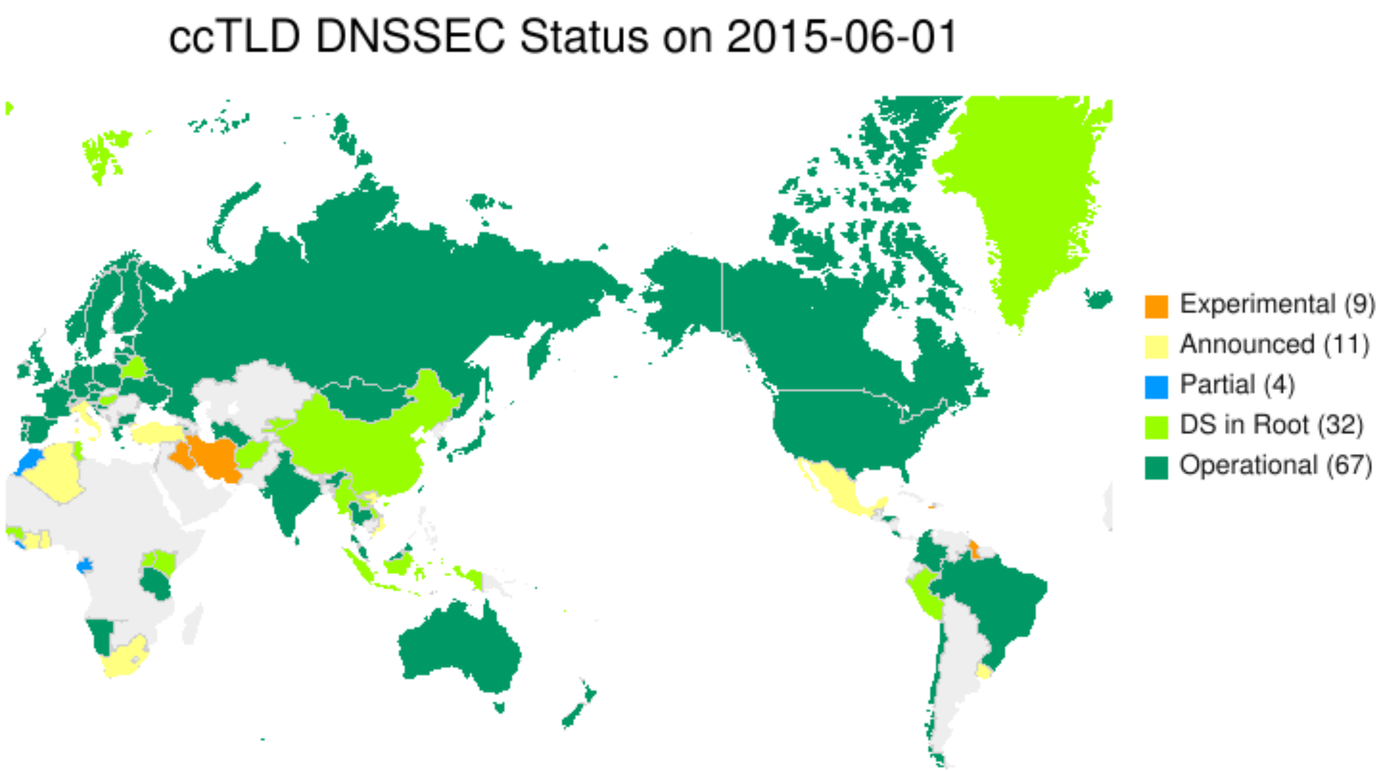


DNSSEC: Key management

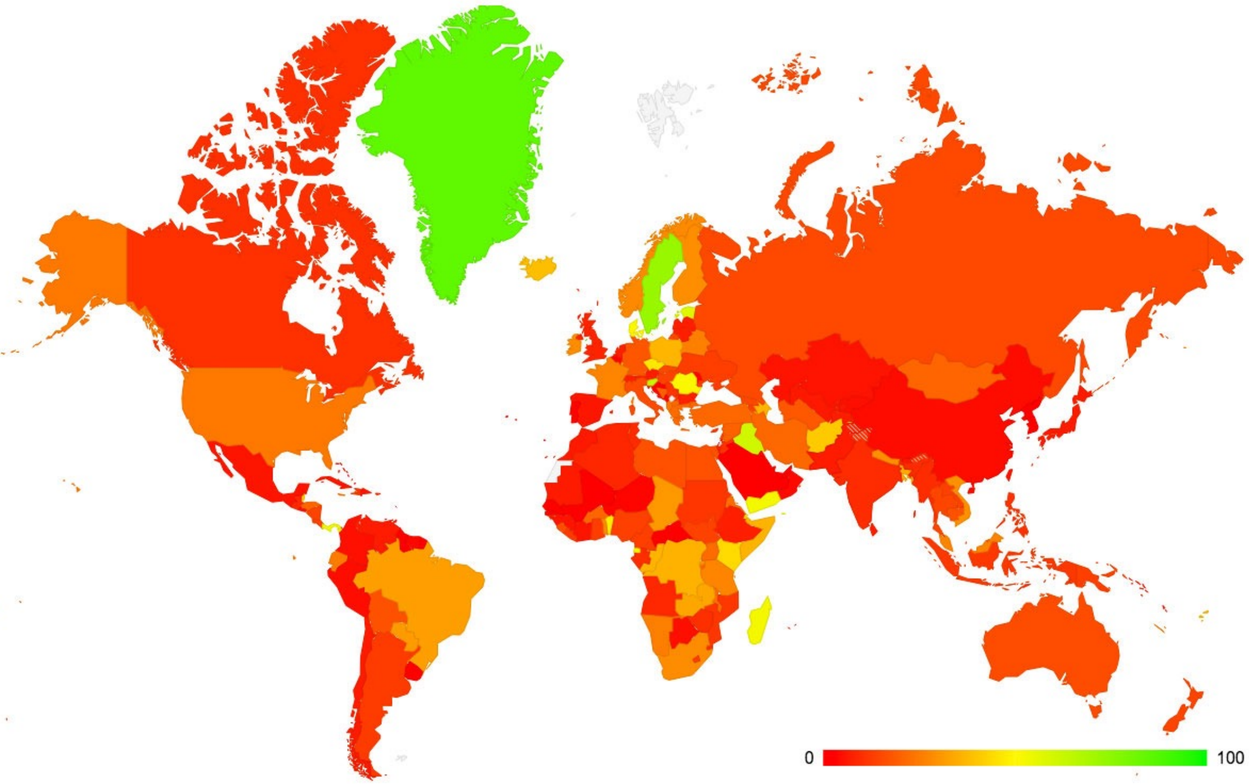
- Each zone is responsible for its own key operations
- Each zone needs to have parent publish DS record reflecting the key that signs its DNSKEY set.
- Two key roles defined
  - Key signing key i.e. trust anchor for the zone
  - Zone signing key signs most of the data
- Note: same key can fill both roles, depends on operators policy

DNSSEC deployment status #1

- Software
  - Most DNS servers support
  - Most DNS resolvers support
- Publishing side
  - root is signed
  - 70% of TLD’s signed
  - millions of domains signed
- Not all registrars/TLD’s accept (all) DS records



DNSSEC Deployment status #2



- Few high value Domains are signed
  - chicken and egg problem

Over 25% of DNS queries go through DNSSEC validators why not 100%

- Old software not updated
- Trust anchor not enabled
- Operators worried being blamed for publishers mistakes
- Google Public DNS and Verisign Public DNS both validate.

DNSSEC: Implications

- Negatives
  - Zones and Answers get bigger, all these RRSIG's NSEC/NSEC3 and DNSKEY
  - Validation requires more lookups: find DNSKEY records
  - Operators have more ways to make mistakes! Time becomes a factor in DNS
- Positives
  - Answers can be trusted i.e. forging answers is MUCH harder
    - attackers can deny service discovery but not redirect traffic
  - DNS becomes alternative to publish information that was previously looked up over HTTPS
  - Forced lots of cleanup in implementation and operations that have improved DNS

CloudFlare DNS

CloudFlare DNS

- RRDNS is our in-house DNS server written in Go
- Resilient against attacks and abuse
- High performance and great geographical reach
  ==> fast answers (top 3 worldwide in response time)
- Operate over 2M domains for our customers
- We care a lot about answer size

CloudFlare DNS is different

- KV store distributes DNS information to edges
- Some data changes a frequently
- domains added and removed all the time
- Some answers calculated on the fly
  - Geo rules
  - Fetch answers from multiple sources
- Not all data is needed everywhere

DNS DDoS

- We are seeing DDoS attacks using DNS almost all the time.
- Size ranges from few hundred pps to high tens of million pps
- Types of attacks: Direct floods, reflections
- Mitigations:
  - Scale and Anycast
  - Smart tools that scrub traffic before it hits our servers
  - Give actually resolvers answers that help them mitigate attacks
  - small answers

Response rate

- On average we answer less than 1 in 100 DNS packets

- None complains about our lack of responses

CloudFlare DNSSEC

CloudFlare DNSSEC: Answers

- Sign answers on the fly using ECDSA
  - Smaller signatures, stronger algorithm, quicker to sign answers, verification slower than RSA
    - ECDSAP256 => 3000bit RSA
    - 512 bits vs 3000
  - We improved Google Go crypto library to be 21x faster!!! (code is open source)
  - Common DNSKEY answer shrinks from almost 1200 bytes to about 300!!

```
ietf.org.               1800    IN      DNSKEY  256 3 5 AwEAAdDECajHaTjfSoNTY58WcBah1BxPKVIHBz4IfLjfqMvium4lgKtK ZLe97DgJ5/NQrNEGGQmr6fKv
Uj67cfrZUojZ2cGRizVhgkOqZ9scaTVX NuXLM5Tw7VWOVIceeXAuuH2mPIiEV6MhJYUsW6dvmNsJ4XwCgNgroAmX hoMEiWEjBB+wjYZQ5GtZHBFKVXACSWTiCtddHcueOeSVPi5
WH94Vlubh HfiytNPZLrObhUCHT6k0tNE6phLoHnXWU+6vpsY pz6GhMw/R9BFxW5Pd PFIWBgoWk2/XFVRSKG9Lr 61b2z1R126xeUwvw46RVy3hanV3vNO7LM5H niqaYclBbhk=
ietf.org.               1800    IN      DNSKEY  257 3 5 AwEAAa                                  BRfqxz9p/sZ+8AByqyFHLdZc HoOGF7CgB5OKYMvGOgysuYQl
oPlwbq7Ws5WywbutbXyG24lMWy4jijlJ UsaFrS5EvUu4ydmu Rc/TGnEXnN1XQRO                                 a Jdj1cKr2nX1NrmMRowIu3DIVtGbQJmzpukpDVZaYMMAm8M5
vz4U2vRCV ETLgDoQ7rhsiD127J8gVExjO8B0113jCajbFRcM tUtFIjH4z7iXP2ZzD cXsgpe4LYEuenEQAcRBRl                E6oaykHR7rlPqqmw58nIELJUFoMcb/BdRLg byTeurFlnxs=
ietf.org.               1800    IN      RRSIG   DNSKE                                           43650 45586 ietf.org. dpOO1u/mE0ZmcergtT4RA5DdV8E
i3nTYvsuTFKqEou4Smku5UpO1giVp sOpdDRwvei5g2HC8VK/ nKHDncotNR2unaWRvA3yhNgNrGAGSpXh3R0h2 Io/7yDr2TK529YHee0MTVeHqk6YeyyiFvCL1XMLt3jj4/G3pjo
z7mS8M NLgysKQMEZqJHfZhARZeSNIuK/QpRJhBX9UQYrv6IJ/2l5WqdL6C6aeB fYe+bhn3G2s9apnUQFiq0xo3ybyQJmO6UEPjuEnn8uLXnXTlRdthZbnY g5yZReSWb4jVYQKC
yX4Pnm09TtrpduZQqzl20v+8nMITf4HJnSj7EvPN AxmCXg==
```

```
filippo.io.             3600    IN      DNSKEY  257 3 13 DGpDkudNu/XQT1Km
QkXFtKCfZPxHGV07qSTIcDX                                 hiQVR53E69/E57IFm8b6Zw==
filippo.io.                             3600    IN      DNSKEY  256 3 13 koPbw9wmYZ7ggcjn
Q6ayHyhHaDNMYELKTqT+qRG                                 hii+sb0PYFkH1ruxLhe5g==
filippo.io.                             3600    IN      RRSIG   DNSKEY 13 2 3600 20150523
162528 20150422162528 42                                +grfGMuA2al/vQ9S5tBXOJq
ZbeTOYB0hfHG7Sl6hqR1 xfoibSJA1BiX5r9Ujo5YVU/NE1H0TQ==
```

RSA:
1181 BYTES

ECDSA:
305 BYTES

DNSSEC: Black Lies

- Smallest Negative answers with “Black Lies”
- Off-line signed DNSSEC zone needs 2-3 NSEC/NSEC3 records for non existing name proof
- We assert with one NSEC record that the name exists but the type requested does not

```
; <<>> DiG 9.10.3 <<>> doesnotexist.org.br spf +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 13680
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;doesnotexist.org.br.           IN      SPF

;; AUTHORITY SECTION:
8m7ro9ivs4akf6ib7leh8keikecd3135.org.br. 900 IN NSEC3 1 1 10 349C780F5F273482395
8m7ro9ivs4akf6ib7leh8keikecd3135.org.br. 900 IN RRSIG NSEC3 7 3 900 201512081000
66kivi3f210fgsof87bqlh5188miql9t.org.br. 900 IN NSEC3 1 1 10 349C780F5F273482395
66kivi3f210fgsof87bqlh5188miql9t.org.br. 900 IN RRSIG NSEC3 7 3 900 201512081000
0lmdufmov63o9dkeg1fi24cl7vv9lkf5.org.br. 900 IN NSEC3 1 1 10 349C780F5F273482395
0lmdufmov63o9dkeg1fi24cl7vv9lkf5.org.br. 900 IN RRSIG NSEC3 7 3 900 201512081000
org.br.                 900     IN      SOA     a.dns.br. hostmaster.registro.br. 2015120352
org.br.                 900     IN      RRSIG   SOA 7 2 172800

;; Query time: 139 msec
;; SERVER: 10.20.30.8#53(10.20.30.8)
;; WHEN: Thu Dec 03 08:04:15 EST 2015
;; MSG SIZE  rcvd: 1170

<wireles-b-117:/var/tmp 8:04 0> dig doesnotexist.cloudflare.com spf +dnssec | cut -c 1-80

; <<>> DiG 9.10.3 <<>> doesnotexist.cloudflare.com spf +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36097
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;doesnotexist.cloudflare.com.   IN      SPF

;; AUTHORITY SECTION:
cloudflare.com.         2632    IN      SOA     ns3.cloudflare.com. dns.cloudflare.com. 20200
cloudflare.com.         85432   IN      RRSIG   SOA 13 2 86400 20151204134812 20151202114812
doesnotexist.cloudflare.com. 2632 IN    NSEC    \000.doesnotexist.cloudflare.com. RRSI
doesnotexist.cloudflare.com. 2632 IN    RRSIG   NSEC 13 3 3600 20151204134812 2015120

;; Query time: 3 msec
;; SERVER: 10.20.30.8#53(10.20.30.8)
;; WHEN: Thu Dec 03 08:04:20 EST 2015
;; MSG SIZE  rcvd: 371
```

1170 Bytes

371 bytes

Negative Answers: “Black Lies” detail

- True lie: Sign a NOERROR.
- Generate NSEC for the query name, covers over minimal span, only set the RRSIG and NSEC bits ==> NXDOMAIN

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58166
;doesnotexist.cloudflare.com. IN   SSHFP

;; AUTHORITY SECTION:
cloudflare.com.    86400   IN SOA  ns3.cloudflare.com. dns.cloudflare.com. 2020120552 10000 2400 604800 3600
doesnotexist.cloudflare.com. 3600 IN NSEC \000.doesnotexist.cloudflare.com. RRSIG NSEC
cloudflare.com.    86400   IN   RRSIG   SOA 13 2 86400 20151211140218 20151209120218 35273 cloudflare.com. Yw3rDHg
+tp2jYCT9Xcr14GxZ4WDTfV4aDHSpfQgW3t3NR33FwjQ+5nOY r0bhu5BN4Cm4v9OR4LxHz94VLYUHWg==
doesnotexist.cloudflare.com. 3600 IN RRSIG   NSEC 13 3 3600 20151211140218 20151209120218 35273 cloudflare.com.
bUJDOb3h2VnCP+lyC6rXTmDVqNmqchx8m/yZJt2w/14Ii/PIHPXiIw1m 204mgg3uL2jlE5NPad1IGDW0b4fgLw==
;; MSG SIZE  rcvd: 371
```

Negative answers: “The NSEC shotgun”

- RRDNS is optimized for answering exact query
- Query for TXT and there is no TXT?
  - Set many of bits for types that might exist
- The NSEC is a valid denial for TXT, and useless for an attacker that wants to replay it for other queries

```
;cloudflare.com.      IN SSHFP

;; AUTHORITY SECTION:
cloudflare.com.   86400  IN SOA  ns3.cloudflare.com. dns.cloudflare.com. 2020120552 10000 2400 604800 3600
cloudflare.com.   3600 IN  NSEC \000.cloudflare.com. A NS SOA WKS HINFO MX TXT AAAA LOC SRV CERT IPSECKEY
RRSIG NSEC DNSKEY TLSA HIP CDS CDNSKEY OPENPGPKEY SPF
cloudflare.com.   86400  IN RRSIG  SOA 13 2 86400 20151211135723 20151209115723 35273 cloudflare.com. ua5+348YgFLGUghX0Qaw2Ng8XZ4U
+Y2TNe4kpqp95dWyzWu8grkYTmuu W/h+l9siXGlqAjaN4FcfbuJOQBBGgQ==
cloudflare.com.   3600 IN RRSIG  NSEC 13 2 3600 20151211135723 20151209115723 35273 cloudflare.com. 6j7lfYoK9+jCFZ17wqaSsDWJrK
+j6VnaLSF8qv/JxqvoMnxfauFXQjiA P7Py5YYNs670/0SlcTOflTQeF4HuOA==
```

Corner cases: ANY query

- Returns many RRsets
  - Resolvers return
    - what is in cache
    - or asks auth server
  - Auth servers frequently return as many RRsets as fit in answer
- Widely abused in particular on DNSSEC signed zones

ANY uses

1. Debugging: i.e. Human asks and parses answer
2. Probabilistic Optimization: trying to get one or more answers in one query, when answer is not useable the program then falls back on direct queries
3. Misunderstanding by a programmers
4. Amplification attacks

ANY == BAD

- Number of ANY attacks each week
- Lots of work: need to make a DB call for every type allowed in our setup
- Sign number of RRset's ==> expensive
- Answer is useless to many resolvers as credibility is set to lower than direct query ==>
  - followup query is sent to master if direct query for type is received (Unbound and Bind)

ANY: Solution

- We return one USELESS RRType,
  - and sign it when needed
  - based on draft-ietf-dnsop-restrict-any

```
; <<>> DiG 9.9.8 <<>> @ns5.cloudflare.com cloudflare.com any +dnssec
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24882
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;cloudflare.com.      IN ANY

;; ANSWER SECTION:
cloudflare.com.    3789 IN HINFO  "Please stop asking for ANY" "See draft-jabley-dnsop-refuse-any"
cloudflare.com.    3789 IN RRSIG  HINFO 13 2 3789 20151211134615 20151209114615 35273 cloudflare.com.
;; SERVER: 162.159.2.9#53(162.159.2.9)
;; WHEN: Thu Dec 10 07:46:15 EST 2015
;; MSG SIZE  rcvd: 226
```

CloudFlare DNSSEC: Key management

- Reuse keys for many customers, ==> does not decrease security but increases reliability a lot.
  - Fewer keys to distribute to our edge servers
  - Zone signing keys will be rolled on demand
- We sign DNSKEY (and CDS) records centrally in high assurance systems
  - Signatures are for a month at a time
- Key signing keys rollover is not planned until we do a algorithm roll, or change central signing systems

DNSSEC: The long tail

- Signing is the goal, Verification is
- Uploading DS records is hard
- Getting operators to Verify is hard
- Getting installed base to support new algorithms is REAL HARD
  - https://github.com/ogud/DNSSEC_ALG_Check

```
./alg_rep -r 200.160.11.114

Zone dnssec-test.org.  Qtype DNSKEY Resolver [200.160.11.114] debug=false verbose=false Prime= V
DS     :  1  2  3  4  |  1  2  3  4
ALGS   :    NSEC      |     NSEC3
alg-1  :  -  -  -  -  |  x  x  x  x
alg-3  :  V  V  V  V  |  x  x  x  x
alg-5  :  V  V  V  V  |  x  x  x  x
alg-6  :  x  x  x  x  |  V  V  V  V
alg-7  :  x  x  x  x  |  V  V  V  V
alg-8  :  V  V  V  V  |  V  V  V  V
alg-10 :  V  V  V  V  |  V  V  V  V
alg-12 :  V  V  V  V  |  V  V  V  V
alg-13 :  V  V  V  V  |  V  V  V  V
alg-14 :  V  V  V  V  |  V  V  V  V
V == Validates  - == Answer  x == Alg Not specified
T == Timeout S == ServFail O == Other Error
DS algs 1=SHA1 2=SHA2-256 3=GOST 4=SHA2-384
```

```
Zone dnssec-test.org.  Qtype DNSKEY Resolver [193.0.24.4] debug=false verbose=false Prime= V
DS     :  1  2  3  4  |  1  2  3  4
ALGS   :    NSEC      |     NSEC3
alg-1  :  V  V  -  -  |  x  x  x  x  => RSA-MD5 OBSOLETE
alg-3  :  V  V  -  -  |  x  x  x  x  => DSA/SHA1
alg-5  :  V  V  -  -  |  x  x  x  x  => RSA/SHA1
alg-6  :  x  x  x  x  |  V  V  -  -  => RSA-NSEC3-SHA1
alg-7  :  x  x  x  x  |  V  V  -  -  => DSA-NSEC3-SHA1
alg-8  :  V  V  -  -  |  V  V  -  -  => RSA-SHA256
alg-10 :  V  V  -  -  |  V  V  -  -  => RSA-SHA512
alg-12 :  -  -  -  -  |  -  -  -  -  => GOST-ECC
alg-13 :  -  -  -  -  |  -  -  -  -  => ECDSAP256SHA256
alg-14 :  -  -  -  -  |  -  -  -  -  => ECDSAP384SHA384
V == Validates  - == Answer  x == Alg Not specified
T == Timeout S == ServFail O == Other Error
DS algs 1=SHA1 2=SHA2-256 3=GOST 4=SHA2-384
```

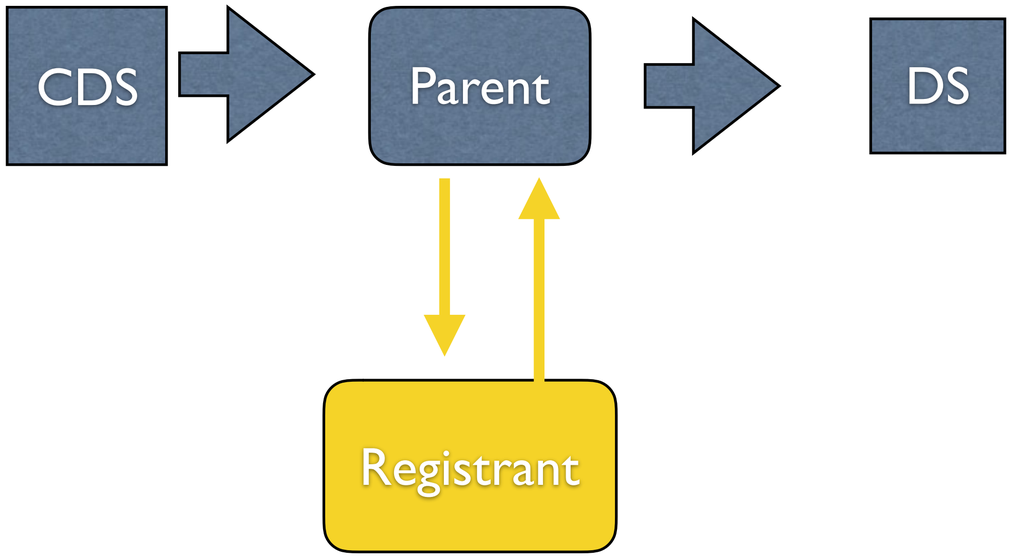
Automating DS : why

- Registrants need to upload DS via user interfaces
- NIC.BR registrar customers and using CloudFlare for DNS got email from NIC.br
  - In 24 hours over 1000 .br domains added DNSSEC and uploaded DS
  - This is great in spite of what they had to do
- A: read email
- B: log into CloudFlare account
- C: enable DNSSEC
- D: Login to Nic.Br registar
- E: follow instructions on how to upload DS records from email
- Estimated time
  - 10 minutes first domain
  - 4 minutes per subsequent one

Automating DS: Scope

- We publish CDS or CDNSKEY records for all signed domains
- Policy statement: “A domain publishing a CDS record is signaling to the world that it wants to be validated”
- Need mechanism to trigger parent to check CDS and add/update DS

Automating DS: How



- Proposing simple REST interface for parental agents to accept request
  - https://tools.ietf.org/html/draft-latour-dnsoperator-to-rrr-protocol
- Need registrars, registries to agree and add interfaces
  - Experimentation starting in .ca, .cl soon [.br????]
  - Experimentation with registrars
- User only needs to log into

What makes DNSSEC operations hard

- Anything that can not be automated or controlled by domain operator
- Understanding of the nuances of DNS protocol
- Timing, Timing, Timing
  - Information lives in caches after authoritative servers get updated
  - Performing certain actions may invalidate information
  - Enabling/Disabling DNSSEC should be done in right sequence of steps

TTL at parent: the hostage taker

- How long does it take globally
  - move operation of com.br domain from one set of NameServer’s to a different ones?
  - disable DNSSEC validation for a com.br domain ?
  - Perform a full KSK roll for a com.br domain?

TTL at parent: answers

- MAX(child NS TTL, 1 DAY)
- 1 Hour
- Two answers
  - DUAL DS (minimum)
    - 1 hour (add DS) + 1 DNSKEY TTL + 1 hour (delete DS)
  - Dual KSK
    - 1 DNSKEY TTL (add) + 1 hour (add) + max(1 DNSKEY TTL , 1 hour)

Questions ?