

LISA'13

27th Large Installation System Administration Conference

NOVEMBER 3-8, 2013
WASHINGTON, D.C.

Sponsored by USENIX
in cooperation with LOPSA

usenix



DNSSEC Tutorial

Shumon Huque
University of Pennsylvania

USENIX LISA Conference
Washington, DC, November 3rd 2013



1

DNSSEC Tutorial

© 2013,2014 Shumon Huque.

This tutorial is being presented at the LISA 2013 Conference held in Washington, DC on Nov 3rd 2013.

Feedback, critique, suggestions on these slides gladly received at <shuque @ upenn.edu>

Reminder: Please fill out the evaluation forms for this course!

Course blurb from LISA conference brochure:

This class will provide system administrators with a detailed understanding of the DNS Security Extensions (DNSSEC). It will provide practical information about configuring DNSSEC using the popular ISC BIND DNS software and will cover both using DNSSEC to cryptographically sign your own DNS zones and configuring DNS resolvers to validate DNSSEC signatures. Many examples of DNS/DNSSEC querying and debugging using the "dig" tool and other diagnostic tools and programs will also be covered. The last part of the course will cover prospects for newer and more exciting uses of the DNSSEC by application protocols that are in the pipeline, such as DANE and TLSA records.

[DNSSEC Tutorial, USENIX LISA 13]

3


Who am I?

- An I.T. Director at the University of Pennsylvania
- Have also been:
 - Programmer (C, Perl, Python, Lisp)
 - UNIX Systems Administrator
 - Network Engineer
- Education: B.S. and M.S. (Computer Science) from Penn
- Also teach a Lab course on Network Protocols at Penn's School of Engineering & Applied Science

[DNSSEC Tutorial, USENIX LISA 13]

4

Who am I?

- Website: <http://www.huque.com/~shuque/>
- Blog: <http://blog.huque.com/>
- Twitter: <https://twitter.com/shuque>  @shuque
- Google Plus: <https://plus.google.com/+ShumonHuque>

Course Topics

1. DNSSEC Tutorial
2. Live queries using 'dig'
3. Configuring DNSSEC in BIND
4. Application uses of DNSSEC
5. DNSSEC deployment status

DNSSEC Tutorial

[DNSSEC Tutorial, USENIX LISA 13]

7

For review

- Assuming familiarity with the basic DNS protocol
- If you need a review, a detailed treatment is provided in Appendix A of this slide deck
- However, we'll repeat a few main slides from that section now just to make sure everyone's on the same page ...

[DNSSEC Tutorial, USENIX LISA 13]

8

DNS

- Domain Name System
- Base specs in RFC 1034 & 1035 (obs 882 & 883)
- Distributed global database
- Indexed by “domain names” (together with a type and class)
- A domain name is a sequence of labels, eg.
 - www.amazon.com.
- Domain Names are case insensitive, but case preserving
- Transport protocol: UDP **and** TCP port 53

[DNSSEC Tutorial, USENIX LISA 13]

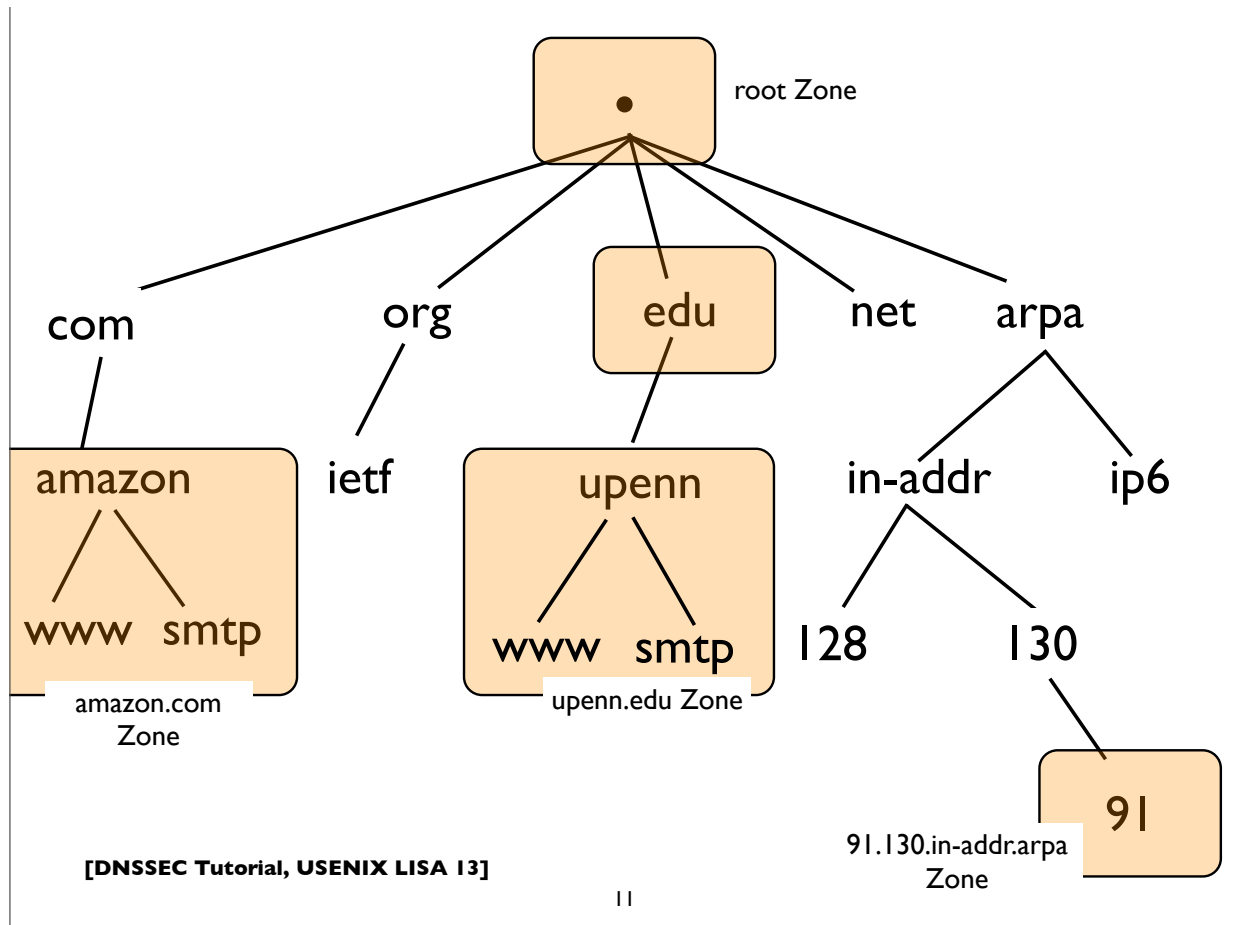
9

DNS

- DNS can be represented as a tree of labels
- Sibling nodes must have unique labels
- Domain name at a particular label can be formed by the sequence of labels traversed by walking up the tree from that label to the root
- Zone - autonomously managed subtree
- Delegations: boundaries between zones

[DNSSEC Tutorial, USENIX LISA 13]

10



DNS main components

- Server Side:
 - Authoritative Servers
 - Resolvers (Recursive Resolvers)
- Client Side:
 - Stub resolvers (usually on DNS client machines)

Authoritative Server

- A server that directly serves data for a particular zone
- Said to be “authoritative” for that zone
- These servers are the ones specified in NS records

[DNSSEC Tutorial, USENIX LISA 13]

13

Resolver

- Aka “Recursive Resolver”, “Cache” etc
- Used by endsystems (stub resolvers) to query (“resolve”) arbitrary domain names
- Receives “recursive” queries from these endsystems
- Resolvers query authoritative servers, following DNS delegations until they obtain the answer they need (this process is called “iterative” resolution)
- Resolvers “cache” (remember) query results for the specified “TTL” (also some negative results are cached)

[DNSSEC Tutorial, USENIX LISA 13]

14

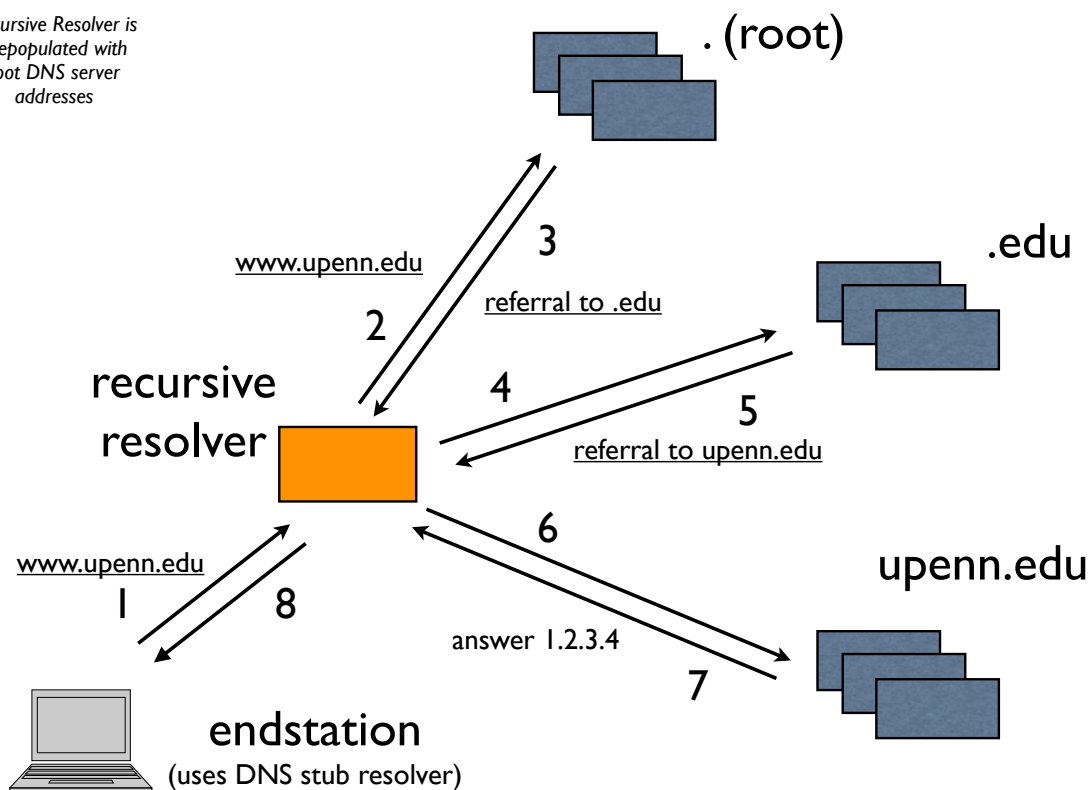
Stub Resolver

- The DNS client software component that resides on most endsystems
- Commonly implemented by the Operating System as a set of library routines
- Has a configured set of addresses of the Recursive Resolvers that should be used to lookup (“resolve”) domain names
 - usually by manual configuration, or dynamically learned via DHCP
- Some stub resolvers also cache results

[DNSSEC Tutorial, USENIX LISA 13]

15

*Recursive Resolver is
prepopulated with
root DNS server
addresses*



[DNSSEC Tutorial, USENIX LISA 13]

16

Parts of a DNS query

- Each DNS query needs a query name, type, and class
- **qname**: a domain name, eg. www.upenn.edu
- **qtype**: A, AAAA, MX, CNAME, PTR, SRV, TXT, NS, SOA, ...
- **qclass**: IN, CH, HS (only “IN” is commonly used)
- Various flags: QR, RD, EDNS Opt, DO etc

[DNSSEC Tutorial, USENIX LISA 13]

17

Resource Records (RR)

- The fundamental unit of data in the DNS database
- A grouping of a {domain name, type, class}, a TTL (time-to-live), and the associated “resource data”
- Has a defined text “presentation format”

```
www.example.com.      86400 IN   A   10.253.12.7
```

*name, or
owner name* *tll* *class* *type* *rdata*

[DNSSEC Tutorial, USENIX LISA 13]

18

Resource Record Sets

- A set of RRs with the same name, class, and type
- The rdata (resource data) associated with each RR in the set must be distinct
- The TTL of all RRs in the set also must match
- RR sets are treated atomically when returning responses

```
www.ucla.edu.      300   IN    A     169.232.33.224
www.ucla.edu.      300   IN    A     169.232.55.224
www.ucla.edu.      300   IN    A     169.232.56.224
```

[DNSSEC Tutorial, USENIX LISA 13]

19

Back to DNSSEC

[DNSSEC Tutorial, USENIX LISA 13]

20

DNSSEC at a glance

- “DNS Security Extensions”
- A system to verify the authenticity of DNS “data” using public key signatures
 - Specs: RFC 4033, 4034, 4035, 5155 (and more)
- Helps detect DNS spoofing, misdirection, cache poisoning ..
- Recall the “Kaminsky attack”
- Additional benefits:
 - Ability to store and use cryptographic keying material in the DNS, eg. SSHFP, IPSECKEY, CERT, DKIM, TLSA, etc ..

[DNSSEC Tutorial, USENIX LISA 13]

21

DNSSEC at a glance

- Each zone has a public and private key pair
- The zone owner uses the private key to sign the zone data, producing digital signatures for each resource record set
- Public key is used by others (DNS resolvers) to validate the signatures (proof of authenticity)
- Public key is published in the zone itself so that resolvers can find it
- Zone public keys are organized in a chain of trust following the normal DNS delegation path
- DNS resolvers authenticate DNS signatures from root to leaf zone containing name.

[DNSSEC Tutorial, USENIX LISA 13]

22

DNSSEC Records

DNSKEY	Contains zone public key
RRSIG	Contains DNSSEC signature
NSEC	Points to next name in zone (used for authenticated denial of existence)
DS	Delegation Signer (certifies public key for subordinate zone)
NSEC3	Enhanced version of NSEC (provides zone enumeration protection and opt-out)
NSEC3PARAM	NSEC3 parameters

[DNSSEC Tutorial, USENIX LISA 13]

23

Signed zone additions

- One or more DNSKEY at the zone apex
- One or more NSEC for every DNS name
- One or more RRSIG for every RR set
- One or more DS records for every secure delegation

- Exceptions: non-authoritative data like delegation NS records and glue have no signatures (RRSIG)

[DNSSEC Tutorial, USENIX LISA 13]

24

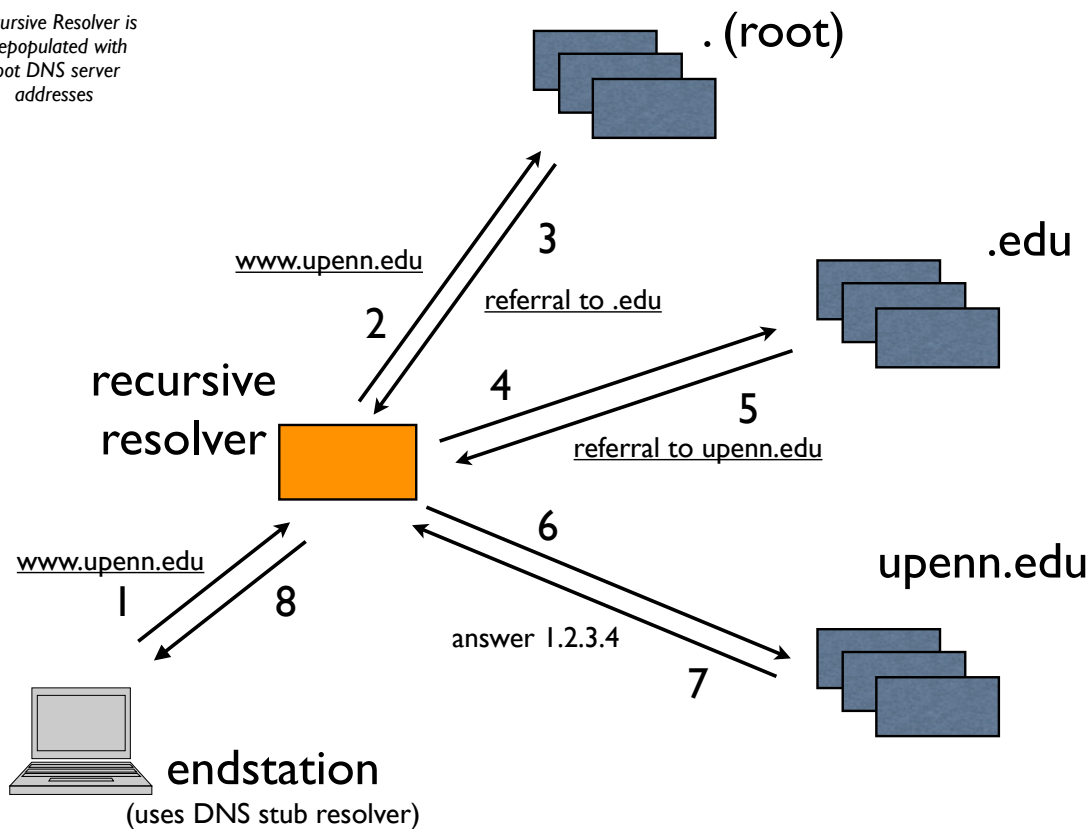
Gory details ...

- RFC 4033: DNSSEC Introduction
- RFC 4034: Resource Records for DNSSEC
- RFC 4035: DNSSEC - Protocol modifications
- RFC 5155: Hashed Authenticated Denial of Existence (NSEC3)
- RFC 6781: DNSSEC Operational Practices
- RFC 6840: Clarifications & Implementation Notes for DNSSEC
- (and a few other related ones ...)

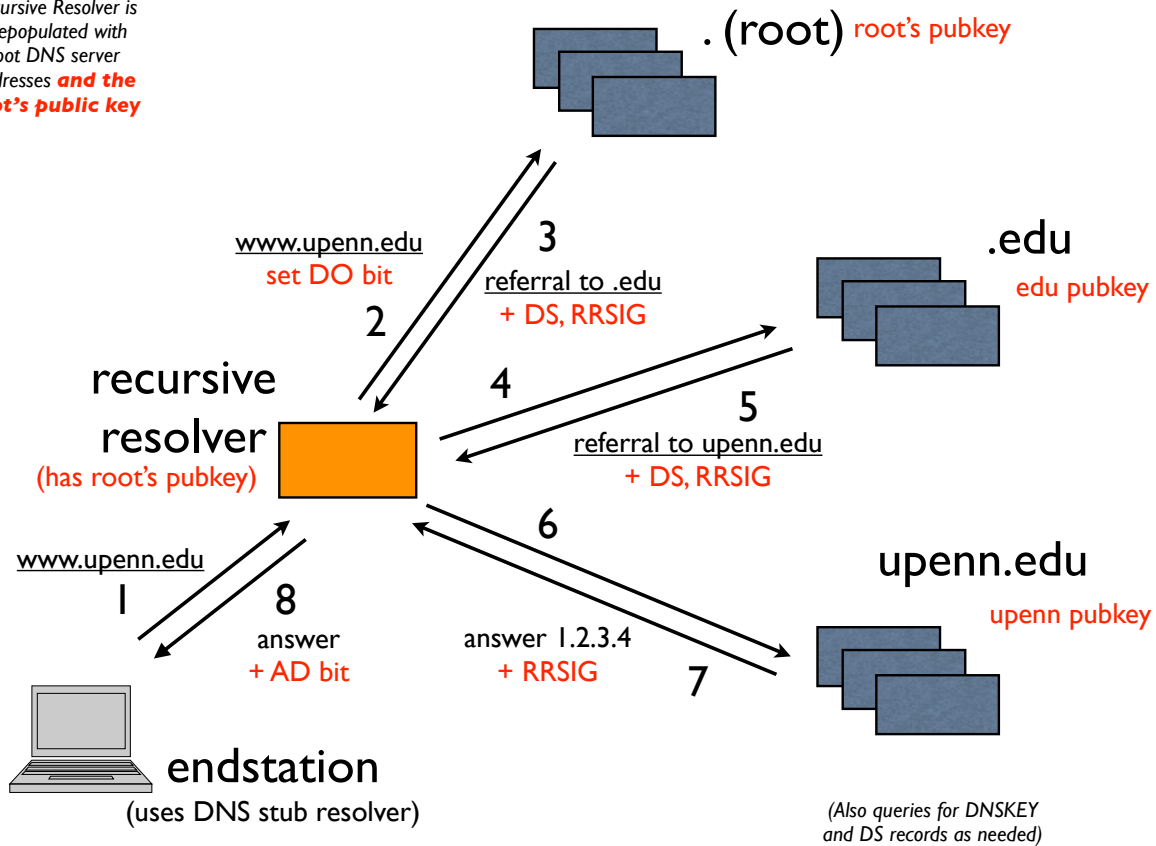
[DNSSEC Tutorial, USENIX LISA 13]

25

*Recursive Resolver is
prepopulated with
root DNS server
addresses*



Recursive Resolver is prepopulated with root DNS server addresses **and the root's public key**



EDNS0

- DNS messages larger than 512 bytes requires:
 - Use of TCP (typically truncated UDP response followed by TCP retry)
 - EDNS0 - a DNS extension mechanism allowing negotiation of larger UDP message buffers
 - RFC 6891 "Extension Mechanisms for DNS (EDNS0)"
- For DNSSEC, EDNS0 does:
 - Negotiation of larger UDP payload sizes
 - Flag to indicate querier is able to process DNSSEC records: the **"DNSSEC OK" or "DO" bit**

Opt “pseudo” RR

- OPT resource record (RR type code 41)
- Pseudo RR (doesn't exist as data in a zone)
- Appears in the “Additional Section” of a DNS message
- Contains maximum UDP Payload Size, extended RCODEs and flags
- Only flag defined to date: DNSSEC OK (DO)

[DNSSEC Tutorial, USENIX LISA 13]

29

New Header Flags: CD, AD

- AD - “Authenticated Data”
- CD - “Checking Disabled”

[DNSSEC Tutorial, USENIX LISA 13]

30

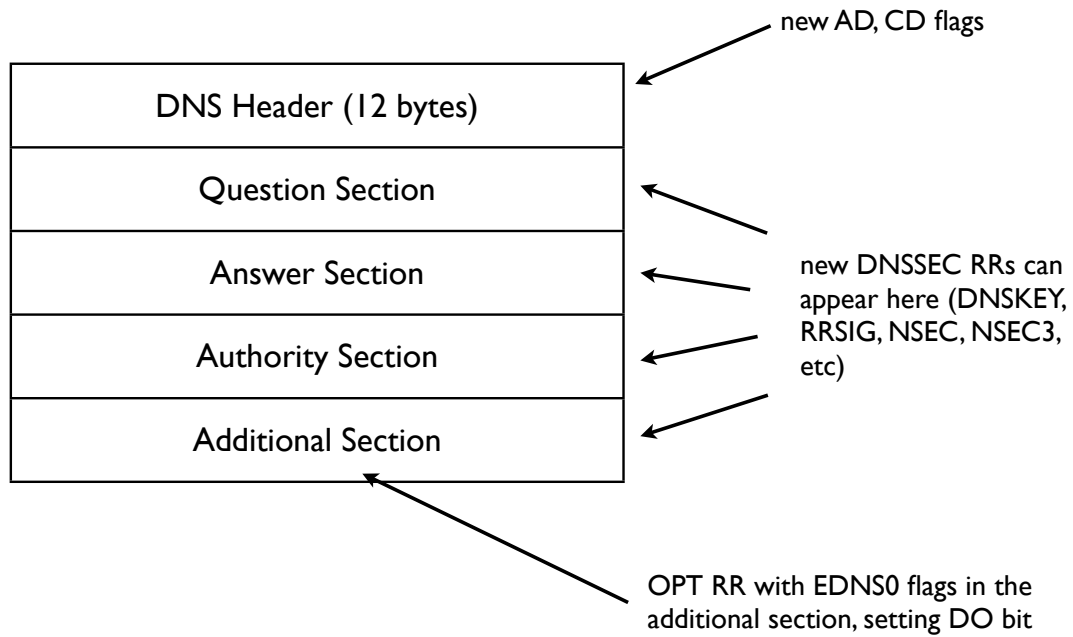
AD Flag

- AD - **“Authenticated Data”**
- Resolver sets this flag in responses when the queried record is signed with a valid, unexpired signature and an authenticated chain of trust all the way to a configured trust anchor (which could be the preconfigured/tracked root key)
- All data in the included *answer* and *authority* sections has been appropriately authenticated by the resolver
- Can also be set in a DNS query - to indicate querier understands responses with AD bit (eg. if it wants authenticated state but not necessarily DNSSEC RRs)

CD Flag

- CD - **“Checking Disabled”**
- Querier sets CD flag to indicate that “pending” (non-authenticated data) is acceptable to it, eg. because it is willing to do its own cryptographic validation of the signatures.
- DNSSEC enabled servers must not return “bad” data (eg. that have bad signatures) though (*)
- A conceivable use is that of a validating stub resolver.

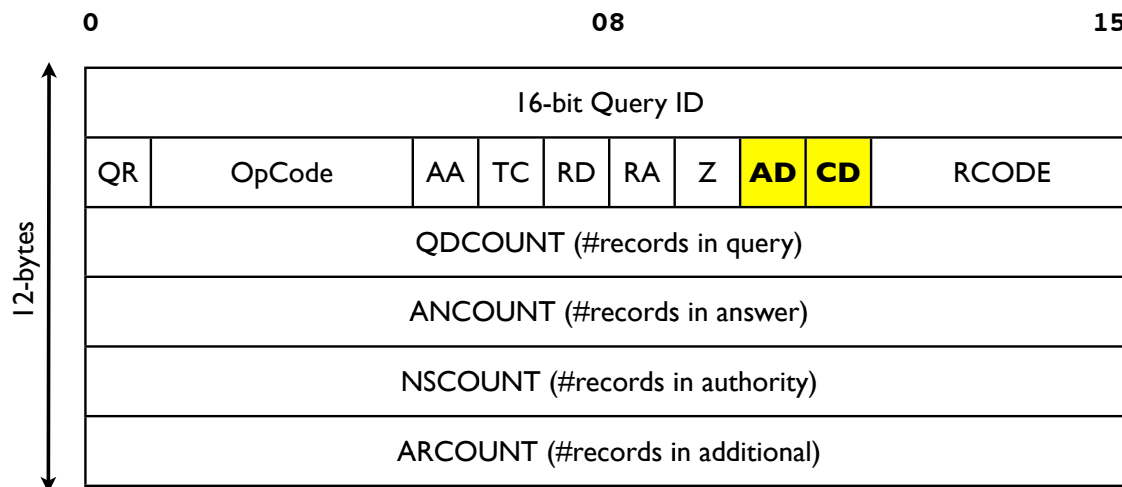
DNS Packet Format



[DNSSEC Tutorial, USENIX LISA 13]

33

DNS Header



[DNSSEC Tutorial, USENIX LISA 13]

34

DNS Response Codes

Common Response codes:

0	NOERROR	No Error
1	FORMERR	Format Error
2	SERVFAIL	Server Failure
3	NXDOMAIN	Not existent domain name
4	NOTIMPL	Function not implemented
5	REFUSED	Query Refused, usually by policy

Standard response code for DNSSEC responses that fail to authenticate (validate) properly, eg. bad signature, expired signature etc is SERVFAIL

[DNSSEC Tutorial, USENIX LISA 13]

35

Extended RCodes

Extended RCODES do not appear in the DNS header (since there isn't enough space there). They instead appear in the OPT pseudo RR, which has a special format designed to accommodate them.

Extended RCodes used by EDNS0, TSIG, TKEY, etc:

16	BADVERS	Bad OPT version
16	BADSIG	TSIG Signature Failure
17	BADKEY	Key not recognized
18	BADTIME	Signature out of time window
19	BADMODE	Bad TKEY Mode
20	BADNAME	Duplicate Key Name
21	BADALG	Algorithm not supported
22	BADTRUNK	Bad Truncation

[DNSSEC Tutorial, USENIX LISA 13]

36

Multiple DNSKEYs

- Typically, a 2-level hierarchy of DNSKEYs is employed
- KSK: Key Signing Key
 - Signs other keys (can be larger, ie. stronger, and kept offline; used as the trust anchor and certified by the parent zone in the DS)
- ZSK: Zone Signing Key
 - Signs all data in the zone (can be lower strength and impose less computational overhead; can be changed without co-ordination with parent zone)

Protection of signing keys

- Keep offline? Problems with dynamic signing
- Keep only KSK offline? But need to bring them online for key rollovers (even only ZSK rollovers)
- If keeping online, lock down housing server rigorously, as you might do a critical authentication server, like a KDC
- Physically secured machine room & racks
- Tamper resistant HSM (Hardware Security Module)

```
$ dig jabber.upenn.edu AAAA
```

```
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 337

;; QUESTION SECTION:
;jabber.upenn.edu.                IN      AAAA

;; ANSWER SECTION:
jabber.upenn.edu.                86400  IN      AAAA    2001:468:1802:101::805b:2ac

;; AUTHORITY SECTION:
upenn.edu.                       86400  IN      NS      dns2.udel.edu.
upenn.edu.                       86400  IN      NS      noc2.dccs.upenn.edu.
upenn.edu.                       86400  IN      NS      noc3.dccs.upenn.edu.
upenn.edu.                       86400  IN      NS      dns1.udel.edu.

;; ADDITIONAL SECTION:
noc2.dccs.upenn.edu.            86400  IN      A       128.91.254.1
noc2.dccs.upenn.edu.            86400  IN      AAAA    2001:468:1802:102::805b:fe01
noc3.dccs.upenn.edu.            86400  IN      A       128.91.251.158
dns1.udel.edu.                  86400  IN      A       128.175.13.16
dns2.udel.edu.                  86400  IN      A       128.175.13.17
```

[DNSSEC Tutorial, USENIX LISA 13]

39

```
$ dig jabber.upenn.edu AAAA +dnssec
```

```
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 690
;; flags: qr aa rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;jabber.upenn.edu.                IN      AAAA
```

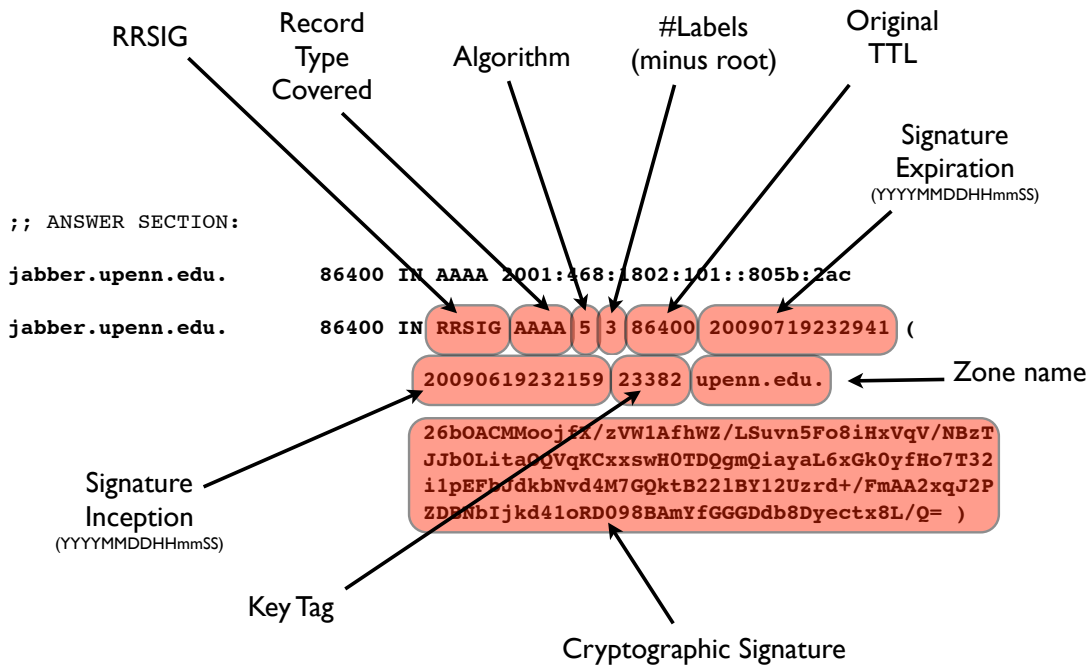
Authenticated Data

Answer &
Signature

DNSSEC Ok

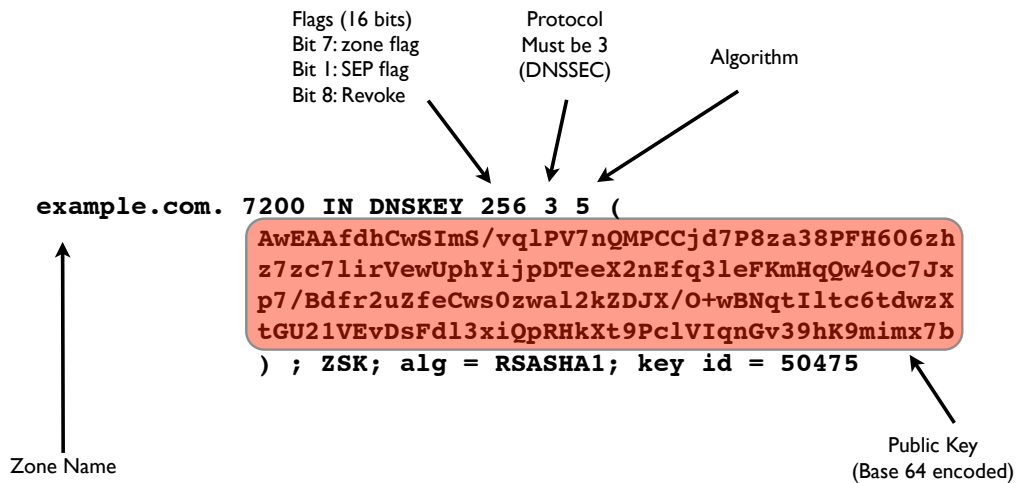
```
;; ANSWER SECTION:
jabber.upenn.edu.                86400  IN      AAAA    2001:468:1802:101::805b:2ac
jabber.upenn.edu.                86400  IN      RRSIG   AAAA 5 3 86400 20090719232941 (
20090619232159 23382 upenn.edu.
26b0ACMMoojfx/zVW1AfhWZ/LSuVn5Fo8iHxVqV/NBzT
JJb0LitaOQVqKCxxswH0TDQgmQiaYaL6xGk0yfHo7T32
i1pEFbJdkbNvd4M7GQktB221BY12Uzrd+/FmAA2xqJ2P
ZDBNbiJkd41oRD098BAmYfGGDdb8Dyectx8L/Q= )
```

```
;; AUTHORITY SECTION:
upenn.edu.                       86400  IN      NS      dns1.udel.edu.
upenn.edu.                       86400  IN      NS      noc3.dccs.upenn.edu.
upenn.edu.                       86400  IN      NS      dns2.udel.edu.
upenn.edu.                       86400  IN      NS      noc2.dccs.upenn.edu.
upenn.edu.                       86400  IN      RRSIG   NS 5 2 86400 20090719232217 (
20090619223616 23382 upenn.edu.
WwP4u9p5zORM+207pRZ46+Qo3chj9tnjxH62Xt9QBR
yu9V7+3ih1IM1HCd9kjsddskT8GJ+5hEzyk8fPIjSli
bqG6hCnCcGdTsGzmPoGdlz95H7Nf2yfrlGLAcSCix6I
EJb8Aj4+OW9Zq1dmeZrnJDXSzm8joQg5+IlkzR4= )
```



DNSKEY record

- Contains zone's DNSSEC public key and associated flags



Common DNSSEC algs

Algorithm#	Algorithm
5	RSA/SHA1 (default in BIND 9)
6	DSA-NSEC3-SHA1
7	RSA-NSEC3-SHA1
8	RSA/SHA256
10	RSA/SHA512
12	ECC-GOST
13	ECDSA Curve P-256 SHA256
14	ECDSA Curve P-384 SHA384

<http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml>

[DNSSEC Tutorial, USENIX LISA 13]

43

```

$ dig upenn.edu DNSKEY

;; ANSWER SECTION:
upenn.edu. 7200 IN DNSKEY 256 3 5 (
AwEAAcDt107stSjvoBA/YVPr+2gvB3v33tXr7ROZ/Jqm
WtNLraxQPzqXM1AhwjtdeQwCAnk01V7+Fw7K94sh6jpI
5bFofS7MGtd0VvNyq52bgRnusgbm1ME2Lx9+o3fy9ppv
7C6bahGrV3aiq9wNVPj/ccJn5AnZCOsi3grVsj6izCYH
) ; key id = 46752
upenn.edu. 7200 IN DNSKEY 256 3 5 (
AwEAAfAHsS33kJEImVk09yFJY5hXumAo+JVVJMjPjUaj
l/rh0fFkdiks2oatVvxHHHqKN9Kg3DoKQss/CzCza4zn
KlqYGvS17RefKR3QLyPBGQN2aOUWxshDgOWLmOtqNpmP
+6Drfn8LJVTOjuwmU801aQcdA/AoOGVPE3zP16G/F+qp
) ; key id = 43248
upenn.edu. 7200 IN DNSKEY 257 3 5 (
AwEAAek95gyBF2nurdIE2Q63VVcMlazOlQEnz0N4Ce89
SB4Juw2eEBerLmEanuGJbrs0oGx3SKCMyhOYL9q1ZrmC
NCf6PnAcwv88NtrYOjHAOmOllAvKAQv8MTBbEwTWBw5
K8jUwzcaGyDjo3U+Hai+ow8Tiev0By+hrcT4DegsbEB8
MEQIqEUO/Kw9wbJLEdpvVXtuV2178G75FUwmrA8jzEka
M7bKg/HSTIMupbwfs4IHYgbG/PkqOZYL3uxm9gncVjbh
4YYd4OG6koVoWteWTS8JdYq4gr9b9AEjhwAzbe7bd7pX
+qd70CCbh0jSOVhPvhrpCHIYZAJIwEAWs711HMM=
) ; key id = 29242
    
```

Diagram annotations:

- Arrows point from the words "flags", "proto", and "algorithm" to the numbers 256, 3, and 5 respectively in the DNSKEY records.
- An arrow points from the word "encoded public key" to the long alphanumeric string in the third DNSKEY record.

Negative answers

- “Authenticated Denial of Existence”
- NSEC or NSEC3 records (and their signatures)
- Chain together DNS records in a zone; can think of them and their signatures as spanning the gaps between names in the zone
- Canonical ordering of names in signed zones needed (RFC 4034, Section 6.1)
- Needed because of the pre-computed signature model of DNSSEC (computational concerns & signing key security)

[DNSSEC Tutorial, USENIX LISA 13]

45

Canonical Order

Sort DNS names in order of most significant (rightmost) labels first. Then within each label, sort them as octet strings, case-folding ASCII letters to lowercase.

```
example.com
a.example.com
blah.a.example.com
Z.a.example.com
zABC.a.EXAMPLE.com
z.example.com
\001.z.example.com
*.z.example.com
\200.z.example.com
```

(See RFC 4034, Section 6.1)

[DNSSEC Tutorial, USENIX LISA 13]

46

NSEC3 differences

- NSEC3 instead of NSEC records
- Owner name is a cryptographic hash of the name (flattened) rather than the actual name - provides zone enumeration defense
- Some names may not have an NSEC3 (the “opt-out” feature)
- Additional apex record: NSEC3PARAM
- Increased CPU usage implications
- See RFC 5155 (Hashed Authenticated Denial of Existence) for details

[DNSSEC Tutorial, USENIX LISA 13]

47

NSEC record

- “Next Secure” record
- Describes interval between consecutive names in a zone
- Type-bitmap defines RRtypes available at owner
- Side Effect: allows enumeration of zone contents

a.example.com. 300 IN NSEC d.example.com. **A MX RRSIG NSEC**

The diagram shows an NSEC record: `a.example.com. 300 IN NSEC d.example.com. A MX RRSIG NSEC`. Four arrows point from labels below to the corresponding fields in the record:

- Owner Name points to `a.example.com.`
- SOA min TTL points to `300`
- Next Owner Name points to `d.example.com.`
- Type Bitmap (List of Types defined at Owner Name) points to the red rounded rectangle containing `A MX RRSIG NSEC`.

[DNSSEC Tutorial, USENIX LISA 13]

48

An authenticated negative answer (nxdomain)

```
$ dig +dnssec +multi bozo.upenn.edu A

;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 7708

;; ;; AUTHORITY SECTION:

[SOA and RRSIG(SOA) records omitted for brevity]

upenn.edu.      3600 IN   NSEC _kerberos.upenn.edu. NS SOA MX RRSIG NSEC DNSKEY TYPE65534
upenn.edu.      3600 IN   RRSIG NSEC 5 2 3600 (
20120508051318 20120408042226 50475 upenn.edu.
ZzTYjeHECy5xLo+wrXq1VwmtNI3Wz7cpNLBdg+3xM9ph
H9jOndAViCKwsDa4uLBYBcKss9qbbYjVtMp5w0lmVpwm
cwxYheAyEN+w2VPBhLZ9qjfib8Q6Lfi3r3lC8EDJciL0
1LSQwP2gyFx7V6VG08z1lW6fuB6A/6/3/55xwW0= )

cagrid.bmif.upenn.edu. 3600 IN   NSEC BRYNMAWR-GW.upenn.edu. CNAME RRSIG NSEC
cagrid.bmif.upenn.edu. 3600 IN   RRSIG NSEC 5 4 3600 (
20120507190845 20120407181400 50475 upenn.edu.
yn4Au0Q4EViyu0LonWLWviTUn6kLYfyMMERajl2Jdaob
CYLfwNWMrXYPH6IZu03dYSkIRg7WEoyEGckk5J5Gudok
ikdFEEuuBjV4cdUCMp67lvUjCGVclFwnKhb5ni/FmieH
q7yFeztBt/IsKxtbcFSX0Isjt5mtNqt5is+UNpY= )
```

*.upenn.edu would have been between upenn.edu and _kerberos.upenn.edu

bozo.upenn.edu would have been between cagrid.bmif.upenn.edu & brynmawr-gw.upenn.edu

An authenticated negative answer (nodata)

NOERROR (nodata) responses can be authenticated with one signed NSEC record, which just reports all available RRTYPEs at that name

```
$ dig +dnssec upenn.edu A

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44529

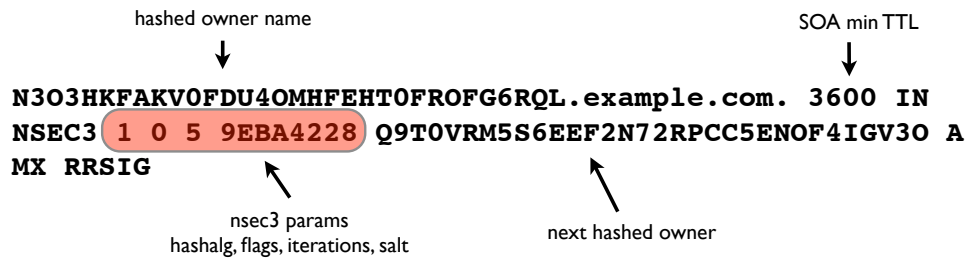
;; AUTHORITY SECTION:

[SOA and RRSIG(SOA) records omitted for brevity]

upenn.edu.      3600 IN   NSEC _kerberos.upenn.edu. NS SOA MX RRSIG NSEC DNSKEY TYPE65534
upenn.edu.      3600 IN   RRSIG NSEC 5 2 3600 (
20120508051318 20120408042226 50475 upenn.edu.
ZzTYjeHECy5xLo+wrXq1VwmtNI3Wz7cpNLBdg+3xM9ph
H9jOndAViCKwsDa4uLBYBcKss9qbbYjVtMp5w0lmVpwm
cwxYheAyEN+w2VPBhLZ9qjfib8Q6Lfi3r3lC8EDJciL0
1LSQwP2gyFx7V6VG08z1lW6fuB6A/6/3/55xwW0= )
```

NSEC3 record

- New version of NSEC3 that provides defense against zone enumeration (see RFC 5155 for details)
- Hashed owner names (base 32 with extended hex alphabet)
- Optional “opt-out” feature
- rdata: nsec3 parameters (hash alg, flags, iterations), hashed next owner name, type bitmap

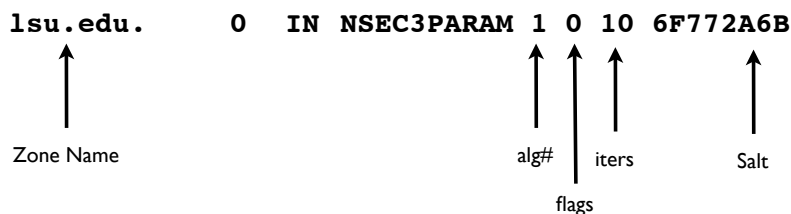


[DNSSEC Tutorial, USENIX LISA 13]

51

NSEC3PARAM record

- NSEC3PARAM record at zone apex also holds the parameters
- Hash algorithm, Flags, #Iterations, Salt
- This is used by secondary nameservers for the zone, to choose an appropriate set of NSEC3 RRs for responses



[DNSSEC Tutorial, USENIX LISA 13]

52

Authenticated negative answer (NSEC3 nxdomain)

(Example taken from RFC 5155 Appendix B. Consult for details) covers "next closer name"
Question: a.c.x.w.example. IN A

```
;; AUTHORITY SECTION:
0p9mhavEqvm6t7vb15lop2u3t2rp3tom.example. NSEC3 1 1 12 aabbccdd (
    2t7b4g4vsa5smi47k61mv5bv1a22bojr MX DNSKEY NS
    SOA NSEC3PARAM RRSIG )
0p9mhavEqvm6t7vb15lop2u3t2rp3tom.example. RRSIG NSEC3 7 2 3600 (
    20150420235959 20051021000000 40430 example.
    OSgWsm26B+cS+dDL8b5QrWr/dEWhtCsKlwKL
    IBHYH6blRxK9rC0bMJPwQ4mLIuw85H2EY762
    BOCXJZMnpuwHPA== )
b4um86eghhd6nea196smvml04ors995.example. NSEC3 1 1 12 aabbccdd (
    gjeqe526plbflg8mklp59enfd789njgi MX RRSIG )
b4um86eghhd6nea196smvml04ors995.example. RRSIG NSEC3 7 2 3600 (
    20150420235959 20051021000000 40430 example.
    ZkPG3M32lmoHM6pa3D6gZFGb/rhL//Bs3Omh
    5u4m/CUiwTbLEVOaAKKZd7S9590eiX43aLX3
    pOv0TSTYiTxiZg== )
35mthgpgculqg68fab165klnsnk3dpv1.example. NSEC3 1 1 12 aabbccdd (
    b4um86eghhd6nea196smvml04ors995 NS DS RRSIG )
35mthgpgculqg68fab165klnsnk3dpv1.example. RRSIG NSEC3 7 2 3600 (
    20150420235959 20051021000000 40430 example.
    g6jPUUpduAJKR1jUsN8gB4UagAX0NxY9shwQ
    Aynzo8EUWH+z6hEIBLUTPGj15eZl16VhQqgZ
    XtAIR3chwGw+SA== )
```

← matches closest encloser

← covers wildcard at closest encloser

Authenticated negative answer (NSEC3 nodata)

NOERROR (nodata) responses can be authenticated with one signed NSEC record, which just reports all available RRTYPEs at that name (for qtype != DS)

In the example below blah.huque.com exists (TXT) but not for the MX record type.

```
$ dig +dnssec blah.huque.com. MX
```

```
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 65366
```

```
;; AUTHORITY SECTION:
```

```
[SOA and RRSIG(SOA) omitted for brevity]
```

```
Q9T0VRM5S6EEF2N72RPCC5ENOF4IGV30.huque.com. 3284 IN RRSIG NSEC3 8 3 3600 (
    20121114122449 20121015121429 14703 huque.com.
    lSu/WBjB3rBsU8ObV4bChPIMwCk93ac1B4b0Pq14m+Zo
    XOkgu+PAqWLBm8FFeWwnT74XOWMXe+jvNMLSQ/nWfEjE
    s+l5lWsm4XJma0Pl+SoSHdIq1vJ9KfeEiWD8xLbpKH/N
    3qwjnf4p4Fcma8LB6va4niZiJulMGNFzgRmtFDE= )
```

```
Q9T0VRM5S6EEF2N72RPCC5ENOF4IGV30.huque.com. 3284 IN NSEC3 1 0 5 9EBA4228 (
    1M2GGNB8TPSI4SPF73V8RJS95FLHBNC0 TXT RRSIG )
```

↑
Hash of blah.huque.com.

↑
Next hashed name

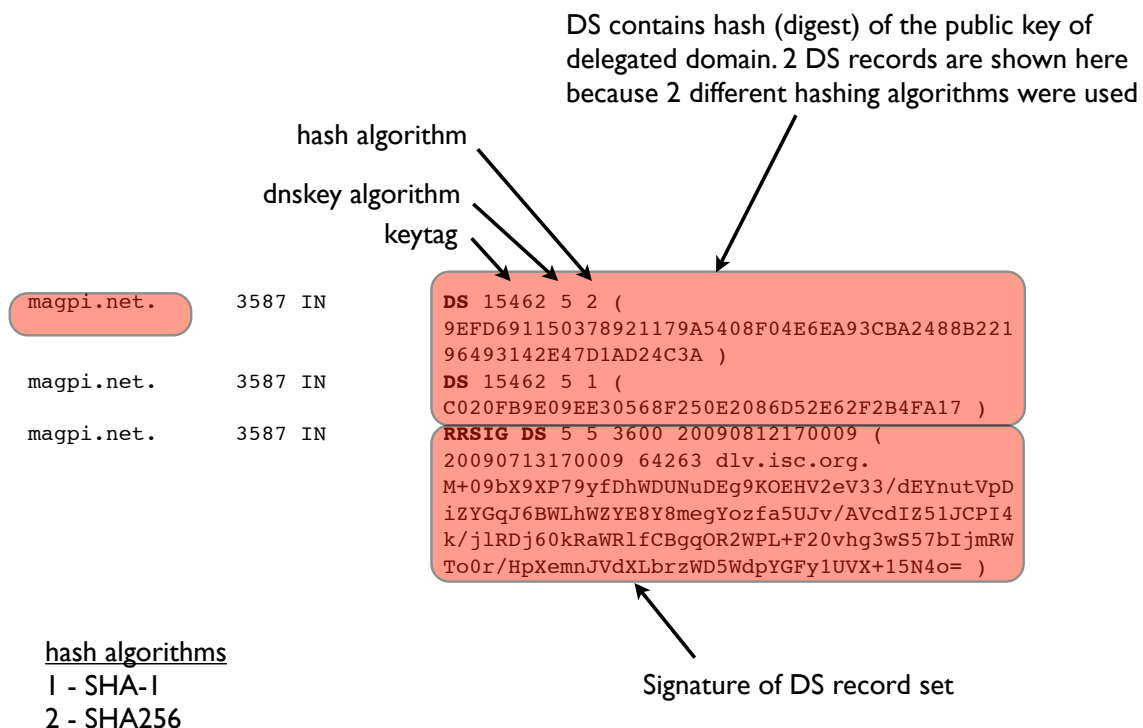
↑
Type bitmap

Secure Delegations

- Indicated by DS (Delegation Signer) record
- Appears in the delegating (ie. parent) zone
- Contains a hash of the public key of the child zone's
- Validating resolvers use the presence of the DS record *and* its corresponding signature (RRSIG) to securely authenticate the delegation

[DNSSEC Tutorial, USENIX LISA 13]

55



A DS record in a parent zone refers to a specific DNSKEY record in a child zone

```
upenn.edu.      86247 IN DS 18463 5 2 (
6003992326DA06785C9E30B259750FAB0960BF57054B
DDFFDEEE1188977DABB8 )
86247 IN DS 18463 5 1 (
0C45B3D090B221E0E33BBEB5A619D89416BAF197 )
86247 IN RRSIG DS 8 2 86400 (
20131004044110 20130927033110 21638 edu.
piBkiV626itFwzUdoKk81171jbN2+EUCNYESuKShN2kw
7GapeS0gTW5kafTtFlB7AjNN8AR85YrfH56XKwEvRoo1
yN2Tz1EE0eLtiQnMEwAi0FZ7NKz7eb2IMezGtFvUUqHm
x0qf2B5r4JHPhyB74M10+BY16LWyeoujUsezD0w= )
```

edu zone

```
upenn.edu.      7200 IN DNSKEY 257 3 5 (
AwEAAf02wZZMtX2ofTKfJ/xoQffn17NFJV0Y5s4F3tMd
kktC/abDax+SB0MJWRaczniGKoirdVL7ZbVS2DYRoUvJ
v44VLRtLlbggxDnhpP4fCJ88Yu33/5GFZwmgxco40A6y
xhwEniIveQ5B7LJ0Vh8KyfqU6obu7wFR7pSV1UVybLZf
F3n1Kb+6KRWtave5JLbhfYfXYxhUpVWlbeKYmorO9SH
sQaoR3vr7L168MEe4VRE+SKcuNRkzLbg5XQDnImYanv6
Pf4tYaNTGYPgkXSVJKeGUdncJOxZ8NrAqOncGdgQML3x
ALGWHWlRpmEN6EQdf7+qv1vm7uHxN1L0MhS7B4U=
) ; KSK; alg = RSASHA1; key id = 18463

[ ... other DNSKEYs and RRSIGs omitted ...]
```

upenn.edu zone

Parent unsigned?

- What if you want to deploy DNSSEC, but ...
- Your parent zone isn't signed
- Or it is signed, but you are unable to get a secure delegation installed in it because your domain registrar can't do it

DNSSEC Lookaside Validation (DLV)

- A mechanism to securely locate DNSSEC trust anchors “off path”
- Intended as an early deployment aid until top-down deployment of DNSSEC is completed
- DLV Registry operated by Internet Systems Consortium (<https://www.isc.org/solutions/dlv>)
- If you can't find a DS record for example.com, look for a DLV record for example.com.<dlv-domain>

```
magpi.net.dlv.isc.org.  IN  DLV 15463 5 2  
      (9EFD691150378921179A5408F04E6EA93CBA  
      2488B22196493142E47D1AD24C3A )
```

[DNSSEC Tutorial, USENIX LISA 13]

59

Live DNS queries with dig

[DNSSEC Tutorial, USENIX LISA 13]

60

In this section, we'll look at some live DNS queries with the “**dig**” tool, widely available on most UNIX/Linux platforms.

Common invocations:

```
dig <qname>
dig <qname> <qtype>
dig @server <qname> <qtype>
dig -x <ipaddress>
dig +trace <qname> <qtype>
```

Useful additional flags for DNSSEC related operations ...

```
+dnssec    request DNSSEC RRs via DO=1
+multi     pretty print output across
           multiple lines with annotation
+adflag    set AD flag
+cdflag    set CD flag

+sigchase  obsolete. Use “drill” instead
```

Configuring DNSSEC in BIND

[DNSSEC Tutorial, USENIX LISA 13]

63

General advice

- Use the latest possible version of BIND (current is v9.9)
- Many more features that make DNSSEC configuration much much easier, and almost automated ...

[DNSSEC Tutorial, USENIX LISA 13]

64

Additional details

- The BIND ARM (Administrator's Reference Manual)
- <http://www.isc.org/software/bind/documentation>
- For latest BIND version (9.9):
 - <http://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/Bv9ARM.html>
- Essential reading for the BIND DNS operator

Summary of steps

- DNS Resolver Operator
 - Configure resolver to perform DNSSEC validation
- DNS Zone operator
 - Sign zone(s) with DNSSEC
 - Secure zone transfers (typically with TSIG)
 - Obtain secure delegation (DS record) at parent zone

Validating Resolver

In named.conf:

```
options {
    [...]
    dnssec-enable yes;
    dnssec-validation auto;
    dnssec-lookaside auto;
    [...]
};
```

This will use BIND's built-in keys for the root and the ISC DLV registry, and will automatically rollover keys as they are detected.

[DNSSEC Tutorial, USENIX LISA 13]

67

Validating Resolver

```
# named.conf Validating recursive resolver example
```

```
acl trusted {
    192.0.2.0/24;           # my clients IPv4 address block
    2001:db8:f470::/48;   # my clients IPv6 address block
}
```

```
options {
    max-cache-size 1024M;
    listen-on-v6 { any; };
    dnssec-enable yes;
    dnssec-validation auto;
    dnssec-lookaside auto;
    allow-query-cache { trusted; };
    allow-recursion { trusted; };
};
```

```
zone "." {
    type hint;
    file "named.root";
};
```

who's allowed to use the recursive resolver.
(note: some people run open servers)

root nameserver addresses. latest version at
www.internic.net/domain/named.root

[DNSSEC Tutorial, USENIX LISA 13]

68

Validating Resolver

Manually configured keys (if needed):

```
# manually configured static key
trusted-keys {
    . 257 3 8 "AwE...jlsdjfld=";
};

# managed keys (with automated rollover)
managed-keys {
    "." initial-key 257 3 8 "Awlsdjflkdjfl";
};
```

[DNSSEC Tutorial, USENIX LISA 13]

69

Generating keys

Generating Keys:

```
dnssec-keygen <zone>
dnssec-keygen -f KSK <zone>    # KSK generation
dnssec-keygen -3 <zone>        # NSEC3 compat alg
```

Creates files:

```
K<zone>+mmm+nnnn.key and
K<zone>+mmm +nnnn.private
```

[DNSSEC Tutorial, USENIX LISA 13]

70

Signing zones

Signing Zone:

```
dnssec-signzone -o <origin> -S <zonefile>
```

```
-o origin: zone origin  
-S: smart signing  
-N [keep|increment|unixtime] # serial number  
-3: NSEC3 signing  
-g: generate DS records for children from dsset-  
    or keyset- files  
-l domain: generate DLV records at domain  
-s YYYYMMDDHHMMSS # sig start time  
-e YYYYMMDDHHMMSS # sig end time  
-T ttl: ttl for DNSKEY, default from SOA
```

[DNSSEC Tutorial, USENIX LISA 13]

71

Signing zones

Signing Zone with NSEC3:

```
dnssec-signzone -o <origin> -3 <salt>  
                -H <iterations> -S <zonefile>
```

```
-3 <salt>: hex-encoded salt  
-H <iterations>: num of hash iterations (def 10)  
-A: set opt-out flag
```

[DNSSEC Tutorial, USENIX LISA 13]

72

Authoritative Server

```
options {
    [...]
    dnssec-enable yes;
    [...]
};
```

[DNSSEC Tutorial, USENIX LISA 13]

73

Authoritative Server

The master (primary master) authoritative server should define an access control list to limit the servers (usually only its slave servers) which can perform zone transfers of the DNS database. Note however, that this is a policy decision. Some folks allow anyone to transfer the contents of their zone.

```
# List of authorized secondary/slave servers
acl transferlist {
    192.0.2.2/32;
    192.0.2.3/32;
    2001:db8:f470:1234:2/128;
    2001:db8:f470:1234:3/128;
}

options {
    [...]
    allow-transfer {
        transferlist;
    };
    [...]
};
```

[DNSSEC Tutorial, USENIX LISA 13]

74

Authoritative Server

Authoritative Servers, need **zone definitions** for the zones they are serving. They should also disable recursion if not also providing recursive resolver service to endusers.

```
options {
  [ ... various options ...];
  recursion no;
};

zone "example.com" {
  type master;
  file "zone.example.com";
};

zone "example.com" {
  type slave;
  file "zone.example.com";
  masters { 10.2.2.2; };
};
```

← if authoritative only

← on master server

← on slave server

[DNSSEC Tutorial, USENIX LISA 13]

75

zone xfr with TSIG

Authenticating Zone Transfers with TSIG:

On primary master server:

Generate TSIG key with (example):

```
$ dnssec-keygen -a HMAC-MD5 -b 128 -n HOST slavel.example.com.
```

File: zonetransfer.key:

```
key "slavel.example.com." {
  algorithm "hmac-md5";
  secret "xjlsjdlfdhfhdfldfljdfsljdljsdlfjdlkf=";
};
```

File: named.conf:

```
include "/usr/local/bind/zonetransfer.key"
```

```
options {
  [...]
  allow-transfer { key slavel.example.com.; };
  [...]
};
```

secret key taken from K* files produced by dnssec-keygen

can also be used within individual zone stanzas

[DNSSEC Tutorial, USENIX LISA 13]

76

zone xfr with TSIG

Authenticating Zone Transfers with TSIG (continued):

On secondary (slave) server (use same key as configured on master):

```
File: named.conf:
include "/usr/local/bind/zonetransfer.key"

zone "example.com" {
    type slave;
    masters { 10.12.7.26 key slave1.example.com.; };
    [...]
};
```

It is also possible to sign and authenticate all transactions with a master server (not just AXFR/IXFR) with a "server" statement:

```
server 10.12.7.26 {
    keys { slave1.example.com.; };
};
```

[DNSSEC Tutorial, USENIX LISA 13]

77

Dynamic Update + DNSSEC

The easiest way, in my opinion.

- * Configure dynamic zones (ie. zones updated only with the Dynamic Update protocol, eg. with the nsupdate program)
- * Make DNSSEC keys available to named
- * When dynamic updates are made, named will automatically sign the records and generate or re-generate related DNSSEC metadata

- * Latest BIND versions include special options to make this really easy.

[DNSSEC Tutorial, USENIX LISA 13]

78

Other measures?

- Ideally, these shouldn't be necessary but ...
- If needed, to workaround some types of firewalls and middleboxes (on at least one server)
- Constrain EDNS0 payload size (< PMTU)
 - eg. "edns-udp-size 1472"
- Configure minimal-responses ("minimal-responses yes")
- Make sure DNS over TCP is allowed (see RFC 5966) - you should always do this!

[DNSSEC Tutorial, USENIX LISA 13]

79

Accurate time

- DNSSEC has an important dependency on accurate time
 - Validating resolvers need to check signature validity time
 - Signing servers need to produce correct signature validity intervals
- Make sure your servers have accurate time
- I'd recommend configuring them to get authenticated time from an NTP server

[DNSSEC Tutorial, USENIX LISA 13]

80

Demo: signing a zone

[DNSSEC Tutorial, USENIX LISA 13]

81

Live example of signing a zone with DNSSEC
(Time permitting!)

[DNSSEC Tutorial, USENIX LISA 13]

82

Signing a zone

Steps for reference.

```
# Create zone for "example.com" and configure named
[...]

# Generate KSK and ZSK (in this example RSASHA256 2048/1024bit)
dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK example.com
dnssec-keygen -a RSASHA256 -b 1024 -n ZONE example.com

# Sign zone (will generate "zonefile.signed")
dnssec-signzone -o example.com -N increment -S zonefile

# Reconfigure named.conf to serve "zonefile.signed"
[...]
```

[DNSSEC Tutorial, USENIX LISA 13]

83

Signing a zone (dynamic)

```
# Generate KSK and ZSK as before, but don't use dnssec-signzone
[...]

# Setup named.conf with the "auto-dnssec" option for the zone
zone "example.com" {
    type master;
    update-policy local;                # allow-update for expl key
    auto-dnssec allow;                # also see "maintain"
    file "zones/example.com/zonefile";
    key-directory "zones/example.com";
};

# Instruct nameserver to sign the zone.
rndc sign example.com

# From now, use dynamic update (eg. via nsupdate) to update
# zone contents.
```

[DNSSEC Tutorial, USENIX LISA 13]

84

“auto-dnssec maintain”

Will automatically do initial signing (no need to issue “rndc sign <zone>”), re-sign records periodically, and handle key rollovers by examining timing metadata in key files set with “dnssec-settime”

```
zone "example.com" {  
    type master;  
    update-policy local;  
    auto-dnssec maintain;  
    file "zones/example.com/zonefile";  
    key-directory "zones/example.com";  
};
```

[DNSSEC Tutorial, USENIX LISA 13]

85

NSEC3 dynamic zone

Use “**rndc signing**” command

```
signing -nsec3param hash flags iterations salt zone  
        [class [view]]
```

Add NSEC3 chain to zone if already signed.

Prime zone with NSEC3 chain if not yet signed.

eg. rndc signing -nsec3param 1 0 5 9EBA4228 example.com

Alternatively, use dynamic update (eg. via nsupdate) to add an NSEC3PARAM record to the zone apex.

[DNSSEC Tutorial, USENIX LISA 13]

86

Updating a zone (dynamic)

```
# Example of using dynamic update to add an ldap.example.com
# A RR to the zone .. This will cause named to automatically
# compute and add RRSIGs and NSEC/NSEC3s as needed, and install
# them in the zone.

$ nsupdate -l
ttl 86400
zone example.com.
update add ldap.example.com. A 10.4.4.4
send
^D
$
```

[DNSSEC Tutorial, USENIX LISA 13]

87

Other methods

Newest versions of BIND have some other ways that might make it easier to deploy DNSSEC in some environments where it's not easy to modify the master server ...

* **Inline Signing** (BIND 9.9)

This feature greatly simplifies the deployment of DNSSEC by allowing completely automatic, fully transparent signing of zones. Using the new 'inline-signing' option in a master server allows named to switch on DNSSEC in a zone without modifying the original zone file in any way. Using it in a slave server allows a zone to be signed even if it's served from a master database that doesn't support DNSSEC.

Some example configurations may be found at

<https://kb.isc.org/article/AA-00626/0/Inline-Signing-in-ISC-BIND-9.9.0-Examples.html>

[DNSSEC Tutorial, USENIX LISA 13]

88

Key Rollover

[DNSSEC Tutorial, USENIX LISA 13]

89

Key Rollover

- Conventional wisdom is that DNSSEC keys should be changed (“rolled over”) at regular intervals. However, not everyone agrees, including some noted security experts
- If you choose strong enough keys, there is no cryptographic reason to routinely roll them
- There are good operational reasons to change keys *after specific events*, eg. turnover of a staff member who had access to the private keys, or a system compromise of the server
- Some argue routine key rollover instills practice & confidence that you’ll be able to do it properly when you really need to. However, do we do this for other applications (Kerberos, PKI/CAs, SSL)?

[DNSSEC Tutorial, USENIX LISA 13]

90

Key Rollover

- However, most sites do routinely change DNSSEC keys
- Typically, ZSKs are rolled over more frequently (eg. a few times per year, this can be done transparently, and with no co-ordination with the parent zone)
- KSKs are rolled less frequently (typically once per year or less). This does require co-ordinating with the parent zone to sign and install new DS records for the KSKs.
- Note: ICANN is planning a rollover of the root KSK
 - <http://www.icann.org/en/news/public-comment/root-zone-consultation-08mar13-en.htm>

Key Rollover

- RFC 6781: DNSSEC Operational Practices (v2)
 - Covers general practices, procedures, recommendations
- Most commonly used:
 - KSK rollover: double signature policy
 - ZSK rollover: pre-publish policy

KSK: Double signature

- Generate new KSK; publish (public part) in zone
- Sign DNSKEY RRset with both keys
- Publish additional DS record in parent for new key
- Wait until DS is propagated and TTL of the old DS record
- Remove the old KSK and re-sign DNSKEY RRset with only new key, and remove old DS record from parent

[DNSSEC Tutorial, USENIX LISA 13]

93

ZSK: Pre-publish

- Generate new ZSK, and publish the DNSKEY in the zone, but do not yet sign zone data with it
- Wait zone propagation time + TTL of the DNSKEY RRset
- Use new ZSK for signing zone records instead of old ZSK, but leave the old ZSK published in the zone
- Wait zone propagation time + largest TTL of all records in the zone
- Remove old key & re-sign DNSKEY RRset

[DNSSEC Tutorial, USENIX LISA 13]

94

dnssec-settime

```
(with "auto-dnssec maintain")
$ dnssec-settime -p all Kexample.com.+008+04065
Created: Fri Apr 19 21:16:43 2013
Publish: Fri Apr 19 21:16:43 2013
Activate: Fri Apr 19 21:16:43 2013
Revoke: UNSET
Inactive: UNSET
Delete: UNSET

-L ttl          default TTL for this key
# timing params: args are of form:
#             YYYYMMDDHHMMSS
#             YYYYMMDDHH
#             +/- <seconds>   (or w/ suffix y/mo/w/d/h/mi)
#             none             -> to unset
-P date/offset  publication time
-A date/offset  active time
-R date/offset  revoke time
-I date/offset  inactive time
-D date/offset  delete time
```

[DNSSEC Tutorial, USENIX LISA 13]

95

Re-signing Records

- Regardless of key rollover, DNS records in a zone need to be re-signed periodically
- Limiting signature validity period reduces susceptibility to replay attacks in the event the data changes (ie. ability for an attacker to replay a previously valid response)

[DNSSEC Tutorial, USENIX LISA 13]

96

Trust Anchor Updates

- RFC 5011: Automated Trust Anchor updates by resolvers
- A method to keep track of trust anchors (eg. the root key) and automatically reconfigure resolvers as those trust anchors are updated (eg. as a result of a scheduled key rollover)

Other DNSSEC caveats

General DNSSEC Caveats

- Zone size increases significantly when signed
- Memory and CPU usage increase
- DNSSEC answers are larger
- Server side & query side impacts
- Interference by firewalls, proxies, and other middlebox, eg. botching EDNS0, large packets, DNSSEC meta data , not passing all UDP fragments, etc
- Fallback to TCP increases
- Many modern resolvers already ask for DNSSEC by default (ie. set the DNSSEC-OK bit in their queries)

[DNSSEC Tutorial, USENIX LISA 13]

99

Amplification Attacks

- Increased susceptibility to Distributed Denial of Service (DDoS) attacks, using DNS response amplification
- <http://blog.huque.com/2013/04/dns-amplification-attacks.html>
- Look at Response Rate Limiting and other countermeasures
 - <http://www.redbarn.org/dns/ratelimits>

[DNSSEC Tutorial, USENIX LISA 13]

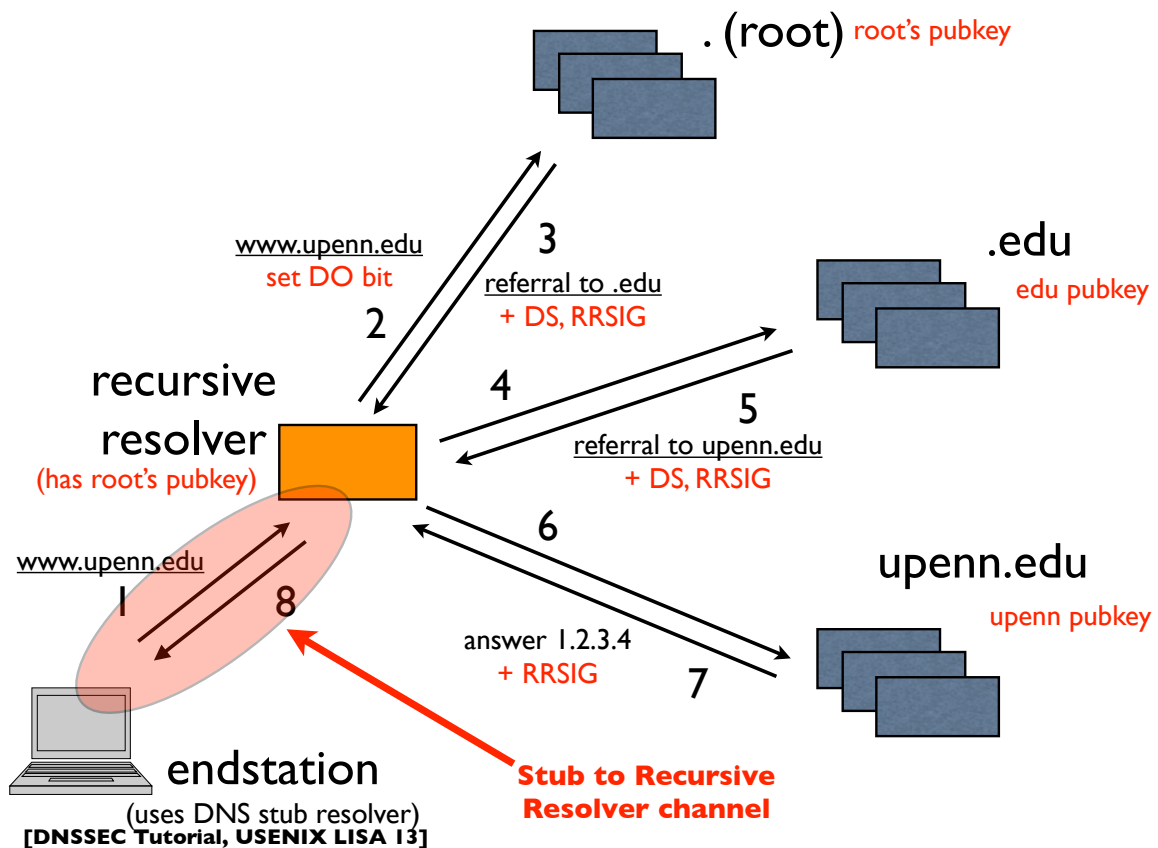
100

Securing the last hop

- How do we protect the stub resolver?
- Employ a channel security mechanism between stub and the upstream recursive resolver:
 - TSIG, SIG(0), IPSEC, etc
- Have the stub validate DNSSEC responses? Set CD bit and authenticate signatures directly?
- Run a full service validating DNS Resolver on clients?

[DNSSEC Tutorial, USENIX LISA 13]

101



102

Channel Security

- For stub channel security, simple symmetric key TSIG won't work
- Can't distribute same TSIG key to many clients, because that allows any of them to forge answers to all others
- Need per client keys and thus a key management infrastructure
- GSS-TSIG has a chicken-egg problem, because DNS is often used to locate Kerberos servers
- SIG(0) may be better - distribute single public key to clients
- Microsoft has an implementation of IPsec (GSS authenticated)
 - <http://technet.microsoft.com/en-us/library/ee649124%28v=ws.10%29.aspx>

[DNSSEC Tutorial, USENIX LISA 13]

103

DNSSCurve

- <http://dnscurve.org/>
- Some people think this is a competitor to DNSSEC, but it really isn't
- Encrypts/authenticates packets between resolvers and authoritative servers
- Uses very fast elliptic curve crypto
- DNS caching model better suited to object security, where response can come from any entity (authority, forwarder, intermediate cache, etc), but we can still authenticate the "data" inside the response
- But, we may need transport security as well (we live in the PRISM world of mass surveillance now!)

[DNSSEC Tutorial, USENIX LISA 13]

104

Other tools

Zone/validation testers

- Checking correct operation/deployment:
 - DNSviz: <http://dnsviz.net/>
 - <http://dnssec-debugger.verisignlabs.com/>
 - DNSCheck: <http://dnscheck.iis.se/>
- DNSSEC Validation testing
 - <http://dnssectest.sidn.nl/>
 - <http://test.dnssec-or-not.com/>

DNSSEC Trigger

- DNSSEC Trigger
 - <http://nlnetlabs.nl/projects/dnssec-trigger/>
 - Local resolver hack; probe for DNSSEC capable servers and instruct local resolver to use/validate
 - Last resort: tunnel over SSL to open DNSSEC validator elsewhere

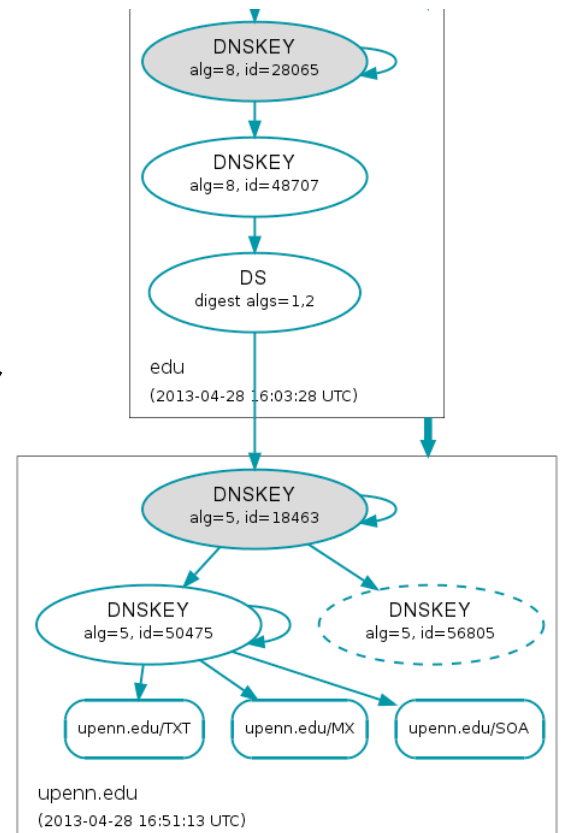
Zone maintenance

- 3rd party tools that some folks use to deploy/manage DNSSEC with BIND (mostly everything can be done in BIND itself these days):
 - OpenDNSSEC
 - zkt
 - <http://www.dnssec-tools.org/>
- Microsoft DNSSEC deployment guide
 - <http://www.microsoft.com/en-us/download/details.aspx?id=15204>

dnsviz

<http://dnsviz.net/>

DNSSEC zone and trust chain visualizer/debugger



[DNSSEC Tutorial, USENIX LISA 13]

109

Application use of DNSSEC

[DNSSEC Tutorial, USENIX LISA 13]

110

Application use of DNSSEC

- One of the more exciting prospects for DNSSEC
- DNSSEC allows applications to securely obtain (authenticate) cryptographic keying material stored in the DNS
- A variety of existing and proposed record types have been designed to store crypto material:
 - SSHFP, IPSECKEY, CERT
 - DKIM _domainkey TXT record (p=... public key data)
 - TLSA (upcoming, see IETF DANE working group)

[DNSSEC Tutorial, USENIX LISA 13]

111

Application use of DNSSEC

- Securely obtaining other assertions from the DNS
 - DKIM/ADSP
 - Route Origination Authorizations (controversial - see RPKI, the standardized mechanism to do this, which will allow BGP path validation also)

[DNSSEC Tutorial, USENIX LISA 13]

112

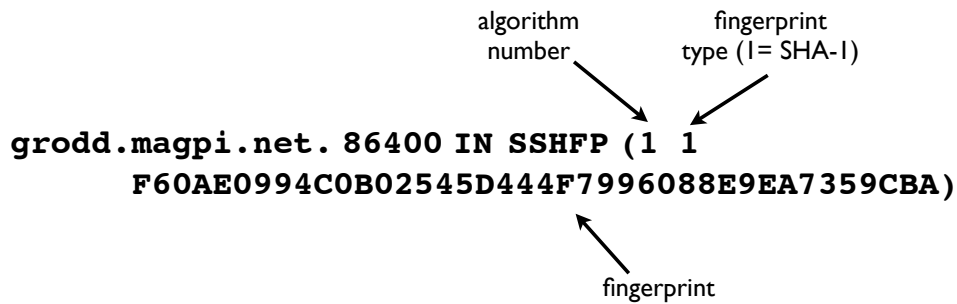
SSHFP record

- SSH Host Key Fingerprint (RFC 4255)
- Allows you to validate SSH host keys using DNS (securely using DNSSEC)

algorithm number fingerprint type (1= SHA-1)

```
grodd.magpi.net. 86400 IN SSHFP (1 1  
F60AE0994C0B02545D444F7996088E9EA7359CBA)
```

fingerprint



In **OpenSSH**, you can use the client configuration directive **"VerifyHostKeyDNS"** to use this.

[DNSSEC Tutorial, USENIX LISA 13]

113

IPSECKEY record

- RFC 4025: method for storing IPSEC keying material in DNS
- rdata format: precedence, gateway-type, algorithm, gateway address, public key (base64 encoded)

```
38.2.0.192.in-addr.arpa. 7200 IN IPSECKEY ( 10 1 2  
192.0.2.38  
AQNRU3mG7TVT02BkR47usntb102uFJtugbo6BSGvgqt4AQ== )
```

[DNSSEC Tutorial, USENIX LISA 13]

114

Public CA model problems

- Applications need to trust a large number of global certificate authorities, and this trust appears to be unfounded
- No namespace constraints! **Any** of them can issue certificates for **any** entity on the Internet, whether you have a business relationship with them or not
- Least common denominator security: our collective security is equivalent to weakest one
- Furthermore, many of them issue subordinate CA certificates to their customers, again with no naming constraints
- Most are incapable of issuing certs with any but the most basic capabilities (eg. alternate name forms or other extensions)

[DNSSEC Tutorial, USENIX LISA 13]

115

Public CA model problems

- Analysis of the HTTPS Certificate Ecosystem:
 - <http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf>
- Approximately 1,800 separate entities are capable of issuing certificates for anyone!

[DNSSEC Tutorial, USENIX LISA 13]

116

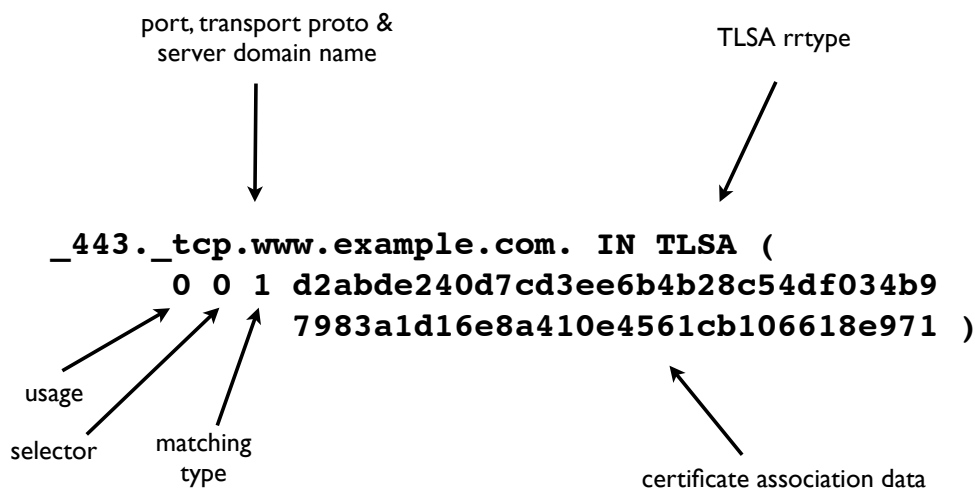
DANE/TLSA record

- RFC 6698: The DNS-Based Authentication of Named Entities (DANE) Protocol for Transport Layer Security (TLS)
 - <http://tools.ietf.org/html/rfc6698>
- Use DNSSEC for better & more secure ways to authenticate SSL/TLS certificates:
 - by specifying authorized public CAs, allowable end entity certs, authorizing new non-public CAs, or even directly authenticating certs without involving CAs!
- New record type: **TLSA**

[DNSSEC Tutorial, USENIX LISA 13]

117

TLSA record example



[DNSSEC Tutorial, USENIX LISA 13]

118

TLSA rdata parameters

Usage field:

- 0 CA Constraint
- 1 Service Certificate Constraint
- 2 Trust Anchor Assertion
- 3 Domain Issued Certificate

Selector field:

- 0 Match full certificate
- 1 Match only SubjectPublicKeyInfo

Matching type field:

- 0 Exact match on selected content
- 1 SHA-256 hash of selected content
- 2 SHA-512 hash of selected content

Certificate Association Data: raw cert data in hex

[DNSSEC Tutorial, USENIX LISA 13]

119

TLSA record example

Usage type 1: Service certificate constraint; match an end-entity certificate

```
_443._tcp.www.example.com. IN TLSA (  
  1 1 2 92003ba34942dc74152e2f2c408d29ec  
    a5a520e7f2e06bb944f4dca346baf63c  
    1b177615d466f6c4b71c216a50292bd5  
    8c9ebdd2f74e38fe51ffd48c43326cbc )
```

[DNSSEC Tutorial, USENIX LISA 13]

120

TLSA record example

(my own website; full cert assoc, no CA required)

```
$ dig +dnssec +multi _443._tcp.www.huque.com. TLSA

;; ANSWER SECTION:
_443._tcp.www.huque.com. 7200 IN TLSA 3 0 1 (
    7EF4BD014E9A4F302FC1EE74FB2D29718C5B0F4CB23B
    25B267A1D92F0410890B )

_443._tcp.www.huque.com. 7200 IN RRSIG TLSA 8 5 7200 (
    20131028121743 20130928111915 14703 huque.com.
    rjF6V1stQO50zG08s8m8DfBfqDvjqqzW3Im0Jc04HEDG
    fyvzQlCDX7Dxnbk7ZBoFGtNsVlx5XGS57k0ZLURsRWt
    wY+pqzcJ1ELVol6iOwNsOv+h9ZDyCa1GF7gL4k3DyKVe
    6cLquFa7RlywORqLYF32+adUP88/j63MmehR2VA= )
```

[DNSSEC Tutorial, USENIX LISA 13]

121

TLSA record (SMTP e.g.)

```
$ dig +dnssec +multi _25._tcp.nl.netlabs.nl. TLSA

;; ANSWER SECTION:
_25._tcp.nl.netlabs.nl. 10200 IN CNAME
3.1.1._dane.nl.netlabs.nl.
_25._tcp.nl.netlabs.nl. 10200 IN RRSIG CNAME 8 4 10200 (
    20130529005004 20130501005004 42393 nl.netlabs.nl.
    SNKS6Bo8SsqRxDuxF9dRiwqom4YqOArpLAWjv1WHf5fr
    aURdyssZ3V/R8jBRwMNHQNqIQVlDc4i84OsBs2Vpolil
    j0Gy5mfqgnxRCh5b6TtLDE5t4lcFg0k5FgaqtLXCd0an
    f8zdv8nQM/9U0aXgnQLXuUDv4ZpDPXkxPuokKIE= )
3.1.1._dane.nl.netlabs.nl. 10200 IN TLSA 3 1 1 (
    OD1FCBD71686199607A132744A4918FC209565C91FA8
    E9FFEEA0AAFD6B9305F6 )
3.1.1._dane.nl.netlabs.nl. 10200 IN RRSIG TLSA 8 6 10200 (
    20130529005004 20130501005004 42393 nl.netlabs.nl.
    mE8cSI5wCbx4lsQTHoWZTweh1Jo+A0ZDETnNDGKJvafL
    2Q7cMhoqq9J5mvaKFm1MN8qgiaRbt56c90cahFA3xkO3
    loDljLlcUlXpVoRDzWe73MjjyuU76UrsyqNdxmHKB6xR
    mEFxkvcQ5EM6blfDGRHOfnMFV15ezi9GwkB7DcI= )
```

[DNSSEC Tutorial, USENIX LISA 13]

122

DANE/TLSA tools

- TLSA record generation:
 - swede, hash-slinger, ...
 - https://www.huque.com/bin/gen_tlsa
- TLSA validators:
 - Browser enhancements in progress by some
 - Bloodhound Mozilla fork <https://www.dnssec-tools.org/download/>
 - firefox plugin? <http://people.redhat.com/pwouters/>
 - <http://www.internetsociety.org/deploy360/resources/dane/>

[DNSSEC Tutorial, USENIX LISA 13]

123

DNSSEC Deployment Status

[DNSSEC Tutorial, USENIX LISA 13]

124

Deployment status

- DNSSEC Root signed (July 2010)
- Many TLDs signed: 123 of 318 (39%) as of Sept 2013 (112 w/ DS):
 - GTLD: edu gov com net org biz info arpa
 - ccTLD: many, including a number of IDNs
 - See http://stats.research.icann.org/dns/tld_report/
 - Also <http://www.huque.com/app/dnsstat/category/tld/>
- Reverse trees: in-addr.arpa ip6.arpa
- Note: not all TLD registrars support DNSSEC yet (ie. ability to install a DS record in the TLD)

[DNSSEC Tutorial, USENIX LISA 13]

125

Registrar support

- Note: not all TLD registrars support DNSSEC yet (ie. ability to install a DS record in the TLD)
- Situation is gradually improving
- ICANN maintains a list at:
 - <http://www.icann.org/en/news/in-focus/dnssec/deployment>

[DNSSEC Tutorial, USENIX LISA 13]

126

Deployment status

- Below the TLDs is where most of the work remains
- Not so encouraging a picture here, but some pockets have significant deployment ...
- .NL - 1.5 million signed zones!

[DNSSEC Tutorial, USENIX LISA 13]

127

CDNs and Hosting Services

- Content Delivery Networks and DNS hosting services are lagging
- Akamai has announced support:
 - http://www.akamai.com/html/about/press/releases/2010/press_080910.html

[DNSSEC Tutorial, USENIX LISA 13]

128

Validator status

- Measuring the extent of deployment of DNSSEC validating resolvers is much more difficult, but there have been some attempts:
 - <http://validator-search.verisignlabs.com/>
 - <http://www.potaroo.net/ispcol/2012-10/counting-dnssec.html>
 - <http://www.iepg.org/2013-07-ietf87/2013-07-28-dnssec.pdf>

Deployed validators

- Heard at ICANN'45 (Oct 2012): US gov now requiring DNSSEC validation in all systems operated in that space
- Many universities use validation
- Allegedly Mac OS X 10.9 has validation on by default (confirm)
- Comcast (large US ISP) has DNSSEC validation turned on for their customers
- Google public DNS deployed validation in May 2013:
 - <http://googleonlinesecurity.blogspot.nl/2013/03/google-public-dns-now-supports-dnssec.html>

Attention Tutorial Attendees! Please don't forget to fill out your Tutorial Surveys.

Your feedback is very important to us
and helps us shape the future
of the LISA training program.

Please visit www.usenix.org/lisa13/training/survey
and fill out the appropriate surveys.



Thanks for your help!



Thank you!



Shumon Huque
shuque -@- upenn.edu

Reminder: Please fill out the
evaluation forms for this course!

Appendix A

Review of basic DNS

[DNSSEC Tutorial, USENIX LISA 13]

133

DNS

- Domain Name System
- Base specs in RFC 1034 & 1035 (obs 882 & 883)
- Distributed global database
- Indexed by “domain names” (together with a type and class)
- A domain name is a sequence of labels, eg.
 - www.amazon.com.
- Domain Names are case insensitive, but case preserving
- Transport protocol: UDP and TCP port 53

[DNSSEC Tutorial, USENIX LISA 13]

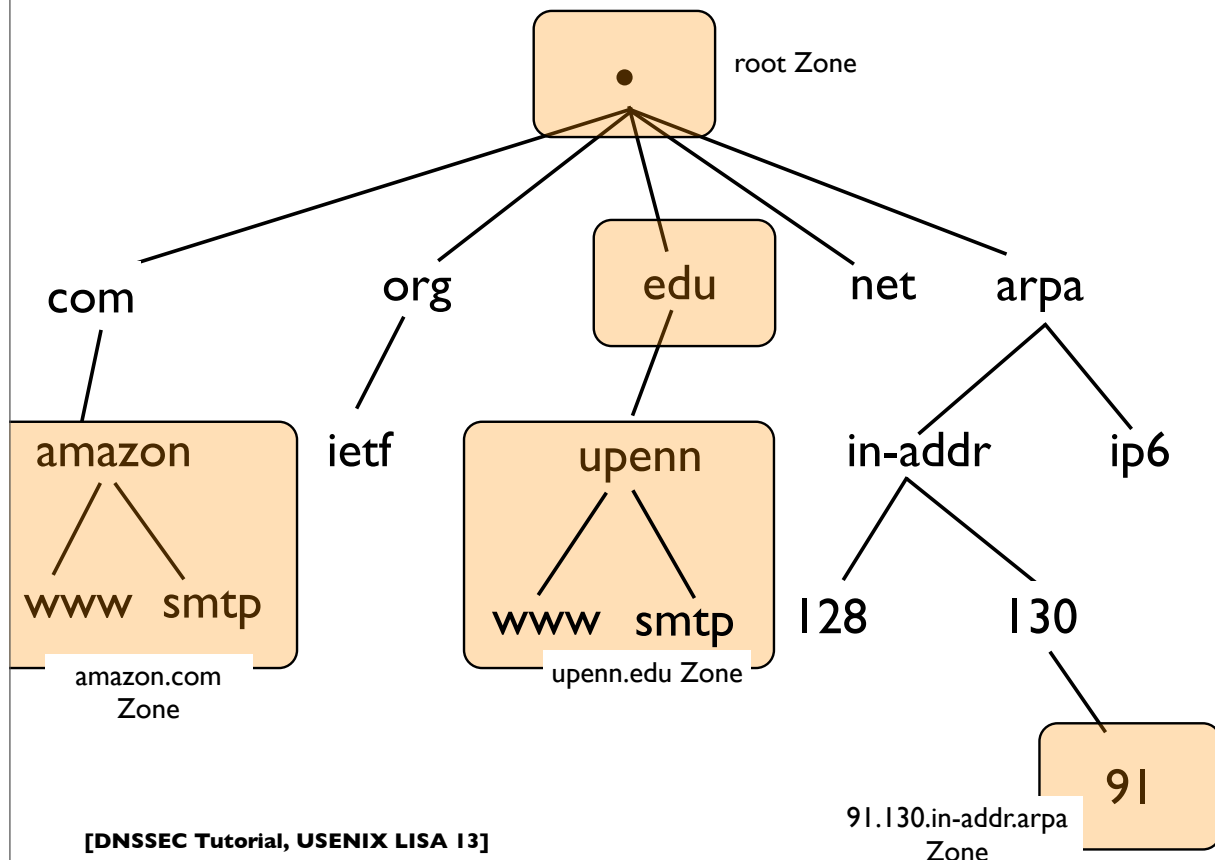
134

DNS

- DNS can be represented as a tree of labels
- Sibling nodes must have unique labels
- Domain name at a particular label can be formed by the sequence of labels traversed by walking up the tree from that label to the root
- Zone - autonomously managed subtree
- Delegations: boundaries between zones

[DNSSEC Tutorial, USENIX LISA 13]

135



[DNSSEC Tutorial, USENIX LISA 13]

136

Root and TLDs

- Root of the DNS (“empty label”)
- Next level of names are called Top Level Domains (TLDs)
- Until recently 3 primary classes of TLDs
 - GTLD: Generic Top Level Domains (.com, .net, .edu, .org etc)
 - CCTLD: Country Code TLD (2 letter codes for each country, eg. .us, .fr, .jp, .de, ...)
 - Infrastructure: eg. .arpa etc (uses: reverse DNS e164, etc)
- IDN cctld (Internationalized domain name ccTLD)
- The new gTLDs - the wild west? (newgtlds.icann.org)

[DNSSEC Tutorial, USENIX LISA 13]

137

DNS main components

- Server Side:
 - Authoritative Servers
 - Resolvers (Recursive Resolvers)
- Client Side:
 - Stub resolvers (usually on DNS client machines)

[DNSSEC Tutorial, USENIX LISA 13]

138

Authoritative Server

- A server that directly serves data for a particular zone
- Said to be “authoritative” for that zone
- These servers are the ones specified in NS records

Resolver

- Aka “Recursive Resolver”, “Cache” etc
- Used by endsystems (stub resolvers) to query (“resolve”) arbitrary domain names
- Receives “recursive” queries from these endsystems
- Resolvers query authoritative servers, following DNS delegations until they obtain the answer they need (this process is called “iterative” resolution)
- Resolvers “cache” (remember) query results for the specified “TTL” (also some negative results are cached)

Stub Resolver

- The DNS client software component that resides on most endsystems
- Commonly implemented by the Operating System as a set of library routines
- Has a configured set of addresses of the Recursive Resolvers that should be used to lookup (“resolve”) domain names
 - usually by manual configuration, or dynamically learned via DHCP
- Some stub resolvers also cache results

[DNSSEC Tutorial, USENIX LISA 13]

141

Stub resolver configuration

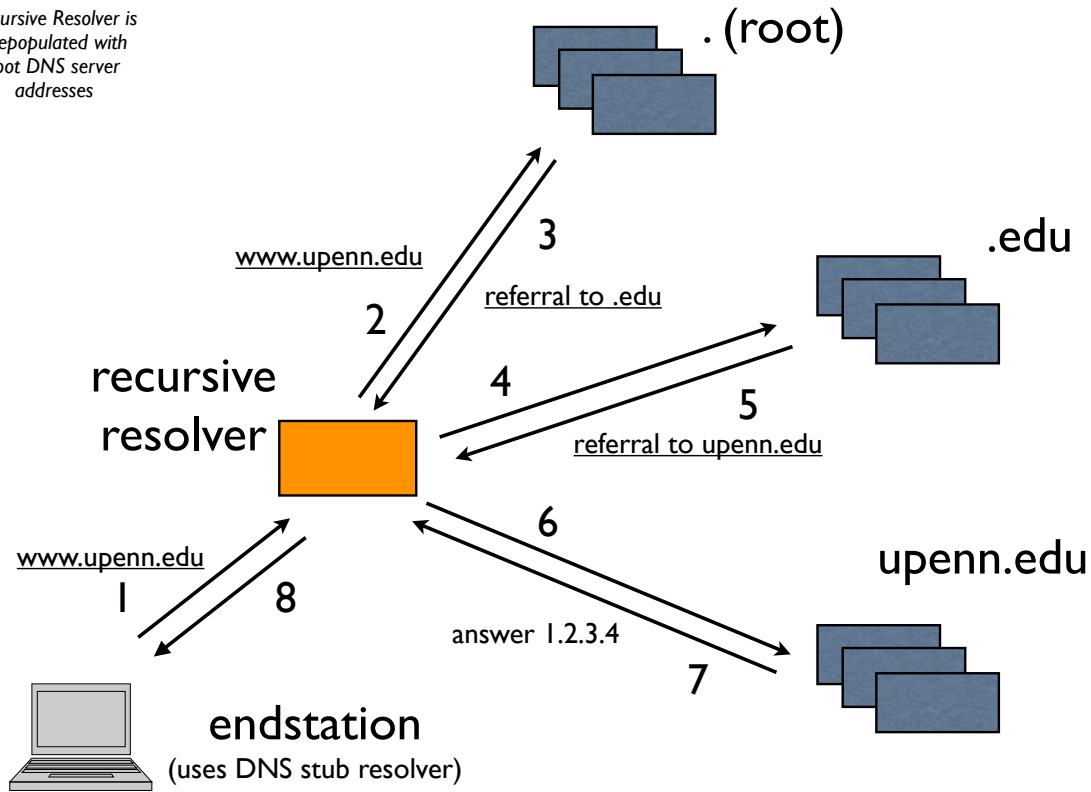
```
$ cat /etc/resolv.conf

search          finance.example.com example.com
;;
nameserver      10.12.3.1
nameserver      10.254.23.71
nameserver      10.15.18.9
;;
options timeout:1 attempts:2 rotate
```

[DNSSEC Tutorial, USENIX LISA 13]

142

Recursive Resolver is
prepopulated with
root DNS server
addresses



[DNSSEC Tutorial, USENIX LISA 13]

143

Parts of a DNS query

- Each DNS query needs a query name, type, and class
- **qname**: a domain name, eg. www.upenn.edu
- **qtype**: A, AAAA, MX, CNAME, PTR, SRV, TXT, NS, SOA, ...
- **qclass**: IN, CH, HS (only “IN” is commonly used)
- Various flags: QR, RD, EDNS Opt, DO etc

[DNSSEC Tutorial, USENIX LISA 13]

144

Life of a typical DNS query

- Type “www.amazon.com” into browser
- Browser calls a name lookup function (eg. `getaddrinfo()`)
- DNS may not be the only name lookup service in use. The lookup function might consult a nameservice switch table to figure out what order of services to consult (eg. `/etc/nsswitch.conf` -- flat file, LDAP, NIS, DNS etc)
- If/when DNS is used, then call DNS specific calls in stub resolver
 - `res_ninit()`, `res_nquery()`, `res_nsearch()`

[DNSSEC Tutorial, USENIX LISA 13]

145

Life of a typical DNS query

- Stub resolver formulates and makes DNS query:
 - qname www.amazon.com, qtype=A, qclass=IN
 - Note: IPv6 enabled resolvers might try AAAA, then A
- Sends query to DNS servers (resolvers) specified in stub resolver configuration (eg. `/etc/resolv.conf`) in the order specified until it gets a successful response, failure, or times out
- If a “search” domain list is configured, on lookup failure, the stub retries queries with domain suffixes from this list appended to the original query

[DNSSEC Tutorial, USENIX LISA 13]

146

Life of a typical DNS query

- DNS resolvers will get the answer:
 - from their authoritative zones if they have any relevant ones
 - from their cache if the answer is already there
 - by iterative queries of the DNS tree, as necessary, eg.
 - root servers, amazon.com servers, ...

[DNSSEC Tutorial, USENIX LISA 13]

147

Resource Records (RR)

- The fundamental unit of data in the DNS database
- A grouping of a {domain name, type, class}, a TTL (time-to-live), and the associated “resource data”
- Has a defined text “presentation format”

```
www.example.com.      86400 IN   A   10.253.12.7
```

*name, or
owner name* *tll* *class* *type* *rdata*

[DNSSEC Tutorial, USENIX LISA 13]

148

Resource Record Sets

- A set of RRs with the same name, class, and type
- The rdata (resource data) associated with each RR in the set must be distinct
- The TTL of all RRs in the set also must match
- RR sets are treated atomically when returning responses

```
www.ucla.edu.      300   IN    A     169.232.33.224
www.ucla.edu.      300   IN    A     169.232.55.224
www.ucla.edu.      300   IN    A     169.232.56.224
```

[DNSSEC Tutorial, USENIX LISA 13]

149

Resource Record types

Type	Description
SOA	marks Start O f a zone of A uthority
NS	NameServer record
A	IPv4 Address record
AAAA	IPv6 Address record
CNAME	Canonical name (ie. an alias)
MX	Mail Exchanger record
SRV	Service Location record
PTR	Pointer (most commonly for reverse DNS)
TXT	Text record (free form text with no semantics)
NAPTR	Naming Authority Pointer Record

[DNSSEC Tutorial, USENIX LISA 13]

for full list, see
www.iana.org/assignments/dns-parameters

150

Other special RRtypes

Type	Description
TSIG	Transaction Signature (RFC 2845)
TKEY	Transaction Key (RFC 2930) - estab secret keys
AXFR	Zone Transfer
IXFR	Incremental Zone Transfer (RFC 1995)
OPT	Opt pseudo RR (RFC 2671 - EDNS0)

[DNSSEC Tutorial, USENIX LISA 13]

for full list, see
www.iana.org/assignments/dns-parameters

151

SOA record

- Defines the start of a new zone; and important parameters for the zone
- Always appears at the apex of the zone
- Serial number should be incremented on zone content updates

```
google.com.      86400 IN SOA ns1.google.com. (
                    dns-admin.google.com.
                    2012042000 ; serial number
                    7200      ; refresh (2 hours)
                    1800      ; retry (30 minutes)
                    1209600   ; expire (2 weeks)
                    300       ; minimum (5 minutes)
                    )
```

[DNSSEC Tutorial, USENIX LISA 13]

152

NS record

- Name Server record: owner is the zone name
- Delegates a DNS subtree from parent (ie. create new zone)
- Lists the authoritative servers for the zone
- Appears in both parent and child zones
- rdata contains hostname of the DNS server

```
upenn.edu.      86400 IN NS noc3.dccs.upenn.edu.  
upenn.edu.      86400 IN NS noc2.dccs.upenn.edu.  
upenn.edu.      86400 IN NS dns2.udel.edu.  
upenn.edu.      86400 IN NS dns1.udel.edu.  
upenn.edu.      86400 IN NS sns-pb.isc.org.
```

[DNSSEC Tutorial, USENIX LISA 13]

153

A record

- IPv4 Address Record
- rdata contains an IPv4 address

```
www.example.com. 86400 IN A 192.0.43.10
```

[DNSSEC Tutorial, USENIX LISA 13]

154

AAAA record

- IPv6 Address Record
- rdata contains an IPv6 address
- Note: there was another record called A6, which didn't catch on, and which has now been declared historic (RFC 6563)

```
www.example.com. 86400 IN AAAA 2001:500:88:200::10
```

[DNSSEC Tutorial, USENIX LISA 13]

155

CNAME record

- An “alias”, ie. maps one name to another (regardless of type)
- Put another way, “this is another name for this name”
- rdata contains the mapped domain name (“canonical name”)
- CNAME records have special rules

```
www.example.com. 86400 IN CNAME worf.example.com.
```

[DNSSEC Tutorial, USENIX LISA 13]

156

CNAME special rules

[from RFC 1034, Section 3.6.2]

>>> CNAME and no other data rule:

A CNAME RR identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR. **If a CNAME RR is present at a node, no other data should be present; this ensures that the data for a canonical name and its aliases cannot be different.** This rule also insures that a cached CNAME can be used without checking with an authoritative server for other RR types.

[Note: there is now an exception to this because of DNSSEC metadata records, which are allowed to appear with CNAMEs]

>>> CNAME special action processing:

CNAME RRs cause special action in DNS software. **When a name server fails to find a desired RR in the resource set associated with the domain name, it checks to see if the resource set consists of a CNAME record with a matching class. If so, the name server includes the CNAME record in the response and restarts the query at the domain name specified in the data field of the CNAME record.** The one exception to this rule is that queries which match the CNAME type are not restarted.

[DNSSEC Tutorial, USENIX LISA 13]

157

CNAME special rules

Illustration of special action processing of CNAMEs:

```
$ dig www.sas.upenn.edu A

;; QUESTION SECTION:
;www.sas.upenn.edu.      IN A

;; ANSWER SECTION:
www.sas.upenn.edu.      300  IN  CNAME  virgo.sas.upenn.edu.
virgo.sas.upenn.edu.    900  IN  A      128.91.55.21
```

[DNSSEC Tutorial, USENIX LISA 13]

158

PTR record

- Pointer record
- The most common use is to map IP addresses back to domain names (reverse DNS mappings)
- IPv4 uses in-addr.arpa, and IPv6 uses ip6.arpa subtrees

IPv4 PTR records

- Uses “**in-addr.arpa**” subtree
- The LHS of the PTR record (“owner name”) is constructed by the following method:
 - Reverse all octets in the IPv4 address
 - Make each octet a DNS label
 - Append “in-addr.arpa.” to the domain name

IPv4 PTR example

```
host1.example.com.    IN    A    192.0.2.17
192.0.2.17           (orig IPv4 address)
17.2.0.192           (reverse octets)
17.2.0.192.in-addr.arpa. (append in-addr.arpa.)
```

Resulting PTR record:

```
17.2.0.192.in-addr.arpa. IN PTR host1.example.com.
```

[DNSSEC Tutorial, USENIX LISA 13]

161

IPv6 addresses

- 128-bits (four times as large)
- 8 fields of 16 bits each (4 hex digits) separated by colons (:)
- [Hex digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f]
- 2^{128} possible addresses (an incomprehensibly large number)

```
2001:0db8:3902:00c2:0000:0000:0000:fe04
```

($2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$)

[DNSSEC Tutorial, USENIX LISA 13]

162

IPv6 addresses

- Zero suppression & compression for more compact format
 - Suppress (omit) leading zeros in each field
 - Replace consecutive fields of all zeros with a double colon (::) - only one sequence of zero fields can be compressed this way

2001:0db8:3902:00c2:0000:0000:0000:fe04



2001:db8:3902:c2::fe04

[DNSSEC Tutorial, USENIX LISA 13]

163

IPv6 PTR records

- Uses “**ip6.arpa**” subtree
- The LHS of the PTR record (“owner name”) is constructed by the following method:
 - Expand all the zeros in the IPv6 address
 - Reverse all the hex digits
 - Make each hex digit a DNS label
 - Append “ip6.arpa.” to the domain name (note: the older “ip6.int” was formally deprecated in 2005, RFC 4159)

[DNSSEC Tutorial, USENIX LISA 13]

164

SRV record

- Service Location record (RFC 2782)
- Allows designation of server(s) providing service for a particular application and transport at a domain name
- Owner name has special form: `_service._transport.<domain>`
- rdata contains priority, weight, port and server hostname
- Some applications using SRV records include: LDAP, Kerberos, XMPP, SIP, Windows Active Directory, ...

[DNSSEC Tutorial, USENIX LISA 13]

167

SRV record

service name	transport	priority	weight	port	server name
<code>_ldap._tcp.example.com</code>		600	IN	SRV 1 0 389	<code>ldap1.example.com</code>
<code>_ldap._tcp.example.com</code>		600	IN	SRV 2 1 389	<code>ldap2.example.com</code>
<code>_ldap._tcp.example.com</code>		600	IN	SRV 2 2 389	<code>ldap3.example.com</code>
<code>_ldap._tcp.example.com</code>		600	IN	SRV 2 1 289	<code>ldap4.example.com</code>

- Priority defines the order in which to query servers (lower number = higher priority)
- Weight defines the proportion in which to send queries to servers at the same priority level (load distribution)

[DNSSEC Tutorial, USENIX LISA 13]

168

TXT record

- free form descriptive text strings, with no defined semantics
- Although some applications have defined their own meanings (eg. DKIM, SPF, ...)
- rdata: one or more character strings

```
blah.example.com. 300 IN TXT "Hello World" "Goodbye"
```

[DNSSEC Tutorial, USENIX LISA 13]

169

NAPTR record

- Naming Authority Pointer Record (RFC 3403 - DDDS)
- Very complex record, and induces additional complex processing on resolver (lookup and rewrite)
- Uses: URL resolver discovery service, E164, SIP, ...

```
*.freenum IN NAPTR (100 10 "u" "E2U+sip"  
"!^\\+*([^\\*]*)!sip:\\1@sip.magpi.org!" .)
```

[DNSSEC Tutorial, USENIX LISA 13]

170

Wildcards

- RRs with owner names starting with the label “*” (asterisk)
- When the wildcard is *matched*, the DNS server returns a response with:
 - query name returned as owner name
 - rest of RR content taken from the wildcard record

```
mail.example.com.    300  IN  A    10.1.1.1
www.example.com.    300  IN  A    10.1.1.2
*.example.com.      300  IN  A    10.1.1.7
```

```
Here, query for blah.example.com returns:
blah.example.com.   300  IN  A    10.1.1.7
```

[DNSSEC Tutorial, USENIX LISA 13]

171

ANY query type

- A pseudo record type used in DNS queries only
- Used to match any record type for the queried domain name
- Server will return all records of all types for that domain name that it possesses (note: caches may return incomplete data; to obtain all data for the name, you need to issue ANY query to authoritative servers)
- For debugging and troubleshooting purposes only; *do not use in production code*

[DNSSEC Tutorial, USENIX LISA 13]

172

Master Zone file format

- RFC 1035, Section 5 for details
- Entries in the master zone file are DNS resource records in their textual “presentation format”

[DNSSEC Tutorial, USENIX LISA 13]

173

Zone file example

Zone: example.com

```
@           3600 IN SOA master.example.com. hostmaster.example.com. (
                                1001514808 ; serial
                                10800      ; refresh (3 hours)
                                3600       ; retry (1 hour)
                                604800    ; expire (1 week)
                                3600      ; minimum (1 hour)
                                )
            86400 IN NS      ns1.example.com.
            86400 IN NS      ns2.example.com.
            86400 IN MX      10 mail1.example.com.
            86400 IN MX      20 mail2.example.com.
ns1         86400 IN A       10.1.1.1
ns2         86400 IN A       10.1.1.2
www         900   IN A       10.1.2.2
mail1      3600 IN A       10.3.3.3
mail2      3600 IN A       10.3.3.4
```

[TCOM 504, Spring 2012]

174

Master Zone file format

@ Denotes current origin; defaulting to zone name
Appended to any domain name not ending in a period.
() Parens used to group data that crosses a line boundary
; Starts a comment
\$ORIGIN Resets the origin for subsequent relative names

RRs beginning with whitespace implicitly inherit last owner name.
TTL and Class fields are optional (default to last explicitly stated)

Extensions usable in BIND master files:

\$TTL Define TTL parameter for subsequent records
\$GENERATE Programmatically generate records, eg.
eg. \$GENERATE 10-90 client-\$ A 10.4.4.\$
\$GENERATE 0-62 blah-#{0,3,x} A 192.168.154.#{+64,0,d}

[DNSSEC Tutorial, USENIX LISA 13]

175

Size restrictions

- Label: 63 octets max
- Domain Name: 255 octets max
- TTL: positive signed 32-bit integer
- Entire DNS message: 512 bytes (UDP) - plain DNS
- Messages larger than 512 bytes requires:
 - Use of TCP (often truncated UDP response followed by TCP retry)
 - EDNS0 - a DNS extension mechanism allowing negotiation of larger UDP message buffers

[DNSSEC Tutorial, USENIX LISA 13]

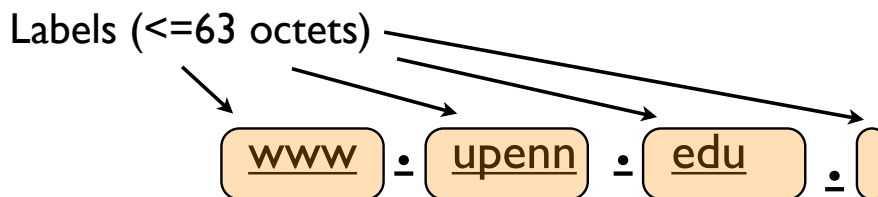
176

Textual vs wire format

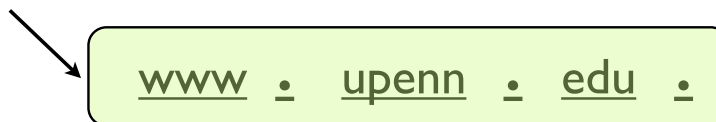
- The human readable “textual representation” or “presentation format” of a domain name is different from the the domain name as it actually appears in DNS protocol messages (“on the wire” or “wire format”)
- Text format: labels written in ASCII delimited by periods
- Wire format: label bytes one after the other, always ending with the empty label. each label is composed of a label length followed by the label bytes

[DNSSEC Tutorial, USENIX LISA 13]

177



Domain Name (<= 255 octets)



Wire format of this domain name:

(hex) 03777777057570656e6e0365647500

4 Component labels:

www	0x	03	77	77	77
upenn	0x	05	75	70	65 6e 6e
edu	0x	03	65	64	75
.	0x	00			

label length in 1st octet (lower 6-bits)

[DNSSEC Tutorial, USENIX LISA 13]

178

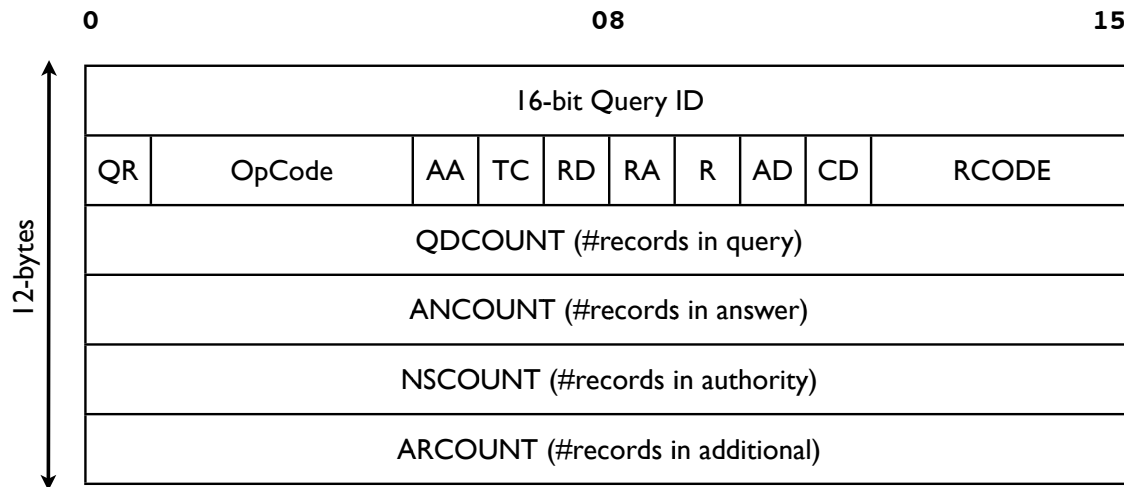
IDNs and punycode

- IDN: Internationalized Domain Name
- Uses an ASCII encoding called “Punycode” to represent non-english characters in domain names
- See RFC 3492: Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)
 - `xn--80ao21a.` (A Kazakh TLD)

DNS Packet Format

DNS Header (12 bytes)
Question Section
Answer Section
Authority Section
Additional Section

DNS Header



[DNSSEC Tutorial, USENIX LISA 13]

181

DNS Header

QR: set to 1 in DNS response messages

OpCode:

- 0 Standard Query
- 1 Inverse Query (deprecated)
- 2 Status request (undefined and unused?)
- 4 Notify
- 5 Update
- 3,6-15 Undefined

- AA** = Authoritative answer (ie. not from cache)
- TC** = message was truncated (exceeded 512 byte UDP limit)
- RD** = Recursion desired
- RA** = Recursion available
- R** = Reserved/Unused
- AD** = Authenticated Data (DNSSEC)
- CD** = Checking Disabled (DNSSEC)

[DNSSEC Tutorial, USENIX LISA 13]

182

DNS Response Codes

Common Response codes:

0	NOERROR	No Error
1	FORMERR	Format Error
2	SERVFAIL	Server Failure
3	NXDOMAIN	Not existent domain name
4	NOTIMPL	Function not implemented
5	REFUSED	Query Refused, usually by policy

Used by DNS Dynamic Update (RFC 2136):

6	YXDomain	Name Exists when it should not
7	YXRRSet	RR Set Exists when it should not
8	NXRRSet	RR Set that should exist does not
9	NotAuth	Server not authoritative for zone
10	NotZone	Name not contained in zone
11-15	Unassigned	

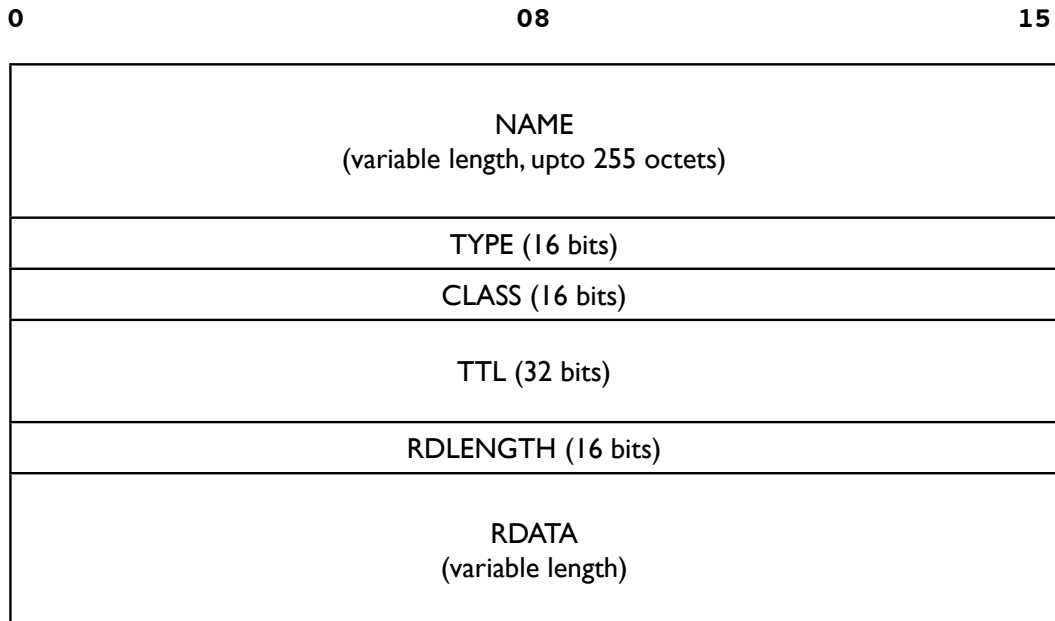
Extended RCodes

Extended RCODES do not appear in the DNS header (since there isn't enough space there). They instead appear in the OPT pseudo RR, which has a special format designed to accommodate them.

Extended RCodes used by EDNS0, TSIG, TKEY, etc:

16	BADVERS	Bad OPT version
16	BADSIG	TSIG Signature Failure
17	BADKEY	Key not recognized
18	BADTIME	Signature out of time window
19	BADMODE	Bad TKEY Mode
20	BADNAME	Duplicate Key Name
21	BADALG	Algorithm not supported
22	BADTRUNK	Bad Truncation

DNS RR common format



[DNSSEC Tutorial, USENIX LISA 13]

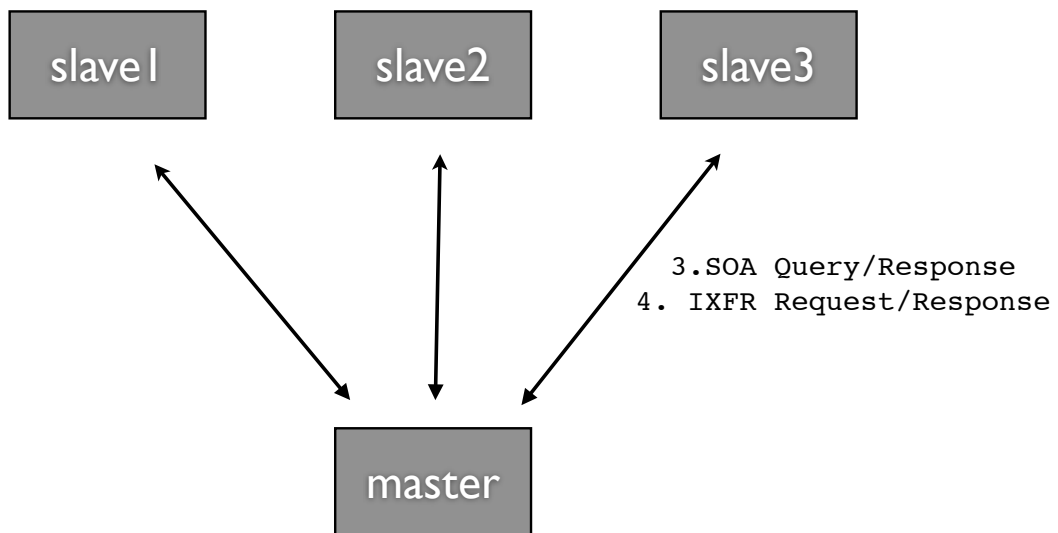
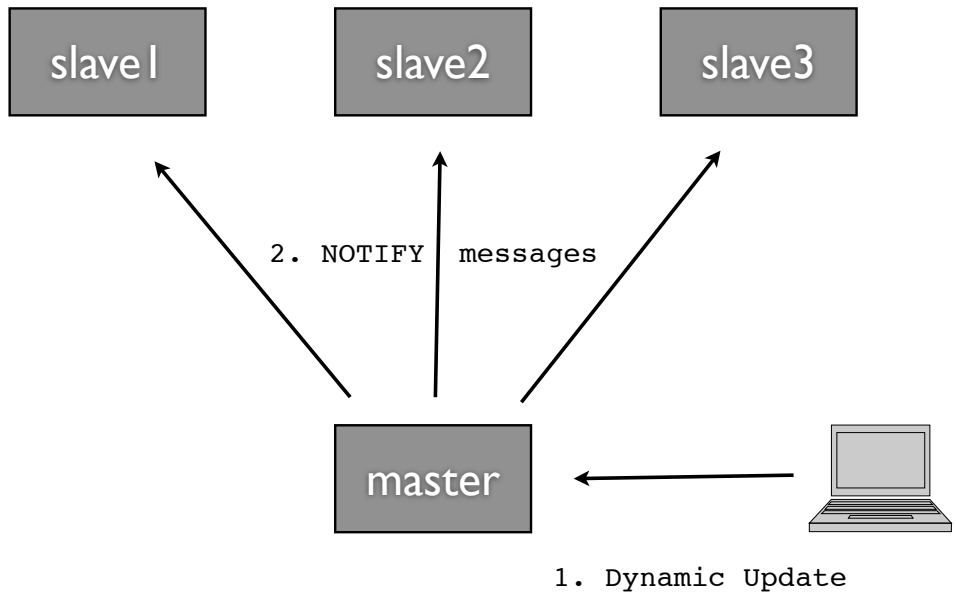
185

Zone Data Synchronization

- Authoritative server operators can synchronize zone data on their servers in a number of ways
- However, DNS provides a way to do this using the DNS protocol itself: **Zone Transfers**, and it's widely used
- Full zone transfers: **AXFR**: slaves send period transfer requests to masters (SOA refresh interval)
- Incremental zone transfers: **IXFR**, usually in combination with the NOTIFY mechanism (see RFC 1995 and 1996)
 - Commonly used in conjunction with Dynamic Update
- A good idea to authenticate zone transfers with TSIG

[DNSSEC Tutorial, USENIX LISA 13]

186



Zone Delegation

- Decentralized administration of DNS subtrees
- Delegations cause new zones to be created, that are (typically) served by different servers, run by different people
- Boundaries between zones (sometimes called zone cuts)
- An NS record set is needed in both the parent and child zones; these indicate the delegation, and the set of new nameservers involved in serving the child zone
- “Glue records” may be needed in the parent zone in order to find the addresses of the servers

[DNSSEC Tutorial, USENIX LISA 13]

189

Zone Delegation

Example of delegation of google.com in .com zone:

```
;; NS Record Set for google
google.com.      172800 IN NS  ns2.google.com.
google.com.      172800 IN NS  ns1.google.com.
google.com.      172800 IN NS  ns3.google.com.
google.com.      172800 IN NS  ns4.google.com.
```

```
;; Glue records for google nameservers
ns2.google.com.  172800 IN A   216.239.34.10
ns1.google.com.  172800 IN A   216.239.32.10
ns3.google.com.  172800 IN A   216.239.36.10
ns4.google.com.  172800 IN A   216.239.38.10
```

The glue records in the .COM zone are needed because the google DNS servers are inside the child google.com zone, otherwise they couldn't be found.

[DNSSEC Tutorial, USENIX LISA 13]

190

