

DNSSEC 2010

Internet Systems
Consortium

Version 2.0



About the Instructor

- Peter Loshner

- email: ploser@isc.org

- Office: +1-650-423-1313

About ISC

- Internet Systems Consortium, Inc.
 - Headquartered in Redwood City, CA
 - 501(c)(3) Nonprofit Corporation
- ISC is a public benefit corporation dedicated to supporting the infrastructure of the universal connected self-organizing Internet — and the autonomy of its participants — by developing and maintaining core production quality software, protocols, and operations.

About ISC

- Engineering
 - Software Development
- Support
 - BIND and ISC DHCP
- Operations
 - F-Root, Secondary Name Service (SNS), Security Internet Exchange (SIE)

DNSSEC 2010

DNSSEC 2010

- Unfortunately, to be able to talk about "DNSSEC 2010", we need to understand "DNS 1983" first.
- To further that understanding, here's a brief overview...

DNS 1983

Overview

What is DNS?

- Globally distributed, loosely coherent, scalable, reliable, dynamic database
- Comprised of three components
 - "Name Space"
 - Servers making that space available
 - Resolvers (clients) which query the servers about the namespace

So, What is DNS?

- When asked for the address of `www.isc.org`, the distributed database known as DNS provides the answer:

`www.isc.org` → `204.152.184.88`

- The database also contains the inverse "mapping"

What is DNS?

- DNS can contain additional information as well:
 - Addresses (ipv4, ipv6, geographic)
 - Security key information
 - Mail exchange information
 - Text records
 - Entertaining host information

What is DNS?

- DNS also provides negative results:

`mmm.isc.org` → NXDOMAIN

- No such object in the database

What is DNS?

- NOTE:
 - No such object in the database is different from
 - No data of requested type for that object

`www.isc.org hinfo` → `null, NOERROR`

Globally Distributed

- Data is maintained locally but retrievable globally
 - No single point of failure
- DNS lookups (queries) can come from any device
- Remote DNS data is locally cacheable to improve performance

Loosly Coherent

- Database is always internally consistent
 - Every component (zone) has a serial number
 - Serial numbers are incremented on data change
- Changes to the master database are replicated to slave servers
- Cached data expires according to timeouts set by administrators

Scalable

- No limit to the size of the database
- No limit to the number of queries
- Queries are automatically distributed between master, slave and cache servers

Reliable

- Data is replicated
 - Masters transfer data to slaves
- Clients can query master or slave servers
- DNS uses UDP or TCP
 - DNS protocol deals with UDP retransmission, sequencing, etc.

Dynamic

- Database can be updated dynamically
 - Add/delete/modify any record
- Modification of master triggers replication
 - Only master can be dynamically updated

Name Space

Name Space

- Hierarchical name space
- Names chosen based on
 - Location
 - Unit
 - Object

Name Space

- Fully qualified domain name (FQDN)

`www.isc.org.`

- DNS provides mapping from the FQDN into "resource records"
- Names are used as database keys

Domain Names

- A domain name consists of one or more labels or words
 - Labels are separated by "."
 - Labels have at most 63 characters
- A FQDN has at most 255 characters, including dots

Resource Records

- The DNS maps those names into data called Resource Records (RRs)

`www.isc.org. A → 204.152.184.88`

- This name has this address

Name Servers

Name Servers

- Name Servers answer DNS queries
- Server Types
 - Authoritative
 - master (primary)
 - slave (secondary)
 - (Caching) recursive servers
 - Mixed function

Resolvers

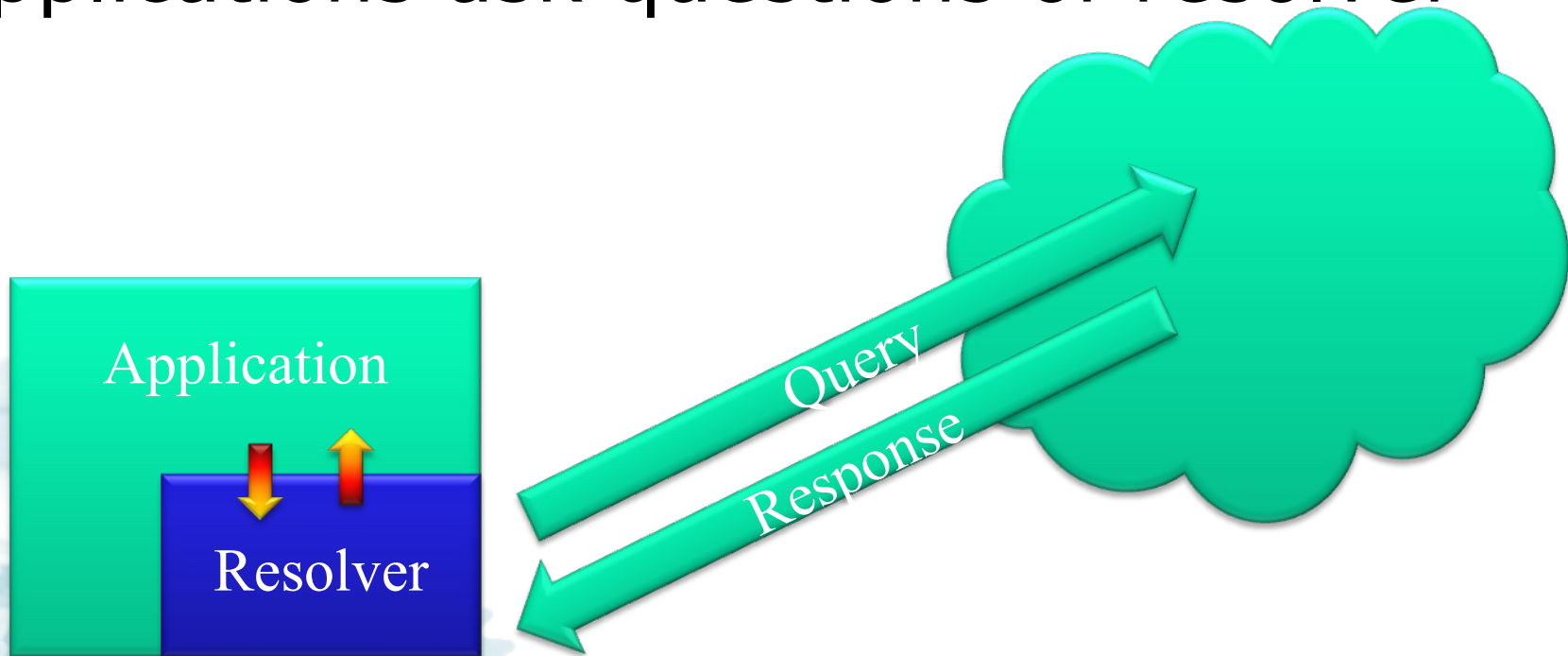
Resolvers

- Resolvers send queries to the DNS system on behalf of applications
- Normally implemented in system libraries (`libc`):

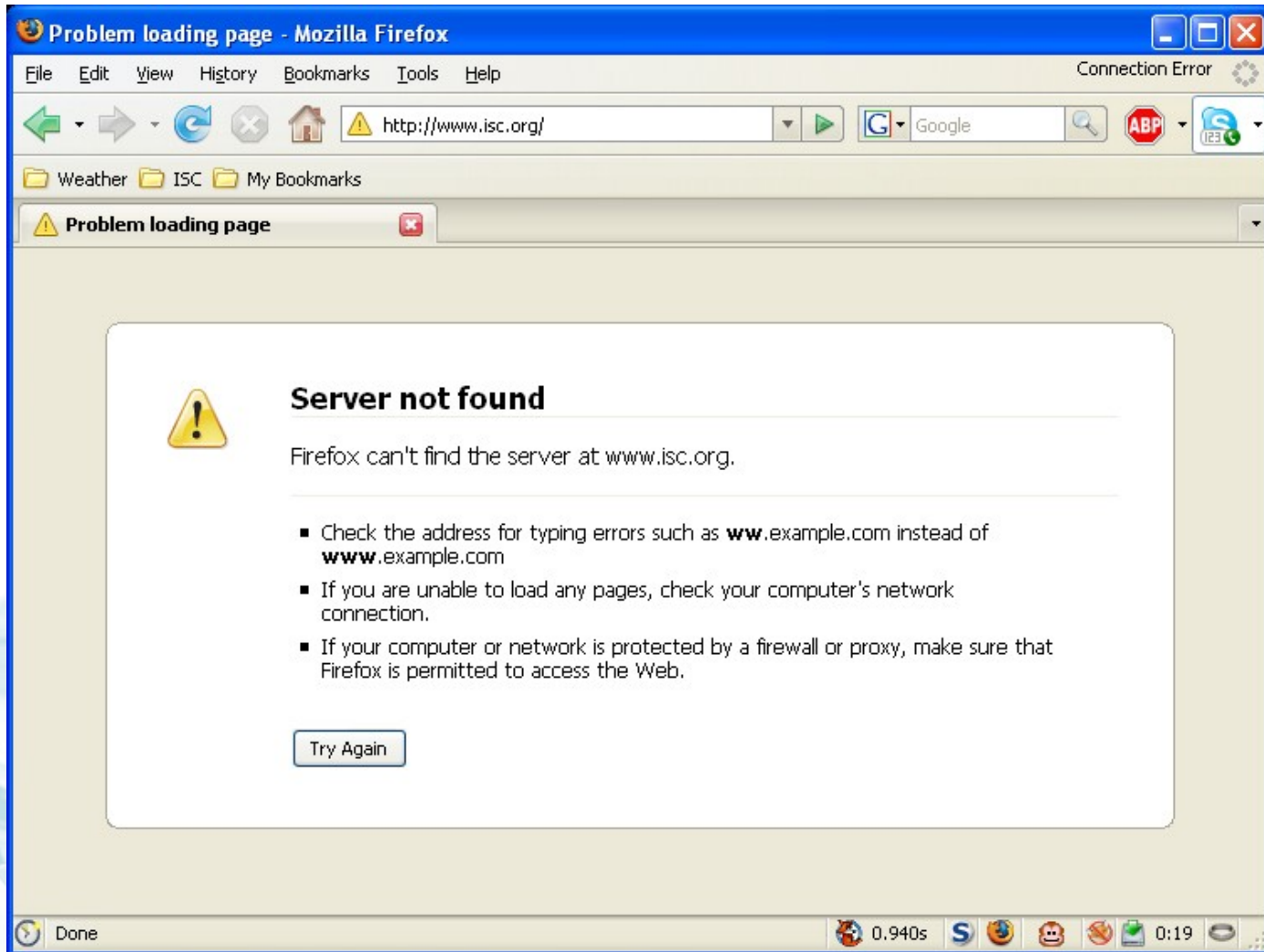
```
getnameinfo([...]); and getaddrinfo([...]);  
formerly  
gethostbyname([...]); and gethostbyaddr([...]);
```

Resolvers

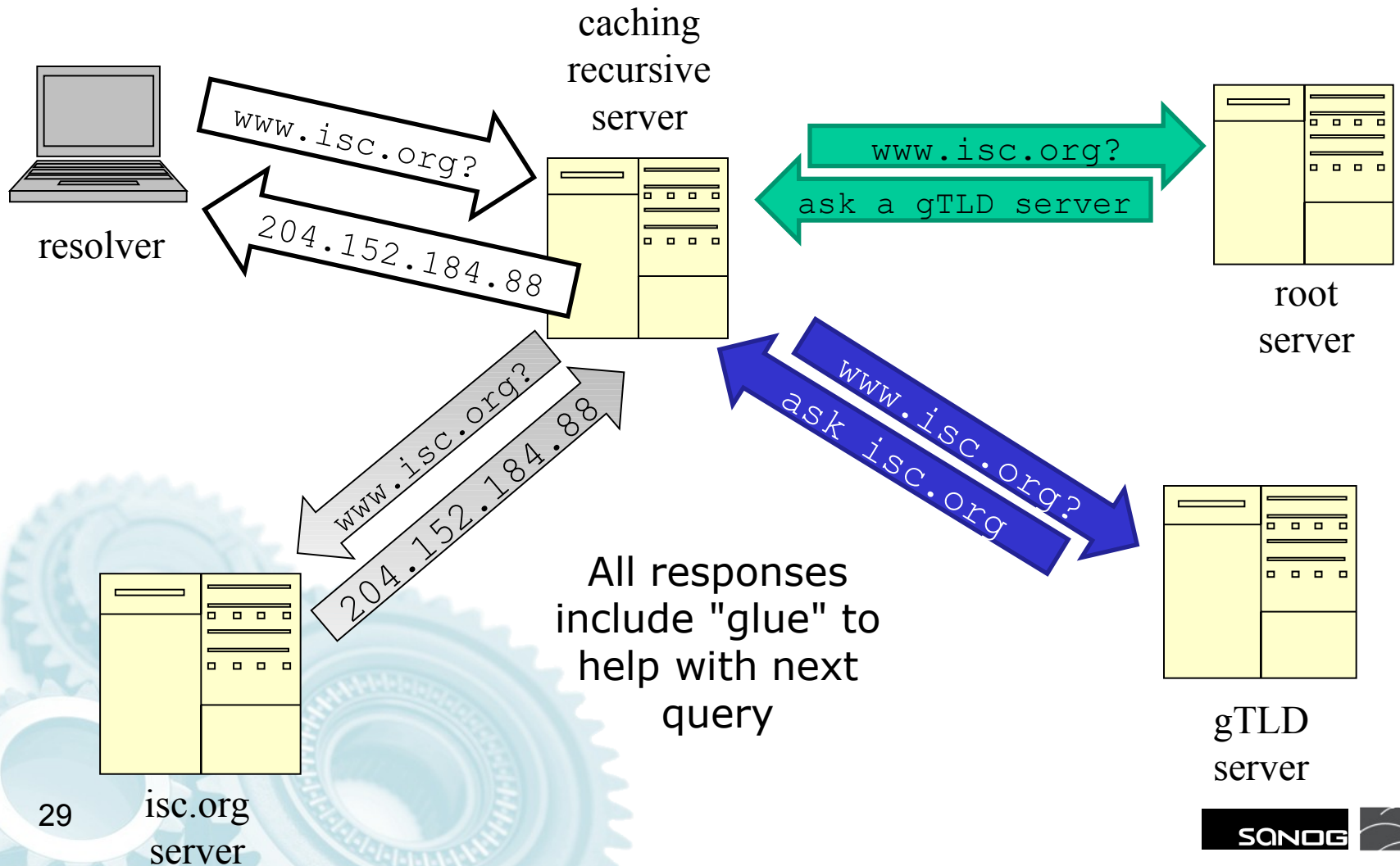
- Resolver – DNS Client
- Applications ask questions of resolver



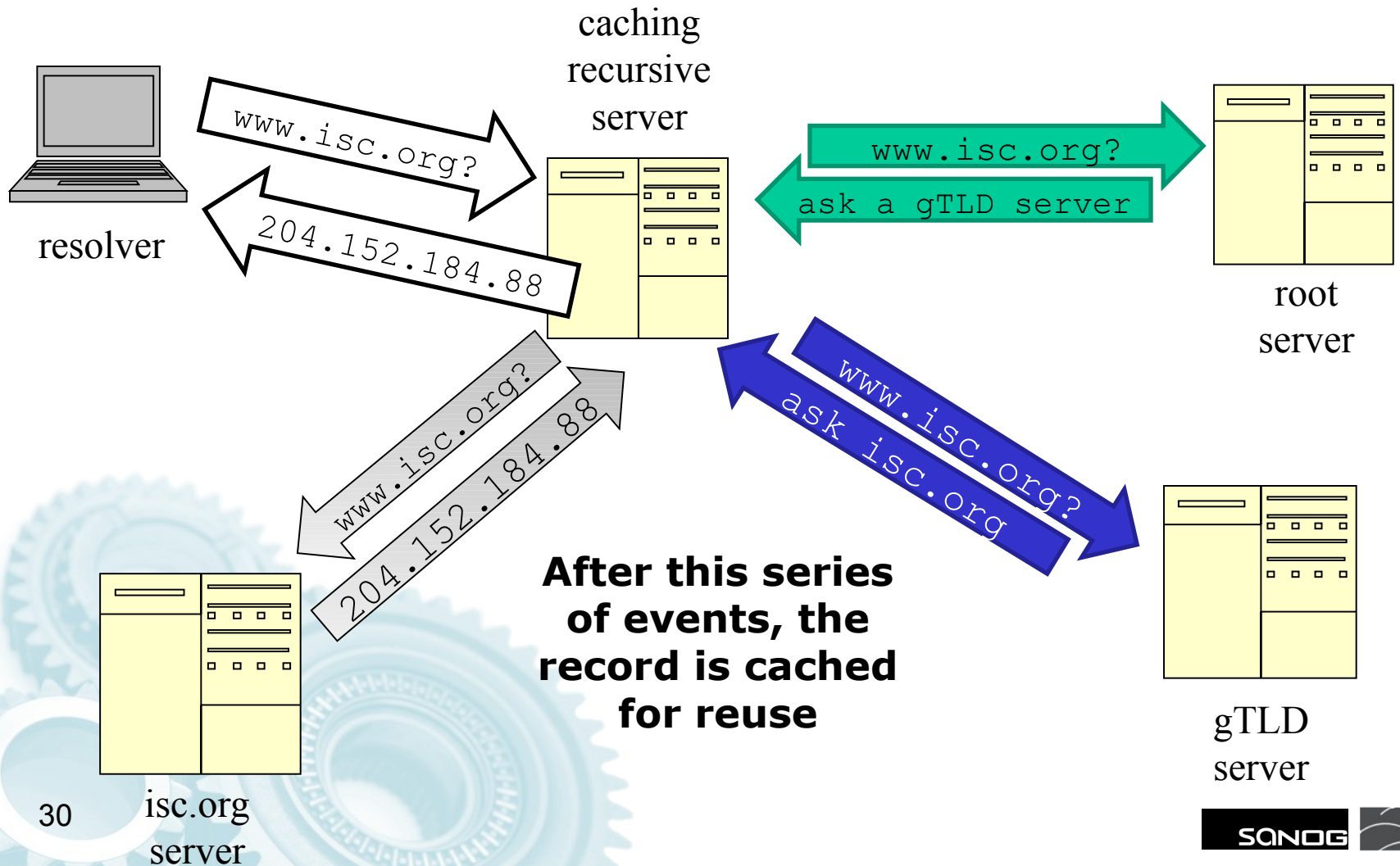
Resolving DNS queries



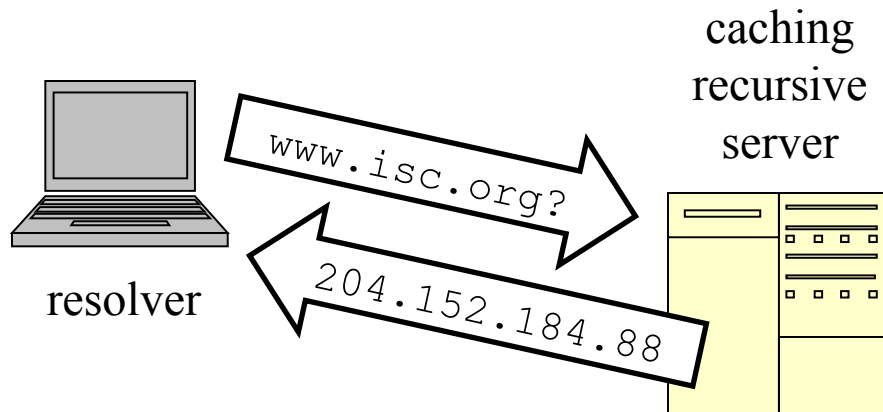
Resolving DNS queries



Resolving DNS queries



Resolving DNS queries



Cached answer
returned with no need
for external queries



Things to note

- At no point is any validation done as to the legitimacy of the data that is being offered by the upstream server.
- That is the role of DNSSEC
- Welcome to 2010...

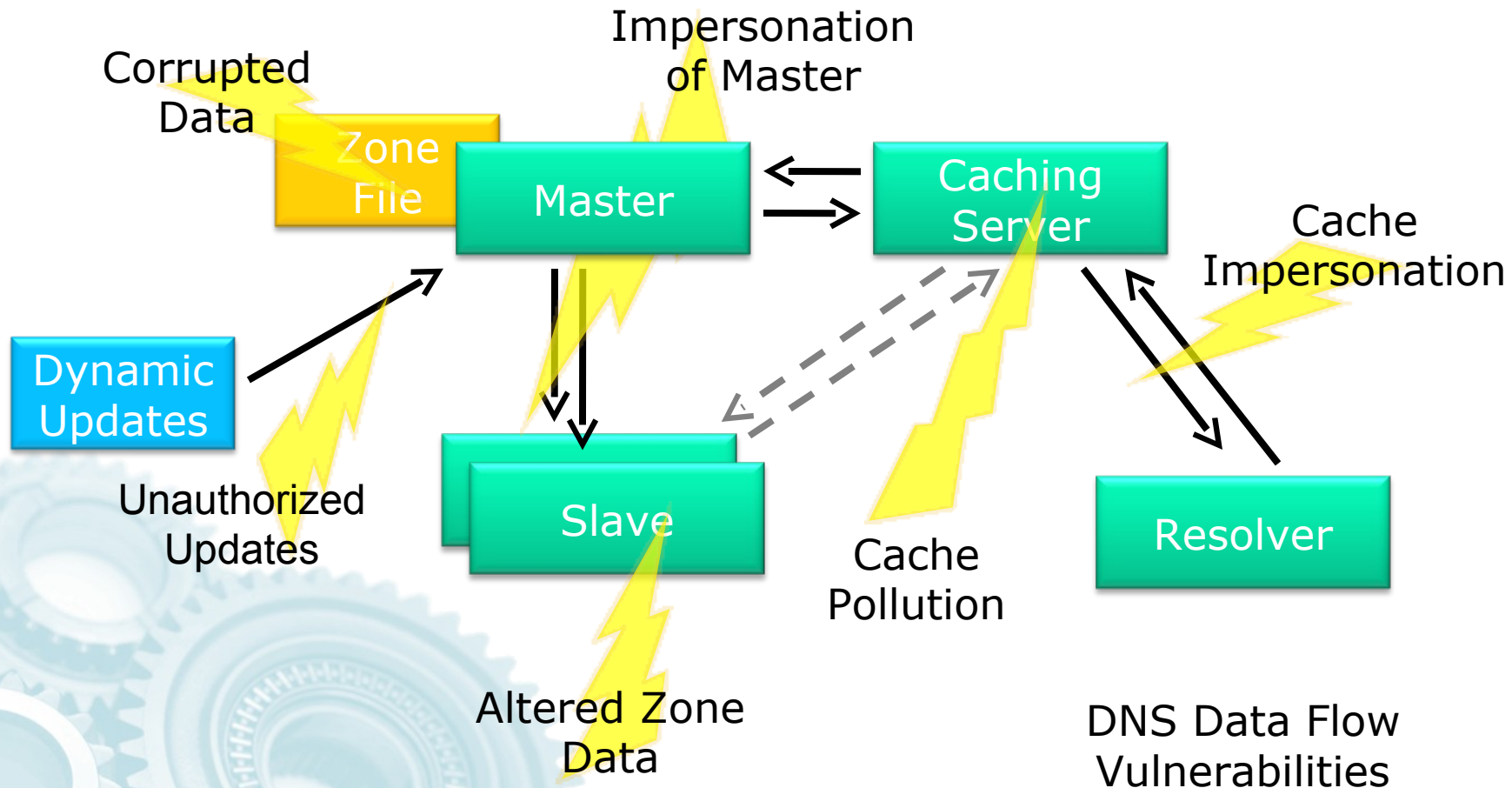
DNSSEC

Introduction

Introduction

- Contemplate for a moment the amount of trust that we put into the DNS infrastructure
- If DNS were to suddenly become unreliable or untrustworthy, what would the result be?

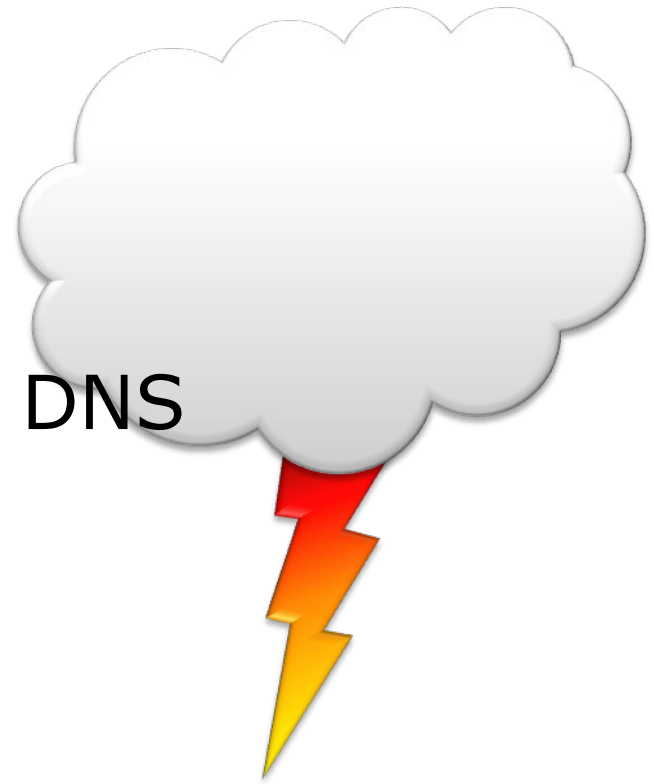
Introduction



DNS Data Flow Vulnerabilities

Introduction

- Threat model
 - There are bad guys
 - What can be done from a DNS perspective?
 - What should we NOT do?



Introduction

- DNSSEC Design Goals
 - Interoperability with existing infrastructure
 - Allow gradual phase-in
 - No "flag days"



Introduction

- With millions of recursive, caching servers on the Internet...
 - Each one needs to be able to be able to look up data from millions of zones
 - There is no way to distribute secret keys
 - Existing technology (TSIG) did not scale well

Introduction

- Central concept:

DNS data is augmented by a signature

- Validating resolver can use the signature to verify that the data is authentic

Introduction

- DNSSEC is based on public key (asymmetrical) cryptography
 - Private key is used to sign DNS data
 - Public key is published via DNS so that validators can retrieve it
 - The public key is then used to validate the signatures, and there-by, the DNS data

Introduction

- DNSSEC provides cryptographic proof that the data received in response to a query is un-modified
- It does not deal with validating dynamic updates, nor with master to slave data transfers

Introduction

- Clients using validating resolvers get guaranteed "good" data
 - For some value of "guaranteed"
- Data that does not validate provides a "**SERVFAIL**" response from the upstream resolver

Introduction

- DNSSEC enabled authoritative servers provide digital signatures across RRsets in addition to "standard" DNS data
- DNSSEC validating resolvers provide authenticated responses with proven integrity

Applications

- Applications can be written to be DNSSEC aware
- But most won't be, and legacy applications will be around for a long, long time

Applications

- Three paths to application DNS security:
 1. DNSSEC unaware application queries as normal and validating recursive server performs validation and shields application from bad data
 2. DNSSEC aware application queries for data with DNSSEC OK (DO) bit set and does validation of responses
 3. DNSSEC aware application trusts validating server, thus trusting status indicated by validator

Applications

- There are four possible “answers”:
 1. Secure – Data validates and a linkage can be traced back to a trust-anchor
 2. (good, but) Insecure – A zone key and RR signature is present and allows a name server to validate the data, but there is no secure link to a trust-anchor
 3. Bogus – A trusted anchor and RR signatures exist, but the data fails to validate
 4. Indeterminate – There is no zone key for the given zone, we can't prove anything

Applications

- Unfortunately, at this time, there is not a validating stub-resolver available
- The “last hop” is difficult
- Discussion is currently going on regarding the need for standardization

Trust Validation

- With this knowledge, we are able to prove that data hasn't changed between the authoritative server and the validator, but how do we know we can trust it?
- Since the root (".") is now signed, that's easy, right?...

Trust Validation

- DNSSEC is based on chains of trust
- At the top of each chain is a "trust-anchor"
 - One (signed) root, one trust-anchor
 - While the root is signed, it's (still) not so easy
 - Trust anchors must be gathered and added to DNS configuration

Chain of Trust

- Once a "trust anchor" is inserted, how does it actually create trust that leads down the DNS tree?
- BIND trust anchors consist of the public portion of the key used to sign the key that signs data in a given zone

Chain of Trust

- The public portion of the Zone Signing Key is used to validate data within the given zone.
- The public portion of the Key Signing Key is used to validate the Zone Signing Key.

Chain of Trust

- Delegation of signed zones include a new Resource Record type
 - Delegation Signer – DS
 - Hash of the public portion of the child's Key Signing Key

Chain of Trust

- If the DS record in the parent is signed using the parent's key, we know that the DS record is valid.
- If the hash of the child's Key Signing Key record matches the DS record then we know that the Key Signing Key is valid.

Chain of Trust

- If the Key Signing Key is known to be valid, its signature of the Zone Signing Key proves that the Zone Signing Key is valid.
- If the Zone Signing Key is known to be valid, it can be used to validate other RRs in the zone.

Chain of Trust

- A living example:

www.isc.org

The following slides created using Sandia National Laboratories "DNSViz"

<http://dnsviz.net/>

Trusting isc.org

- .org contains:
 - KSK 21366
 - ZSK 05919
 - Signed w/21366
 - isc.org DS records
 - signed w/ 05919

Trusting isc.org

- isc.org contains:
 - KSK 12892
 - Hashed into DS
 - ZSK 18516
 - Signed w/ 12892
 - SOA, AAAA, A
 - Signed w/ 18516

DNSSEC – Trusting isc.org

- With a trust anchor for .org: we can trust anything below it that is signed
 - And that has DS records in place

Deploying DNSSEC

Deploying DNSSEC

- One-time activities:
 - Clarify authoritative server directory structure and zone file naming
 - Enable DNSSEC on authoritative servers
 - Enable DNSSEC on recursive servers

Deploying DNSSEC

- DNSSEC enable each zone
 - Generate ZSK and KSK
 - Include keys into zonefile
 - Sign the zone
 - Point `named.conf` at the signed zonefile
 - Notify `named` of the configuration changes

Deploying DNSSEC

- Provide parent zone with DS records
- In the case of a DNSSEC unaware parent, provide DLV registry with DLV records

All of those steps...
in detail!

Prepare directory structure

- Tools are available that make zone maintenance easy but they work best with a standardized directory structure
- Put all files for a zone into a single directory

Enable authoritative servers

```
options {  
    dnssec-enable yes;  
};
```

- Requires BIND to have been built on a system with OpenSSL libraries available

Enable recursive servers

```
options {  
    dnssec-enable yes;  
    dnssec-validation yes;  
};
```

- Validation is done on the recursive, not authoritative servers.

Bigger Responses.

With the additional data required for DNSSEC, responses are going to be larger than 512 bytes which is the traditional size of a packet.

```
% dig +short +dnssec www.isc.org AAAA @sfba.sns-pb.isc.org
2001:4f8:0:2::d
AAAA 5 3 600 20100409074748 20100310074748 2658 isc.org.
TUYn3MwAqApLmZ8BT15b1gaXvhmLxWf0KdT13nz3+aESfLLA0yL1jaUW
HauUnsh0meYCRGIZ080VT5EerSW+B422ItLdqPyKP1IA2RvsoSyykQKa
MwHqD9LXFSSMUCgt7Hwr7nBrshtMkeUnUVDuEZ9VAXGf4IgNn/ZJoaKr RBU=
[...]
;; MSG SIZE rcvd: 1245
```

Bigger Responses (cont.)

EDNS (**Extension Mechanisms for **DNS**) was developed by the IETF to allow larger sized UDP packets (from 512-4k) to carry this additional information, and EDNS0 (RFC 2671) was designated for DNSSEC.**

Without this, resolvers would fall back to using TCP which would cause resource issues on the authoritative servers.

How can you test?

Testing for EDNS0 Support

You can check to see if you are behind such a firewall/device by using the DNS-OARC reply-size tester @ <https://www.dns-oarc.net/oarc/services/replysizetest>

```
% dig +short rs.dns-oarc.net txt
rst.x3827.rs.dns-oarc.net.
rst.x3837.x3827.rs.dns-oarc.net.
rst.x3843.x3837.x3827.rs.dns-oarc.net.
"202.144.128.214 DNS reply size limit is at least 3843"
"Tested at 2010-07-21 01:02:57 UTC"
"202.144.128.214 sent EDNS buffer size 4096"
```

Testing for EDNS0 Support

Here is a test result from a resolver that doesn't have EDNS support:

```
% dig +short rs.dns-oarc.net txt
rst.x476.rs.dns-oarc.net.
rst.x485.x476.rs.dns-oarc.net.
rst.x490.x485.x476.rs.dns-oarc.net.
"210.130.137.21 DNS reply size limit is at least 490"
"210.130.137.21 lacks EDNS, defaults to 512"
"Tested at 2010-03-14 00:44:45 UTC"
```

Securing a Zone

- For each zone, two keys are created
 - 1) Zone Signing Key – used to sign the resource records within the zone
 - 2) Key Signing Key – used to sign the Zone signing key and to create the "Secure Entry Point" for the zone

dnssec-keygen

- Used to generate keys
 - Symmetric and asymmetric
 - Transaction Signatures (TSIG) uses symmetric
 - DNSSEC uses asymmetric

dnssec-keygen

- If this were 2009, `dnssec-keygen` would need to be told:
 - Algorithm to use
 - The size of the key (bits)

dnssec-keygen

- Since this is 2010 (BIND 9.7), we have good defaults:

```
dnssec-keygen zonename
```

- Default algorithm: RSASHA1
- Default ZSK length: 1024 bits
 - Previously had to specify algorithm, # bits, etc.

Create the Keys

- Creating the ZSK

```
dnssec-keygen zonename
```

- Creates 2 files

`Kzonename+<alg>+<id>.key`

`Kzonename+<alg>+<id>.private`

`.key` is public portion of the key

`.private` is private portion of the key

Create the Keys

- Creating the KSK:

```
dnssec-keygen -f KSK zonename
```

- Defaults:

- Uses the RSASHA1 algorithm
- 2048 bits in length
- Has the Secure Entry Point (KSK) bit set

Sign the Zone

- Adding the RRSIG, NSEC and associated records to the zone:

```
dnssec-signzone -S zonefile
```

- This assumes that your `zonefile` is named after the zone that it contains..

Sign the Zone

```
dnssec-signzone -S zonefile
```

- Output file is `zonefile.signed`
 - Sorted in alphabetical order
 - ZSK and KSK automatically included
 - RRSIG and NSEC RRs created
 - Much larger than before
 - Serial Number NOT MODIFIED!

Signed Zone

- The signed zone file differs from its input in a number of ways:
 - The file is sorted in alphabetical order by host name
 - Authoritative records are signed
 - NS for delegations are not
 - Glue records are not

Signed Zone

- Records added to signed zone:
 - RRSIG
 - RRset signature
 - NSEC / NSEC3
 - Next SECure record
 - Hashed Authenticated Denial of Existence

Signed Zone – RRSIG

```
dnslab.org. 1800 IN SOA ns1.dnslab.org. aclegg.isc.org. (  
2009020201 1800 600 1209600 300 )
```

```
1800 RRSIG SOA 5 2 1800 20090212111244 (  
20090202111244 49416 dnslab.org.  
C1L1btWb8EHSu71m6ADqY/F87P59kbzMzgCa  
aD6zOKattCUiqK1ZL5BM/6n5Cn3KZdBWUSX3  
Y6cWfbgB5+96Qw== )
```

- RRSIG – RR Set Signature
 - Type Covered
 - Algorithm
 - Original Label Length
 - Covered Resource Record Original TTL

Signed Zone – RRSIG

```
dnslab.org. 1800 IN SOA ns1.dnslab.org. aclegg.isc.org. (  
2009020201 1800 600 1209600 300 )
```

```
1800 RRSIG SOA 5 2 1800 20090212111244 (  
20090202111244 49416 dnslab.org.  
C1L1btWb8EHSu71m6ADqY/F87P59kbzMzgCa  
aD6zOKattCUiqK1ZL5BM/6n5Cn3KZdBWUSX3  
Y6cWfbgB5+96Qw== )
```

- RRSIG – RR Set Signature
 - Expiration and Inception Dates
 - Key Tag
 - Signer's Name
 - Signature Data (Base 64)

Signed Zone – NSEC

```
dnslab.org. 1800 IN SOA ns1.dnslab.org. aclegg.isc.org. (  
2009020201 1800 600 1209600 300 )
```

```
300 NSEC authoritative.dnslab.org. (  
A NS SOA TXT AAAA RRSIG NSEC DNSKEY )
```

- NSEC – Next Secure Record
 - Next Secured Label
 - Types Included in this Label
- Allows enumeration of zone by walking

Signed Zone – NSEC

- If you don't like random people doing zone transfers from you, then you probably don't like NSEC
- After DNSSEC was completed as a standard, operators took notice and refused to deploy for this reason alone
- NSEC3 was provided as a means to provide hashed proof of non-existence

Signed Zone – NSEC3

- While NSEC3 does not allow enumeration of the zone, it does introduce performance penalties
- No-longer is a quick lookup able to determine the proof of non-existence
- Calculations are required on both the authoritative server and the validator to provide the correct NSEC3 record

Signed Zone – NSEC3

- To sign creating NSEC3 records, you MUST use a key generated with an algorithm supported by NSEC3.

```
dnssec-keygen -3
```

- Defaults to NSEC3RSASHA1

Signed Zone - NSEC3

```
dnssec-signzone [...] -3 salt -H  
number zonefile [ZSKfile]
```

- NSEC3 provides hashed proof of non-existence
 - "salt" is a hex string prepended to the labels being signed to defend from dictionary attacks

Signed Zone - NSEC3

```
dnssec-signzone [...] -3 salt -H  
number zonefile
```

- "number" is the number of permutations run to create the hashed label

- Default is 10
- Was (9.5 and prior) 100

Signed Zone – NSEC3

```
localhost.newzone.dnslab.org. 1800 IN A 127.0.0.1
```

```
TR5QI9BF3LB451UBIDAKC1G2IVSAO.newzone.dnslab.org. 300 IN NSEC3 1 (
  0 100 8D5A24 HRFSCG978JPEBE4VAG2QA8RABN6PIDI A RRSIG )
```

- NSEC3 - Hashed Authenticated Denial of Existence
 - Label is hash of the owner name
 - Hash Algorithm
 - Flags
 - Iterations

Signed Zone – NSEC3

```
localhost.newzone.dnslab.org. 1800 IN A 127.0.0.1
```

```
TR5QI9BF3LB451UBIDAKC1G2IVSAO.newzone.dnslab.org. 300 IN NSEC3 1 (
 0 100 8D5A24 HRFFSCG978JPEBE4VAG2QA8RABN6PIDI A RRSIG )
```

- NSEC3 - Hashed Authenticated Denial of Existence
 - Salt
 - Next Hashed Owner Name
 - Types Included in this Label

Un-Signed Zone Example

```
$TTL 30m
@           IN           SOA      ns1.dnslab.org. aclegg.isc.org. (
           2009020205 30m 10m 2w 5m )

           IN           NS       ns1.dnslab.org.
           IN           A        192.153.154.2
localhost  IN           A        127.0.0.1
```

- SOA, NS and 2 A records
- 193 characters

Signed Zone Example

```
; File written on Mon Feb  2 22:23:49 2009
; dnssec_signzone version 9.6.0-P1
sub.dnslab.org. 1800 IN SOA ns1.dnslab.org. (
    aclegg.isc.org.
    2009020205 1800 600 1209600 300)

1800 RRSIG SOA 7 3 1800 20090204210530 (
    20090202212349 22232 sub.dnslab.org.
    b3pK4d7fs4kKIoXarHXSv3MNmVXPsUEpo2Bt
    SEbdcMR9pM3sOSJQgzqGCJDQ2oV47eiyIH3I
    GziXNw7Q73MTMg== )

1800 NS ns1.dnslab.org.
1800 RRSIG NS 7 3 1800 20090204211752 (
    20090202212349 22232 sub.dnslab.org.
    uPFFtGu43ELb5srB5WifvdyxaVdazX8mfuf1
    6HJtQtU2AD/sAw+wF/UjbqjaP0GVcBVofJpT
    By7lSkOb0+R2SA== )

1800 A 192.153.154.2
1800 RRSIG A 7 3 1800 20090204210101 (
    [...]

300 RRSIG NSEC3 7 4 300 20090204212340 (
    20090202212349 22232 sub.dnslab.org.
    RpkubvTfWdZthjeSViRHvIWuWxQYvg61VOB/
    MQy0f9duhGgTImfhqh3ik+VhNkEAJUjuHUie
    /8+dbLkw0tkinQ== )
```

- SOA, NS and 2 A records..
- NSEC: 9 additional RRs
- NSEC3: 11 additional RRs
- 2950 Characters

Update named.conf

Replace

```
zone "zonename" {  
    file "dir/zonefile";  
};
```

With

```
zone "zonename" {  
    file "dir/zonefile.signed";  
};
```

Start serving signed zone

- Tell named to re-read the configuration

```
rndc reconfig  
rndc flush
```

- You are now serving DNSSEC signed zones

Notify parent of DNSSEC

- Your parent zone must now insert a "DS" RR to create a chain-of-trust
- Procedures will differ between organizations, but this must be done securely
 - will require use of `dsset-` and/or `keyset-` files

DNSSEC unaware parent

- Not all TLDs support DNSSEC
- Provide your `DNSKEY` to those that you wish to have validate your zone
 - This must be done securely, not just with "dig"

Trust Anchors

Trust Anchors

- To validate other zones, you must insert "trust anchors" for each zone apex below which you wish to validate
- The ultimate trust anchor would be a signed DNS root (".") with fully populated TLDs

Trust Anchors

- Now that the DNS root (".") is signed, there will only be one required trust anchor
- Even so, it is still possible and probably necessary to have additional trust anchors

Trust Anchors

- As of July 15th, 2010, the DNS root (".") is now signed
- You can find the root trust anchor @ <https://www.iana.org/dnssec/>
- Trust anchors must be obtained by trusted means
- NOTE: DNS is not one of those means

Trust Anchors

```
dig udp53.org DNSKEY
```

```
udp53.org. 14400 IN  DNSKEY 256 3 5 BE[...]/V1  
udp53.org. 14400 IN  DNSKEY 257 3 5 BE[...]1ylot7
```

- Doing the "dig" provides something that can be verified via other means (web, phone, printed media, etc.)

Trust Anchors

- `named.conf` will need to contain:

```
trusted-keys {  
    "udp53.org." 257 3 5 "BE[...]1y1ot7";  
    "isc.org." 257 3 5 "BEAAAAO[...]ZCqoif";  
};
```

- An entry for EVERY zone apex below which you wish to validate

Periodic Maintenance Issues

Periodic Zone Maintenance

- Signatures have lifespans
 - "Born-on" date – 1 hour prior to running `dnssec-signzone`
 - Expiration date – 30 days after running `dnssec-signzone`
- Expired signatures lead to zones that will not validate!

Periodic Zone Maintenance

- Any time you modify a zone – or at least every 30 days (minus TTL) you must re-run `dnssec-signzone`
- If you don't
 - 1) Zone data will be stale
 - 2) Zone data will be GONE

Periodic Key Maintenance

- Keys need to be rotated
 - No "expiration date"
- The longer a key is in public view, the more likely it is to be compromised
- Compromise (theft) of a key may lead to the need to "roll" a key over

ZSK Rollover

- ZSK Rollover (pre-publish key)
 1. Generate second ZSK
 2. Publish both (public) keys, but use only the old one for signing
 3. Wait at least propagation time (slave -> master) + TTL of the `DNSKEY` RR
 4. Use new key for zone signing; leave old one published
 5. Wait at least propagation time + maximum TTL of the old zone
 6. Remove old key

KSK rollover

- KSK Rollover (double signature)
 1. Generate new KSK
 2. Use both keys for key signing
 3. Send new `keyset` or `dsset` to the parent
 4. Wait until the `DS` is propagated + TTL of the old `DS RR`
 5. Remove the old key

Periodic Key Maintenance

- There is not "one answer" as to how often you should roll your keys.
- Some have documented:
 - KSK should be rolled once a year
 - ZSK should be rolled every 3 months

Domain Lookaside Validation

Trust Anchors

- Individual trust anchors do not scale well
- To help solve this problem, ISC created the DLV "Domain Lookaside Validation" RR and registry concept

DLV

- When validating, a resolver looks in the parent zone for a DS record for the zone being validated
- If it does not exist, a query for a DLV record in the DLV registry zone is made
- If successful, the DLV RR is used as the DS for the given zone

DLV Example

- `clegg.com` is signed
- The owner of `clegg.com` has registered with ISC's DLV Registry
- A DNSSEC query is made for the `A RR` for the label `www.clegg.com`
- No `DS` record is found in `.com` for the `clegg.com` zone

DLV Example

- A non-DLV enabled recursor with no explicit trust-anchor will not be able to do validation at this point
- A DLV enabled recursor will look for `clegg.com.dlv.isc.org. DLV RR`
- That `DLV RR` will then be used as the `DS` for the `clegg.com. zone`

Enabling DLV

- Use of DLV to validate is done on the recursive server
 - A trust anchor must be created for the DLV registry
 - `dnssec-lookaside` must be linked to the DLV trust anchor

Enabling DLV

- named.conf:

```
trusted-keys {
    dlv.isc.org. 257 3 5 "BEA[...]uDB";
};

options {
    dnssec-lookaside "."
    trust-anchor dlv.isc.org.;
};
```

Registering with DLV

- Contact the DLV registrar for instructions on how to prove ownership of zone and validity of records to be inserted into their registry.
- Insertion of your DLV RR into the DLV registry must be done in a trusted manner.

ISC's DLV registry

<http://dlv.isc.org>

In Conclusion

- DNSSEC is real and in production
- You can sign your zone now and publish in ISC DLV until your parent is signed.
- Use BIND 9.7.1-P2 or later for any validating recursive servers.

Any questions?