Deploying DNSSEC Using BIND 9.7

Internet Systems Consortium
About the Presenter

• Alan Clegg

aclegg@isc.org

+1-919-355-8851

@Knobee
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.
is the new "COBOL"

Deploy DNSSEC now... or something bad might happen...
is the new "COBOL"

Define a security standard for DNS that can be deployed, and operators will.

Deploy DNSSEC now... or something bad might happen...
Understanding DNSSEC
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

• 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 resolvers 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

• DNSSEC enabled authoritative servers provide digital signatures across RRsets in addition to "standard" DNS responses

• DNSSEC validating resolvers provide authenticated responses with proven integrity
Introduction

• Clients using validating resolvers get "guaranteed good" results

• Data that does not validate provides a "SERVFAIL" response from the upstream resolver
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?

• Now that the root ("." ) is signed, that's easy, right?
Trust Validation

• DNSSEC is based on chains of trust

• At the top of chains are "trust-anchors"
  – One (signed) root, one trust-anchor
  – Until all TLDs are signed, it's not so easy
  – Trust anchors must be gathered and added to DNS configuration through leaps of faith
Trust Validation

• In BIND, trust anchors are added in "trusted-keys" statements

```plaintext
trusted-keys {
  . 257 3 8 "AwEAA[..]ihz0=";
};
```

• This creates an anchor based at the DNS root from which a chain is created
Chain of Trust

• Once a "trust anchor" is inserted, how does it actually create trust that leads down the DNS tree?

• Trust anchors consist of bits capable of validating the key used to sign the key that signs data in a given zone.
Chain of Trust

• First, we must realize that there are TWO keys inserted into each zone

  – Zone Signing Key (ZSK)
    • Used to sign the resource records in the zone being secured
  – Key Signing Key (KSK)
    • Used to sign the Zone Signing Key
Chain of Trust

• Delegation of signed zones include a new Resource Record type

  – Delegation Signer – \textit{DS}

  – Hash of the public portion of the child's Key Signing Key
Chain of Trust

• If the $\text{DS}$ record in the parent is signed using the parent's zone signing key, we know that the $\text{DS}$ record is valid.

• If the hash of the child's Key Signing Key record matches the $\text{DS}$ 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 were created using Sandia National Laboratories "DNSViz"

http://dnsviz.net/
Trusting isc.org

. (root)

- KSK 19036
- ZSK 41248
  • Signed w/19036
- .org DS records
  • signed w/ 41248

(DNSKEY 008+19036

DNSKEY 008+41248

DS (1,2) 007+21366

(2010-07-23 15:26:05 UTC)
Trusting isc.org

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

isc.org

- KSK 12892
  - Hashed into DS

- ZSK 18516
  - Signed w/ 12892

- SOA, AAAA, A
  - Signed w/ 18516
With a trust anchor for root we can trust anything below it that is signed.

And that has DS records in place.
DNSSEC Deployment
BIND 9.7
Recursive Server
Trust Validation

• In BIND, trust anchors are added in "trusted-keys" statements

```plaintext
class trust-anchor {
  257 3 8 "AwEAA[..]ihz0=";
}
```

• But, what happens when a "hard-configured" key changes?
RFC-5011 ready anchors

• Be ready for KSK roll-over:

```plaintext
managed-keys {
  "." initial-key 257 3 8
  "AwEAA[..]k1ihz0=";
};
```

• Defines the initial key used as KSK for the given zone
RFC-5011 ready anchors

• A file is created that tracks key changes

  managed-keys.bind
  managed-keys.bind.jnl

• This file will contain the currently active key, even if the configured key has rolled
RFC-5011 ready anchors

• Newly added "rndc secroots"

  - Creates a file "named.secroots" containing a list of the current managed keys that are in use:

    10-Sep-2010 12:56:08.950

    Start view _default

    ./RSASHA256/19036 ; managed
dlv.isc.org/RSASHA1/19297 ; managed
RFC-5011 ready anchors

• One problem with managed-keys:
  
  – If a key has rolled without being noticed, validation will fail
  
  – This can happen if a validating server is off-line during a key roll-over, etc.
Authoritative Server
DNSSEC Deployment

• Generate required keys
  - `dnssec-keygen`

• Insert them into the zone
  - manual (or dynamic)

• Sign zone data
  - `dnssec-signzone` (or dynamic)

• Perform scheduled zone maintenance
  - manual (or dynamic)
DNSSEC Deployment

• `dnssec-keygen`
  – Used to create the required keys

• Key Signing Key
• Zone Signing Key
DNSSEC Deployment

- `dnssec-keygen`

  - Defaults algorithm to `RSASHA1`

  - Provides defaults for key size if default algorithm is used:
    - `KSK` – 2048 bits
    - `ZSK` – 1024 bits
DNSSEC Deployment

- `dnssec-keygen <zonename>`
- `dnssec-keygen -f KSK <zonename>`

- Produces 2 files per key

  K<zonename>+XXX+YYYY.key
  K<zonename>+XXX+YYYY.private
DNSSEC Deployment

- `dnssec-keygen`

  - Once keys are created, include their public portions (`.key`) into the zone file using standard procedures

  - Keep the `.private` portions secure
DNSSEC Deployment

- `dnssec-signzone`
  - Signs the zone data
  - Creates RRSIG resource records for each authoritative RRset in the zone
  - Transforms zone into "machine generated" file with a `.signed` extension
DNSSEC Deployment

- `dnssec-signzone`

  - BIND 9.7 introduced a new feature.

- Smart Signing
  - Looks in key repository (directory) for keys
  - Keys are included in zone automatically
  - If key files contain timing meta-data, that timing data is used
DNSSEC Deployment

- **named**
  - New dynamic zone configuration
    - `update-policy local;`
      - Automatically creates "local-only" TSIG key
    - Allows BIND to update without complex configuration
DNSSEC Deployment

• named

– New zone options for dynamic zones
  • auto-dnssec off;
    – Default
  • auto-dnssec allow;
    – Enables auto-inclusion of keys from repository
    – Enables "rndc sign"
  • auto-dnssec maintain;
    – Update DNSSEC based on key meta-data
DNSSEC Deployment

• nsupdate

  – New option `-1 (ell)`

• Use the named created "local key"

• Set the server address to localhost
DNSSEC Deployment

- **rndc**
  - New option `sign`

- Takes a dynamic zone, searches for keys in the key repository and signs the zone as needed.
zone test.com {
    type master;
    key-directory "keys";
    update-policy local;
    auto-dnssec maintain;
    file "dynamic/test.com.zone";
};
Making it work...

dnssec-keygen -K /etc/namedb/keys \  
test.com

dnssec-keygen -f KSK -K /etc/namedb/keys \  
test.com

rndc sign test.com

Zone is now signed and published

Zone will be automatically re-signed as needed
DNSSEC "just works"

• Adding or removing zone contents is now as simple as:

```
nsupdate -l
> update add <RRset>
> send
```

• **RRset** is added and signed data updated automatically
Timing Meta-Data

- `dnssec-keygen` creates meta-data in the key file:
  - `-P` – Publication Date (default: now)
  - `-A` – Activation Date (now)
  - `-R` – Revocation Date (none)
  - `-I` – Retirement Date (none)
  - `-D` – Deletion Date (none)
Timing Meta-Data

• These dates are used by named to maintain the zone signatures

• Date formats:

  none (literal)
  YYYYMMDD
  YYYYMMDDHHMMSS
  now+<offset>

  y, mo, w, d, h, mi
Timing Meta-Data

• To pre-publish a KSK without signing:

```bash
dnssec-keygen -K keydir \ 
-f ksk -A none test.com

[...]Ktest.com.+005+11353

rndc sign test.com
```
Timing Meta-Data

- Once you are ready to sign the zone with the given key:

  ```bash
  dnssec-settime -K keydir \
  -A now Ktest.com.+005+11353
  rndc sign test.com
  ```
Timing Meta-Data

- To no-longer sign with the key, but leave it in the zone:

```bash
dnssec-settime -K keydir \  
   -I now Ktest.com.+005+11353 
rndc sign test.com
```
Timing Meta-Data

• And finally, remove the key from the zone:

```
dnssec-settime -K keydir \
-D now Ktest.com.+005+11353
rndc sign test.com
```
Automation Warning!

Be aware that this automation does NOT deal with DS records in the parent or DLV records in a registry.
DNSSEC Deployment

• BIND 9.7.2
  (currently [9/1/2010] release candidate)

allow-new-zones option
  • boolean allowing creation of zones "on the fly"

rndc addzone / rndc delzone
  • add and remove zones without manually editing named.conf
Create & Sign a zone

#!/bin/bash
cd /etc/namedb
cp template master/\${1}

rndc addzone \${1} \{ type master\;
    file "master/\${1}\"; \n    update-policy local; \n    auto-dnssec maintain; \n}\;

dnssec-keygen -f KSK -K /etc/namedb/keys \$1
dnssec-dsfromkey -2 /etc/namedb/keys/K$\{1\}.\.*.key > ds/$\{1\}
dnssec-keygen -K /etc/namedb/keys \$1
rndc sign $\{1\}
Create & Sign (NSEC3)

#!/bin/bash
SALT=`printf %04x%04x $RANDOM $RANDOM`
cd /etc/namedb
cp template master/${1}

rndc addzone ${1} { [..] };

nsupdate -l << //EOF
update add ${1} 30 IN NSEC3PARAM 1 0 10 $SALT
//EOF
dnssec-keygen -3 -f KSK -K /etc/namedb/keys $1
dnssec-dsfromkey -2 /etc/namedb/keys/K${1}.*.key > ds/${1}
dnssec-keygen -3 -K /etc/namedb/keys $1
rndc sign ${1}
Questions?
Comments?

Ready to deploy?