DNSSEC

Background

- The original DNS protocol wasn't designed with security in mind
- It has very few built-in security mechanism
- As the Internet grew wilder & wollier, IETF realized this would be a problem
  - For example DNS spoofing was to easy
- DNSSEC and TSIG were develop to help address this problem

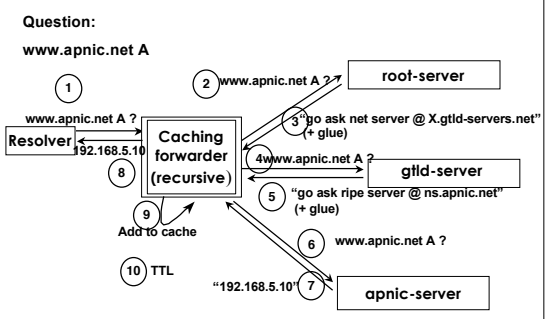
Why DNSSEC?

- DNS is not secure
  - Applications depend on DNS
    - Known vulnerabilities
- DNSSEC protects against data spoofing and corruption

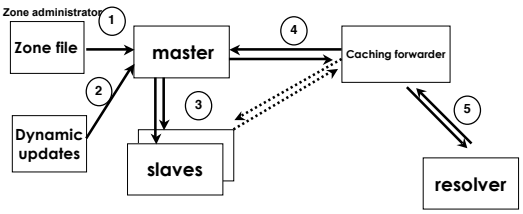
Overview

- Introduction
- DNSSEC mechanisms
  - To authenticate servers (TSIG)
  - To establish authenticity and integrity of data
    - Quick overview
    - New RRs
    - Using public key cryptography to sign a single zone
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers
- Conclusions

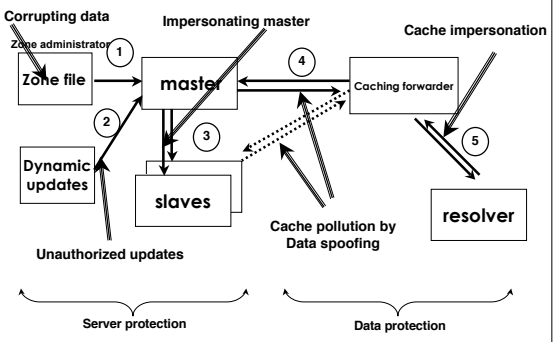
Reminder: DNS Resolving



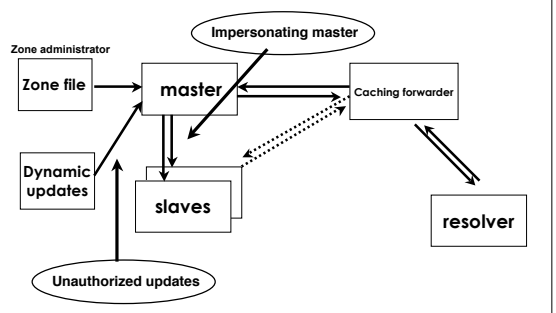
DNS: Data Flow



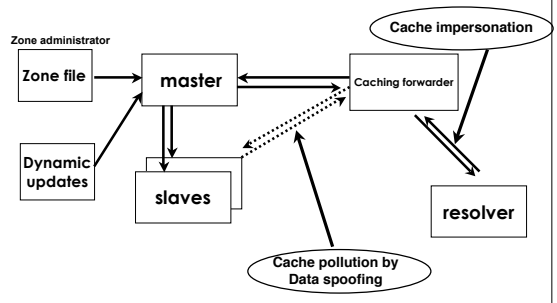
DNS Vulnerabilities



TSIG Protected Vulnerabilities



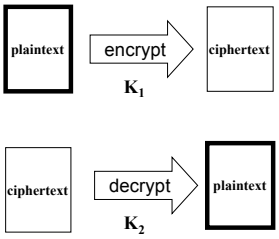
Vulnerabilities protected by DNSKEY / RRSIG / NSEC



Difference Between TSIG and DNSSEC

- TSIG secures transaction
  - Making sure DNS messages come from the right place and aren't modified in transit
- DNSSEC secures (signs) zone data
  - Making sure resource records are those signed by the administrator of the zone
- Only endpoints that share a key can use TSIG to verify DNS messages
- Any endpoints that support DNSSEC can use it to verify signed zone data

Public Key Cryptography Illustrated



The Key Pair: Public and Private

- In practice
  - One key of the pair is kept private
  - The other key is made public, by uploading it to a key server, publishing it via a directory, or having a certification authority sign it in to a certificate

Public Key Cryptography

- The magic behind public key cryptography is the *key pair*
- A key pair is generated from a mathematical formula such that data encrypted with one key can only be decrypted by the other
- In other words, if
  - *E()* is the encryption function
  - *D()* is the decryption function
  - *p* is plaintext data
  - *c* is ciphertext (encrypted) data
- And the key pair is k_1 and k_2

  $c = E_k (p)$, $p = D_k$

The DNSKEY Record

- In DNSSEC, each zone has a key pair
- The private key
  - Is stored somewhere secure
  - Is used to sign zone data
- The public key
  - Is stored in the zone data, as a DNSKEY record
  - Is used to verify zone data
- DNSKEY was known before as the KEY RR

The DNSKEY Record (cont.)

- **Example:**

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MU
                G2DeIQ3Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                kfzj31GajIQKY+5CptLr3buXA10hWqTkF7H6
                RfoRqXQeogmMHfpftf6zMv1LyBUgia7za6ZEzOJB
                OztyvhjL742iU/TpPSEDhm2SNKLijfUppn1U
                aNvv4w==  )
```

- The RDATA fields are:
  - Flags (256, indicating that this key can be used for authentication and confidentiality, and that it's a zone key)
  - Protocol (3, indicating that this is a DNSSEC key)
  - Algorithm (5, indicating that this key is used with the RSA/SHA1 public key algorithm)
  - The key itself, in base 64

The RRSIG Record

- A zone's private key is used to sign the data in the zone
- The signatures are added to the zone in the form of RRSIG records
  - **Example:**

```
foo.example  SOA  ns1.foo.example hostmaster.foo.example. (
                        2005072200 ; serial
                3600    ; refresh (1 hour)
                900     ; retry (15 minutes)
                2592000 ; expire (4 weeks 2 days)
                3600    ; minimum (1 hour)
                        )
RRSIG     SOA 1 2 86400 20050820205058 (
        20050720205058 25993 foo.example.
        D+JBeIZvzzTc//m/Tq0uwccbfjCHd7siiIYJ
        fpqy45XU5mMRumkvMH7NyShjjtg+Qijr2uvh
        J/VpgiVVS38TEg== )
```

The RRSIG Record (cont.)

- The RDATA fields are:
  - The type covered (SOA, indicating that this RRSIG record signs the domain name's SOA record)
  - The protocol (1, indicating that this RRSIG record was produced using RSA)
  - The number of labels in the domain name signed (2)
  - The original TTL on the record (86400)
  - The RRSIG record's expiration (20050820205058, indicating that this RRSIG record expires on August 20, 2005, just before 9 pm GMT)

The RRSIG Record (cont.)

- The RDATA fields are:
  - The RRSIG record's inception (20050720205058, indicating that this RRSIG record was added on July 20, 2005, just before 9 pm GMT)
  - The key ID (25993), used to identify the key to use to verify the RRSIG record
  - The signer's domain name (*foo.example*), used to look up the key to use to verify the RRSIG record
  - The signature itself, in base 64

The NSEC Record

- RRSIG records are fine for authenticating records
- But how about negative responses, like NXDOMAIN or NODATA?
  - These don't contain records to sign
- To authenticate those, DNSSEC adds a new record type, NSEC (the RR formerly known as NXT)

The NSEC Record

- The NSEC record for a given domain name tells you
  - Which record types exist for that domain name
  - Which gaps ("empty space") exists between domain names in a zone
  - The next domain name in the zone

Sorting Zones

- The idea of a "next secure" record suggests that, in DNSSEC, zones have an order
- To sort the domain names in a zone into the proper order
  - Write all owner names as fully qualified
  - Shift all owner names to lowercase
  - Sort lexicographically from highest-level (rightmost) label in owner name to lowest-level (leftmost) label
  - Non-existent labels come before X, so *foo.example* precedes
    - *X.foo.example*

Sorting a Zone

- **So…this zone**

```
foo.example. 86400 IN SOA ns1 hostmaster (
        2001112000 1h 15m 30d 1h    )

        86400   IN      NS      ns1
        86400   IN      MX      10 mail
        3600    IN      A       192.168.0.1
www             86400   IN      CNAME   @
ns1             86400   IN      A       192.168.0.3
ns2             86400   IN      A       10.0.0.1
sub             86400   IN      NS      ns1.sub
sub             86400   IN      NS      ns2.sub
ns1.sub         3600    IN      A       172.16.0.1
ns2.sub         3600    IN      A       172.16.0.2
mail            86400   IN      A       192.168.0.2
```

Sorting a Zone (cont.)

- **Would sort to this**

```
foo.example. 86400 IN SOA ns1.foo.example. (
hostmaster.foo.example. 2001112000 1h 15m 30d 1h )

foo.example.          86400 IN NS          ns1.foo.example.
foo.example.          86400 IN MX          10
foo.example.          3600  IN A           192.168.0.1
mail.foo.example.     86400 IN A           192.168.0.2
ns1.foo.example.      86400 IN A           192.168.0.3
ns2.foo.example.      86400 IN A           10.0.0.1
sub.foo.example.      86400 IN NS   ns1.sub.foo.example.
sub.foo.example.      86400 IN NS   ns2.sub.foo.example.
www.foo.example.      86400 IN CNAME       foo.example.
ns1.sub.foo.example. 3600 IN  A          172.16.0.1
ns2.sub.foo.example. 3600 IN  A          172.16.0.2
```

A Sorted Zone with NSEC Records

```
    foo.example. 86400 IN SOA     ns1.foo.example. hostmaster.foo.example.
         (                        2001112000 ; serial
                                  3600       ; refresh (1 hour)
                                  900        ; retry (15 minutes)
                                  2592000    ; expire (4 weeks 2 days)
                                  3600       ; minimum (1 hour)
                                                                    )
                 86400    NS       ns1.foo.example.
         86400    NS       ns2.foo.example.
         3600     A        192.168.0.1
         86400    MX       10 mail.foo.example.
         86400    NSEC     mail.foo.example. A NS SOA MX NSEC
mail.foo.example.86400    IN A     192.168.0.2
         86400    NSEC     ns1.foo.example. A NSEC
ns1.foo.example. 86400    IN A     192.168.0.3
         86400    NSEC     ns2.foo.example. A NSEC
ns2.foo.example. 86400    IN A     10.0.0.1
         86400    NSEC     sub.foo.example. A NSEC
ns1.sub.foo.example.      3600     IN A 172.16.0.1
ns2.sub.foo.example.      3600     IN A     172.16.0.2
sub.foo.example. 86400    IN NS    ns1.sub.foo.example.
         86400    IN NS    ns2.sub.foo.example.
         86400    KEY      49408 3 3
         86400    NSEC     www.foo.example. NS KEY NSEC
www.foo.example. 86400    IN CNAME foo.example
         86400    NSEC     foo.example. CNAME NSEC
```

The Chain of Trust

- What prevents a hacker from breaking in to the primary master name server and changing the zone data, including the zone's DNSKEY record?
  - The DNSKEY record is signed by the zone's private key
  - This DNSKEY can be securely identified with the according DS RR from the parents' zone
  - The DS RR in the parent zone is signed by the parent zone's private key. (This builds the "chain of trust")
- Security aware resolver have a public key (or a hash of a public key) of a trusted-"root" ("." or DS RR)
- DS RR can be used to start an "Island of security" down from ".

The Chain of Trust

- This established a *chain of trust* from any signed zone to the root
  - The zone's DNSKEY record is self-signed by its private zone- signing key
  - The zone's DS record (in the parent zone) is signed by the parent-zones' private key
  - The parent zone's DS record is used to identify the zone's public key, and so on
  - The root zone's DNSKEY record is widely known

    Or
  - The start of a *chain of trust* can be verified by a locally configured public key (or hash of a Public Key)

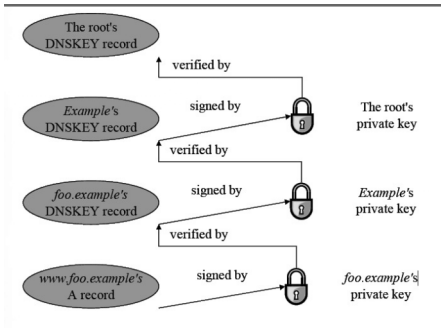
The Chain of Trust (cont.)

- Example:

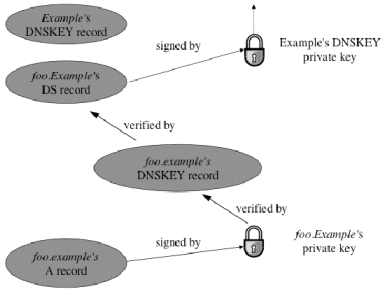
```
$TTL 3600             ;
1 hour foo.example IN DNSKEY  256 3 1 (
  AQPJovN2i//gtI17J28mIHvwy0QZaoA60IUIE98Hy6NT
  sJLHD8kVAbydZTVlej5FJpjpnK5Im+hRnnGVq3GIZYmH )
  ; key id = 25993

  RRSIG DNSKEY 1 2 3600 20011220212022 (
  20011120212022 54056 example.
  PE84yB2wUU4Ai49b3+8pOlesZWIs2nue8bKxf+Kn3tW7
  wCrvaRDsGCvKePWwP8V3A5tkez3wtQif7uIigvlq+w== )
```

The Chain of Trust Illustrated (Detail view)



The Chain of Trust Illustrated (Detail view)



Implementation of the Chain of Trust

- If the chain of trust is broken
- However, administrators of individual name servers can work around this by adding *trusted-keys* statements to their name server's *named.conf* files
- **Example:**

```
trusted-keys {
   "foo.example." 256     3 1
   "AQPJovN2i//gtI17J28mIHvwy0QZaoA60IUIE98Hy6NTsJLHD8
    VAbydZTVlej5FJpjpnK5Im+hRnnGVq3GIZYmH";
    };
```

The DS (Delegation Signer) Record

- The DS RR is used in the DNSKEY authentication process
  - Answer to the Question: is the zone's public key (DNSKEY) valid?
- The DS Key is stored in the parent zone of the DNSKEY's zone
- DS RR simplifies DNS zone management and zone signing

The DS (Delegation Signer) Record

- **Example**
  - DNSKEY in (child-)zone

```
sub.example.com. 86400 IN DNSKEY 256 3 5 (
                       AQOeiiR0GOMYkDshWoSKz9Xz
                       fwJr1AYtsmx3TGkJaNXVbfi/
                       2pHm822aJ5iI9BMzNXxeYCmZ
                       DRD99WYwYqUSdjMmmAphXdvx
                       egXd/M5+X7OrzKBaMbCVdFLU
                       Uh6DhweJBjEVv5f2wwjM9Xzc
                       nOf+EPbtG9DMBmADjFDc2w/r
                                  ljwvFw==
                       ) ;     key id = 60485
```

  - DS in Parent-zone

```
sub.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588
                       179A53B0A 98631FAD1A292118 )
```

The DS (Delegation Signer) Record

- **The RDATA fields are:**
  - Key Tag field (to help identify the key in the child zone)
  - Algorithm field (of the key in the child zone)
  - Digest Type field (Algorithm of the Digest/Hash value)
  - The Digest/Hash of the Key in the child zone, in base64

Steps to Signing a Zone

1. Generate a key pair for the zone
2. Add the DNSKEY record to the zone
3. Sign the zone
4. (Optional) Look how much bigger it got
5. Update the *zone* statement
6. Send the dsset to your parent zone's administrator
7. Wait for DS RR in parent zone
8. Test

1. Generate a Key Pair for the Zone

- **Example:**

```
# dnssec-keygen -a RSA -b 512 -n ZONE foo.example.
Kfoo.example.+001+25993   // 4 files will be generated
* Keyset, dsset, Public & Private Keys

# more Kfoo.example.+001+25993.key
foo.example. IN KEY 256 3 1
AQPJovN2i//gtI17J28mIHvwy0QZaoA60IUIE98Hy6NTsJLHD8kVAbyd
ZTVlej5FJpjpnK5Im+hRnnGVq3GIZYmH

# more Kfoo.example.+001+25993.private
Private-key-format: v1.2
Algorithm: 1 (RSA)
Modulus:
yaLzdov4LSNeydvJiB78MtEGWqAOtCFCBPfB8ujU7CSxw/J
FQG8nWU1ZXo+RSaY6JvZ5xlatxGWJhw==
PublicExponent: Aw== PrivateExponent:
hmyiTwf6yMl/MT0xBWn9dzYEPGq0eBYsA0/WofCN8ndIEJZnhdBNqdbJ53K51qN
odHoWw==
Prime1: 5OjA15uBHdv1wDcQlrOWRviHM6OkqTwOa
4XrawxdvPFX8xsSKzD5S9sUZ9T8NlojVvY+YdnV5XU= Exponent1:
+RlOzvUx4s= Prime2: rime1: 5OjA15uBHdv1wDcQlrOWRviHM6OkqTwOa
+RlOzvUx4s=
Prime2: Prime1: 5OjA15uBHdv1wDcQlrOWRviHM6OkqTwOa+RlOzvUx4s= Prime2:
Prime1:5OjA15uBHdv1wDcQlrOWRviHM6Okqrime1:
```

2. Add the DNSKEY Record to the Zone

- **Example:**

```
# more db.foo.example
$TTL 1d
foo.example. 86400 IN SOA ns1 hostmaster (
2001112000 1h 15m 30d 1h)
          86400 IN NS ns1
          86400 IN NS ns2
          86400 IN MX 10 mail
          3600 IN A 192.168.0.1
www        86400 IN CNAME @
ns1        86400 IN A192.168.0.3
ns2        86400 IN A 10.0.0.1
sub        86400 IN NS ns1.sub
sub        86400 IN NS ns2.sub
ns1.sub    3600 IN A 172.16.0.1
ns2.sub    3600 IN A 172.16.0.2
mail       86400 IN A 192.168.0.2

$INCLUDE Kfoo.example.+001+25993.key
```

3. Sign the Zone

- **Example:**

```
# dnssec-signzone -o foo.example db.foo.example
db.foo.example.signed

# more db.foo.example.signed
; File written on Tue Nov 20 16:55:15 2001
; dnssec_signzone version 9.1.3

foo.example.            86400 IN SOA ns1.foo.example. hostmaster.foo.example.
                        2001112000          ; serial
                        3600                ; refresh (1 hour)
                        900                 ; retry (15 minutes)
                        2592000             ; expire (4 weeks 2 days)
                        3600                ; minimum (1 hour)
                        )
                86400   RRSIG  SOA 1 2 86400 20011220235515 (
                20011120235515 25993 foo.example.
                QiBBXkDhaR1XR+bJo1W+2XhehvA8go45WRublbD3Ghas38tHeaiT1LfN3
                7ZlgnHZtlYobLrE/y1cikXuo58pnA== )
```

4. Look How Much Bigger It Got

- **Example:**

```
# wc db.foo.example*

15 78   587 db.foo.example
112     399  4246 db.foo.example.signed
127     477  4833 total
```

5. Update the Zone Statement

- Example:

```
# cat /etc/named.conf
 zone "foo.example" {
     type master;
     file "db.foo.example.signed";
      };
# rndc reload foo.example
```

6. Send the dsset to Your Parent Zone's Administrator

```
· # mail hostmaster@example
 Subject: Please sign the foo.example dsset

 Dude,

 Would you please sign the foo.example dsset for me
  and put the DS RR in your zone.

 You rock,

 hostmaster@foo.example
```

dsset-foo.example.

7. Wait for DS RR in parent zone

- Example:

foo.example. 86400 IN DS 25993 3 1 (2BB183AF5F22588
179A53B0A98631FAD1A292118)

8. Test

- **Example:**

```
dig soa foo.example +dnssec
 ;; QUESTION SECTION:
;foo.example.      IN        SOA

;; ANSWER SECTION:
foo.example.       86400     IN        SOA       ns1.foo.example.
 hostmaster.foo.example. 2001112000 3600 900 2592000 3600 foo.example. 86400
IN      RRSIG     SOA 1 2 86400 20011221203100 20011121203100 25993 foo.example.
 jaxaqjn19uuFx4NxbW7ftUJvENz1XezONRSuaHpw/awrw3k8uV2aZXaD
 tNz/OWlpvu5p+1Dn3/2AI83ppNLbfw==
;; AUTHORITY SECTION:
foo.example.       86400     IN        NS        ns2.foo.example
foo.example.       86400     IN        NS        ns1.foo.example.
foo.example.       86400     IN        RRSIG     NS 1 2 86400 20011221203100
                             20011121203100 25993 foo.example
        DOIahoJejw99MVSKhJDLaIeARJ8NRQ9QXQxMzV8Nq5cr79moWPPvIV+Q
        pdF/jK3KhvTBQ4cnmjq0axL/c1V2cA==
;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1
```

Resigning a Zone

- Just run *dnssec-signzone* on it again
- Example:

```
# dnssec-signzone -o foo.example db.foo.example.signed \
 Kfoo.example.+001+25993

# mv db.foo.example.signed.signed db.foo.example.signed
# reload foo.example
```

Questions?