DNS/DNSSEC

23 January 2015

APNIC

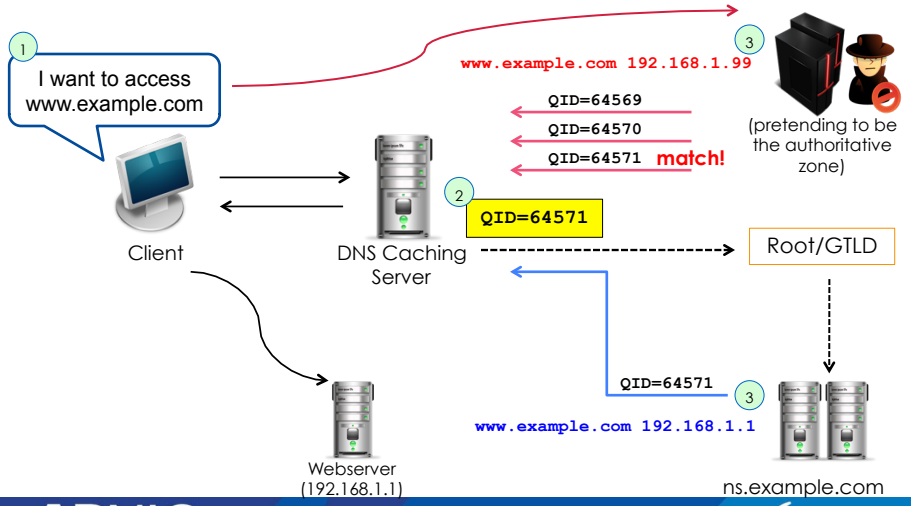
DNS Security - Background

- The original DNS protocol wasn't designed with security in mind
  - It has very few built-in security mechanism
- As the Internet grew wilder, IETF realized this would be a problem
  - For example DNS spoofing was to easy
- DNSSEC and TSIG were develop to help address this problem
- Some security problems:
  - Using reverse DNS to impersonate hosts
  - Software bugs (buffer overflows, bad pointer handling)
  - Bad crypto (predictable sequences, forgeable signatures)
  - Cache poisoning (putting inappropriate data into the cache)

https://wiki.tools.isoc.org/DNSSEC_History_Project

APNIC

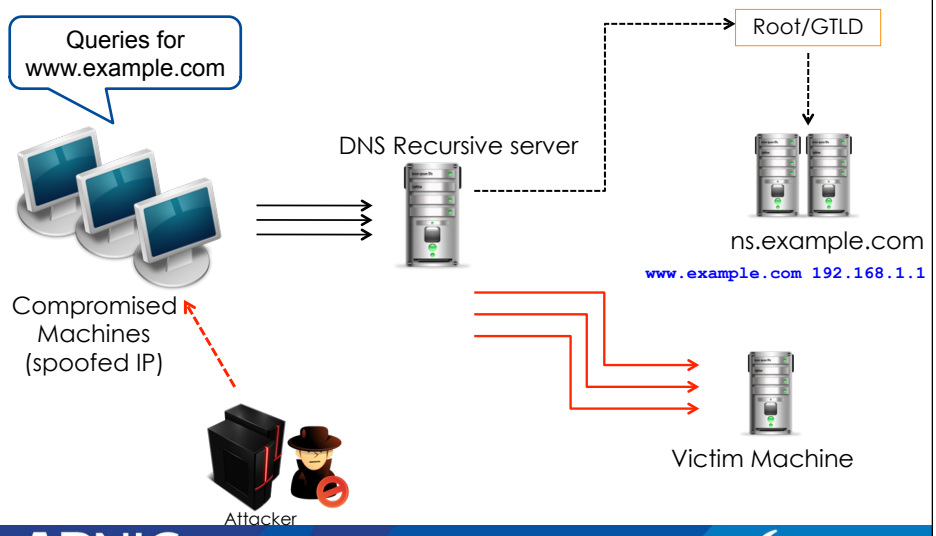
DNS Cache Poisoning



DNS Amplification

- A type of reflection attack combined with amplification
  - Source of attack is reflected off another machine
  - Traffic received is bigger (amplified) than the traffic sent by the attacker
- UDP packet's source address is spoofed

DNS Amplification



APNIC

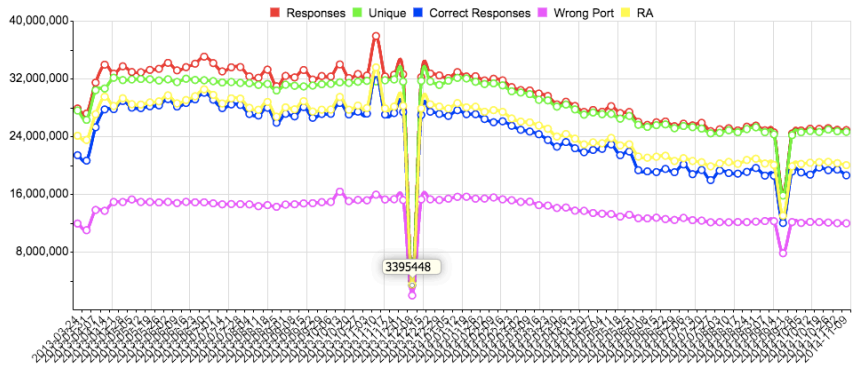
Open Resolvers

- DNS servers that answer recursive queries from any host on the Internet
- http://openresolverproject.org/
- Check if you're running open resolvers
  - http://dns.measurement-factory.com/cgi-bin/openresolvercheck.pl
- More statistics at
  - http://dns.measurement-factory.com/surveys/openresolvers/ASN-reports/latest.html

APNIC

Open Resolvers

As of 19 Nov 2014:
22,796,823 servers responded to udp/53 probe
17,939,673 returned OK



Reference: http://openresolverproject.org/

Response Rate Limiting (RRL)

- Protects against DNS amplification attack
- Implemented in CZ-NIC Knot (v1.2-RC3), NLNetLabs NSD (v3.2.15), and ISC BIND 9 (v9.9.4) release

```
rate-limit {
            responses-per-second 5;
            log-only yes;
    };
```

- If using older versions, a patch is available from
  - http://ss.vix.su/~vjs/rrlrpz.html
  - `patch -p0 -l`

DNS Changer

- “Criminals have learned that if they can control a user’s DNS servers, they can control what sites the user connects to the Internet.”
- How: infect computers with a malicious software (malware)
- This malware changes the user’s DNS settings with that of the attacker’s DNS servers
- Points the DNS configuration to DNS resolvers in specific address blocks and use it for their criminal enterprise
- For more: see the NANOG presentation by Merike

Top DNS Changer Infections

- By country (as of 11 June 2012):
  - USA - 69517
  - IT – 26494
  - IN – 21302
  - GB – 19589
  - DE – 18427
- By ASNs
  - AS9829 (India) – 15568
  - AS3269 – 13406
  - AS7922 – 11964
  - AS3320 – 9250
  - AS7132 – 6743
- More info at http://dcwg.org/

Securing the Nameserver

- Run the most recent version of the DNS software
  - Apply the latest patches
- Hide version
- Restrict queries
  - `Allow-query { acl_match_list; };`
- Prevent unauthorized zone transfers
  - `Allow-transfer { acl_match_list; };`
- Run BIND with the least privilege (use `chroot`)
- Randomize source ports
  - don't use `query-source` option
- Secure the box
- Use TSIG and DNSSEC

Sender Policy Framework (SPF)

- Using DNS for email validation
- Checks the sender IP address
- Defined in RFC 4408 with updates in RFC 6652

```
apnic.net.          3600    IN      TXT     "v=spf1 mx
a:clove.apnic.net a:asmtp.apnic.net ip4:203.119.93.0/24
ip4:203.119.101.0/24 ip4:203.89.255.141/32
ip4:203.190.232.30/32 ip4:122.248.232.184/32
include:_spf.google.com -all"
```

DANE

- DNS-Based Authentication of Named Entities
- RFC 6698 (proposed standard)
- “secure method to associate the certificate that is obtained from the TLS server with a domain name using DNS”
- Adds a TLSA resource record

DNS RPZ

- Resource Policy Zone
- Developed for ISC Bind. Built in from version 9.8
- Turns a recursive DNS server into a “DNS firewall”
- “reputation-based” zones
- Like creating a reputation server for recursive DNS servers
  - Function is similar to DNSBL for email SMTP servers
- Blocks DNS resolution to malicious hosts

Questions

APNIC

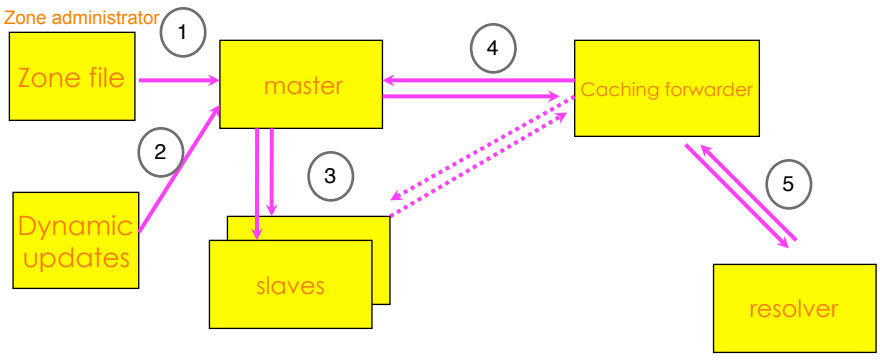
Transaction Signature (TSIG)

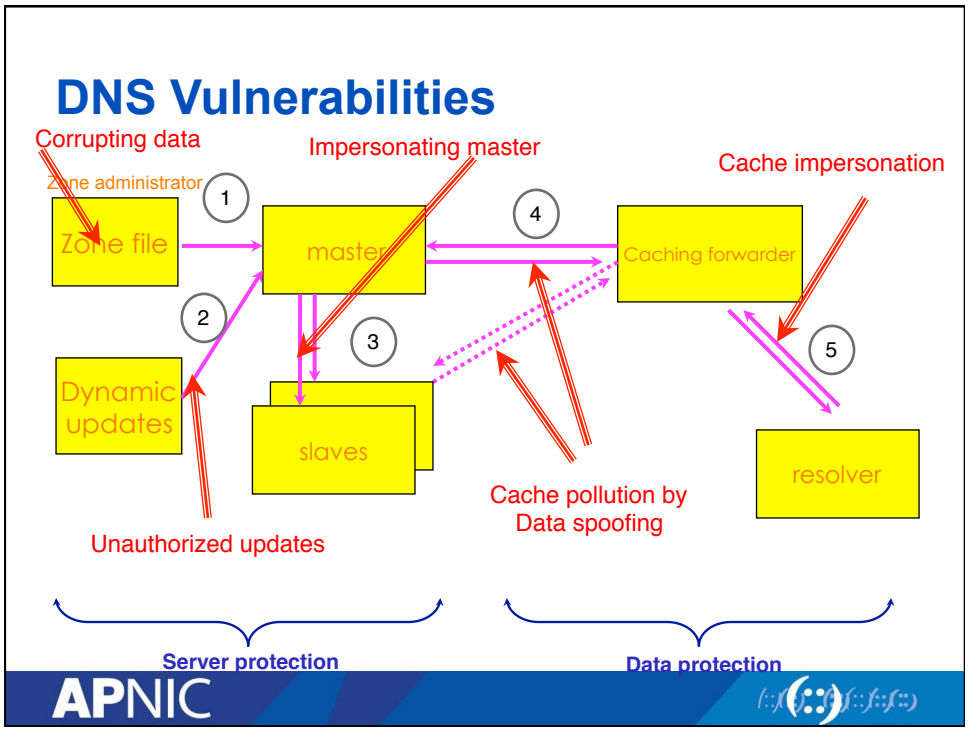
APNIC

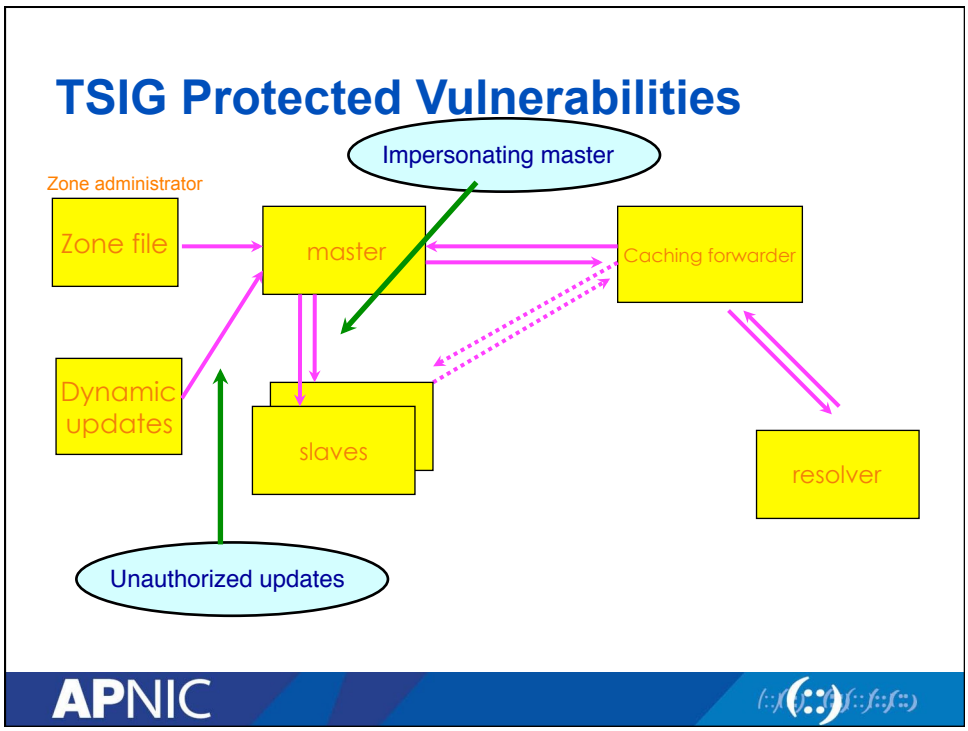
DNS Protocol Vulnerability

- DNS data can be spoofed and corrupted between master server and resolver or forwarder
- The DNS protocol does not allow you to check the validity of DNS data
  - Exploited by bugs in resolver implementation (predictable transaction ID)
  - Polluted caching forwarders can cause harm for quite some time (TTL)
  - Corrupted DNS data might end up in caches and stay there for a long time
- How does a slave (secondary) know it is talking to the proper master (primary)?

DNS: Data Flow




DNS Vulnerabilities
Corrupting data
Impersonating master
Cache impersonation
Zone administrator
Zone file
master
Caching forwarder
resolver
Dynamic updates
slaves
1
2
3
4
5
Unauthorized updates
Cache pollution by Data spoofing
Server protection
Data protection
APNIC


TSIG Protected Vulnerabilities
Impersonating master
Zone administrator
Zone file
master
Caching forwarder
resolver
Dynamic updates
slaves
Unauthorized updates
APNIC

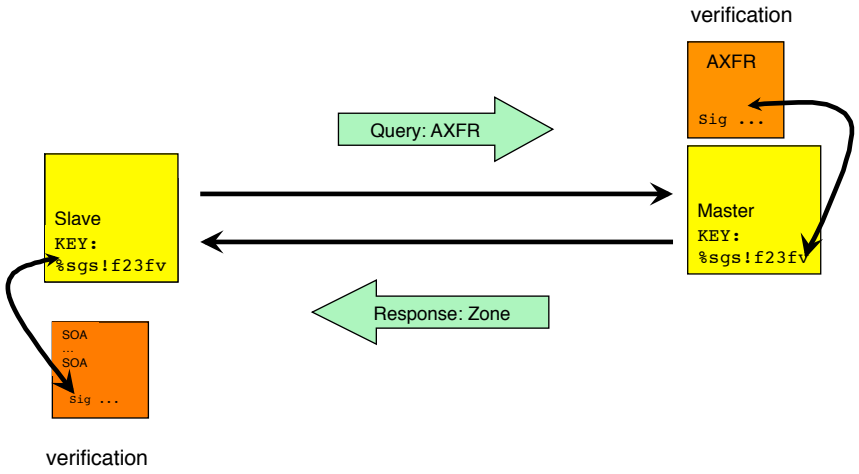
What is TSIG - Transaction Signature?

- A mechanism for protecting a message from a primary to secondary and vice versa
- A keyed-hash is applied (like a digital signature) so recipient can verify the message
  - DNS question or answer
  - & the timestamp
- Based on a shared secret - both sender and receiver are configured with it
  - TSIG/TKEY uses DH, HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA512 among others

What is TSIG - Transaction Signature?

- TSIG (RFC 2845)
  - authorizing dynamic updates & zone transfers
  - authentication of caching forwarders
- Used in server configuration, not in zone file

TSIG example



TSIG steps

1. Generate secret
2. Communicate secret
3. Configure servers
4. Test

TSIG - Names and Secrets

- TSIG name
  - A name is given to the key, the name is what is transmitted in the message (so receiver knows what key the sender used)
- TSIG secret value
  - A value determined during key generation
  - Usually seen in Base64 encoding

TSIG – Generating a Secret

- dnssec-keygen
  - Simple tool to generate keys
  - Used here to generate TSIG keys

```
> dnssec-keygen -a <algorithm> -b <bits> -n host
  <name of the key>
```

TSIG – Generating a Secret

- Example

```
> dnssec-keygen -a HMAC-MD5 -b 128 -n HOST ns1-
  ns2.pcx.net

This will generate the key
> Kns1-ns2.pcx.net.+157+15921

>ls
Kns1-ns2.pcx.net.+157+15921.key
Kns1-ns2.pcx.net.+157+15921.private
```


TSIG – Generating a Secret

- TSIG should never be put in zone files
  - might be confusing because it looks like RR:

```
ns1-ns2.pcx.net. IN KEY 128 3 157 nEfRX9…bbPn7lyQtE=
```

TSIG – Configuring Servers

- Configuring the key
  - in named.conf file, same syntax as for rndc
  - `key { algorithm ...; secret ...;}`
- Making use of the key
  - in named.conf file
  - `server x { key ...; }`
  - where 'x' is an IP number of the other server

Configuration Example – named.conf

Primary server 10.33.40.46

```
key ns1-ns2.pcx. net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.50.35 {
        keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type master;
        file “db.myzone”;
        allow-transfer {
        key ns1-ns2.pcx.net ;};
};
```

Secondary server 10.33.50.35

```
key ns1-ns2.pcx.net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.40.46 {
  keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type slave;
        file “myzone.backup”;
        masters {10.33.40.46;};
};
```

You can save this in a file and refer to it in the named.conf using ‘include’ statement:

```
include “/var/named/master/tsig-key-ns1-ns2”;
```

TSIG Testing : dig

- You can use dig to check TSIG configuration
  - `dig @<server> <zone> AXFR -k <TSIG keyfile>`

```
$ dig @127.0.0.1 example.net AXFR \
    -k Kns1-ns2.pcx.net.+157+15921.key
```

- Wrong key will give “Transfer failed” and on the server the security-category will log this.

TSIG Testing - TIME!

- TSIG is time sensitive - to stop replays
  - Message protection expires in 5 minutes
  - Make sure time is synchronized
  - For testing, set the time
  - In operations, (secure) NTP is needed

TSIG steps

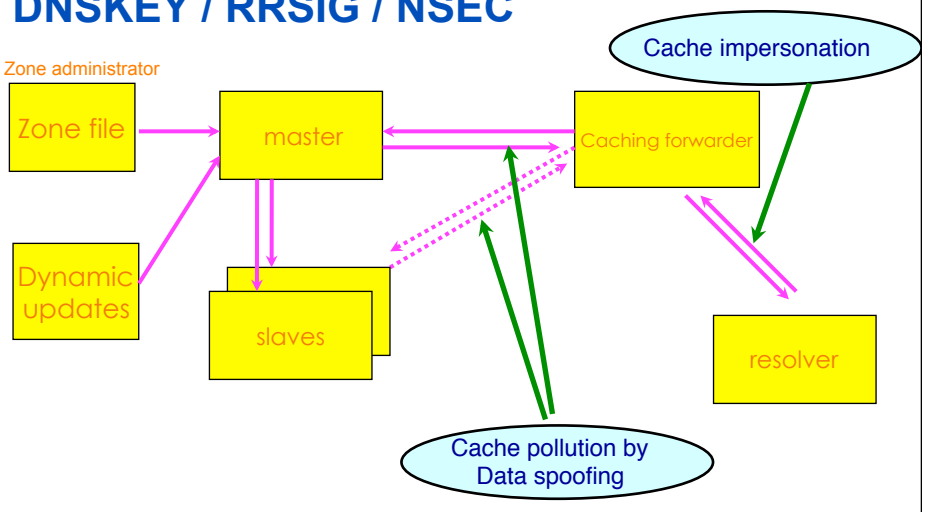
1. Generate secret
   - `dnssec-keygen -a <algorithm> -b <bits> -n host <name of the key>`
2. Communicate secret
   - `scp <keyfile> <user>@<remote-server>:<path>`
3. Configure servers
   - `key { algorithm ...; secret ...;}`
   - `server x { key ...; }`
4. Test
   - `dig @<server> <zone> AXFR -k <TSIG keyfile>`

Questions

DNSSEC

APNIC

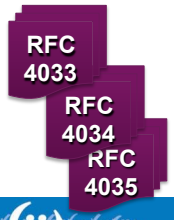
Vulnerabilities protected by DNSKEY / RRSIG / NSEC



APNIC

DNS Security Extensions (DNSSEC)

- Protects the integrity of data in the DNS by establishing a chain of trust
- Uses public key cryptography – each link in the chain has a public/private key pair
- A form of digitally signing the data to attest its validity
- Standard is defined in RFC4033, RFC4034, and RFC4035
- Guarantees
  - Authenticity
  - Integrity
  - Non-existence of a domain



APNIC

DNSSEC History

- **1990**: Steven Bellovin discovers a major flaw in the DNS
- **1995**: Bellovin publishes his research; DNSSEC (as it became later known) becomes a topic within IETF
- **1997**: RFC 2065 (adding security extensions) was published
- **1998**: Dan Kaminsky discovers some security flaw
- **1999**: RFC 2535, the DNSSEC protocol, is published; BIND 9 developed to be DNSSEC-capable
- **2001**: key handling in RFC2535 is causing operational problems
- **2005**: Three new RFCs published to update RFC2535
  - RFC 4033 (DNS Security Introduction and Requirements)
  - RFC 4034 (Resource Records for DNS Security Extensions)
  - RFC 4035 (Protocol Modifications)

https://wiki.tools.isoc.org/DNSSEC_History_Project

APNIC

DNSSEC History

- **2005**: In October, Sweden (.SE) becomes the first ccTLD to deploy DNSSEC
- **2008**: new DNSSEC record created to address privacy concerns (RFC 5155)
- **2010**
  - In July 15, the root zone was signed
  - In July 29, .edu was signed
  - In December 9, .net was signed
- **2011**: In March 31, `.com` was signed

https://wiki.tools.isoc.org/DNSSEC_History_Project

APNIC

DNSSEC Resource Records

- 3 Public key crypto related RRs
  - **RRSIG** = Signature over RRset made using private key
  - **DNSKEY** = Public key, needed for verifying a RRSIG
  - **DS** = Delegation Signer; 'Pointer' for building chains of authentication
- One RR for internal consistency
  - **NSEC** = Next Secure; indicates which name is the next one in the zone and which typecodes are available for the current name
    - authenticated non-existence of data

APNIC

DNSSEC Resource Records

- DNSKEY, RRSIG, and NSEC records provide mechanisms to establish authenticity and integrity of data
- DS record provides a mechanism to delegate trust to public keys of third parties

DNSSEC RRs

- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEY is used to verify the RRSIG
- Children sign their zones with their private key
  - Authenticity of that key established by signature/checksum by the parent (DS)
- Ideal case: one public DNSKEY distributed

RR's and RRsets

- Resource Record:

Name	TTL	class	type	rdata
www.example.net.	7200	IN	A	192.168.1.1

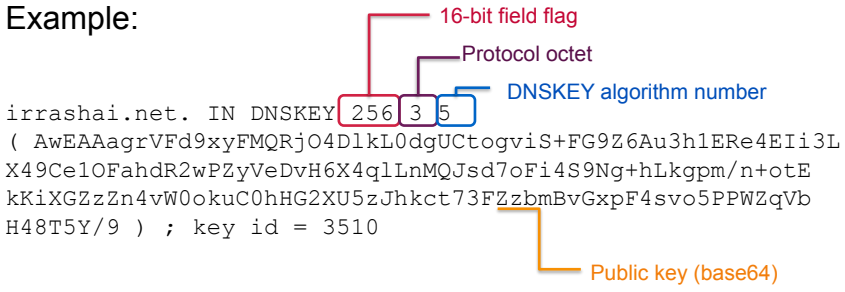
- RRset: RRs with same name, class and type:

www.example.net.	7200	IN	A	192.168.1.1
			A	10.0.0.3
			A	172.10.1.1

- RRsets are signed, not the individual RRs

DNSKEY

- Contains the zone's public key
- Uses public key cryptography to sign and authenticate DNS resource record sets (RRsets).
- Example:



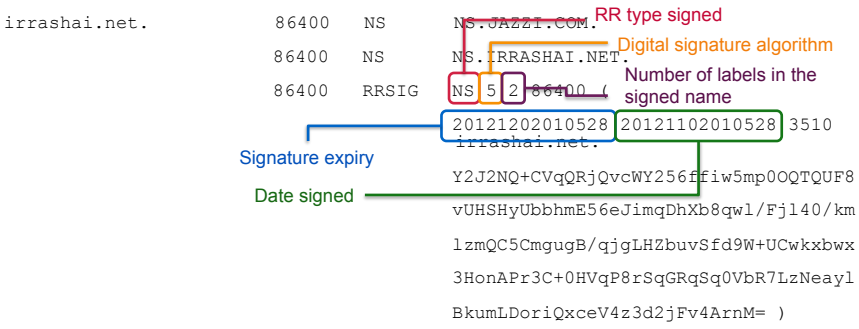
DNSKEY

- Also contains some timing metadata – as a comment in the key file

```
; This is a key-signing key, keyid 19996, for myzone.net.
; Created: 20121102020008 (Fri Nov  2 12:00:08 2012)
; Publish: 20121102020008 (Fri Nov  2 12:00:08 2012)
; Activate: 20121102020008 (Fri Nov  2 12:00:08 2012)
```


RRSIG

- The private part of the key-pair is used to sign the resource record set (RRset) per zone
- The digital signature per RRset is saved in an RRSIG record



NSEC / NSEC3

- Next Secure
- Forms a chain of authoritative owner names in the zone
- Lists two separate things:
  - Next owner name (canonical ordering)
  - Set of RR types present at the NSEC RR's owner name
- Also proves the non-existence of a domain

```
irrashai.net.  NSEC    blog.irrashai.net. A NS SOA MX
                  RRSIG NSEC DNSKEY
```


NSEC / NSEC3

- "The last NSEC wraps around from the last name in the ordered zone to the first"
- Each NSEC record also has a corresponding RRSIG

NSEC RDATA

- Points to the next domain name in the zone
  - also lists what are all the existing RRs for “name”
  - NSEC record for last name “wraps around” to first name in zone
- Used for authenticated denial-of-existence of data
  - authenticated non-existence of TYPEs and labels

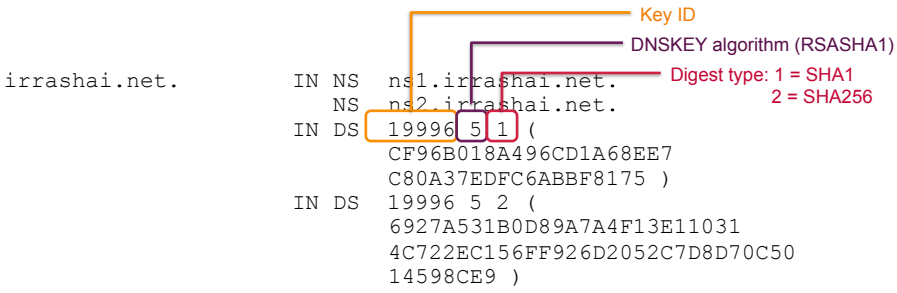
NSEC Record example

```
$ORIGIN example.net.
@ SOA       …
      NS     NS.example.net.
      DNSKEY …
      NSEC    mailbox.example.net. SOA NS NSEC DNSKEY       RRSIG

mailbox      A      192.168.10.2
                    NSEC  www.example.net.  A NSEC RRSIG
 WWW         A      192.168.10.3
                    TXT    Public webserver
                    NSEC  example.net. A NSEC RRSIG TXT
```

Delegation Signer (DS)

- Establishes the chain of trust from parent to child zones
- Found in the parent's zone file
- In this example, irrashai.net has been delegated from .net. This is how it looks like in .net zone file




```
irrashai.net.          IN NS  ns1.irrashai.net.
                       NS     ns2.irrashai.net.
                       IN DS  19996 5 1 (
                              CF96B018A496CD1A68EE7
                              C80A37EDFC6ABBF8175 )
                       IN DS  19996 5 2 (
                              6927A531B0D89A7A4F13E11031
                              4C722EC156FF926D2052C7D8D70C50
                              14598CE9 )
```


Delegation Signer (DS)

- Delegation Signer (DS) RR indicates that:
  - delegated zone is digitally signed
  - indicated key is used for the delegated zone
- Parent is authorative for the DS of the childs zone
  - Not for the NS record delegating the childs zone!
  - DS **should not** be in the childs zone

Types of Keys

- Zone Signing Key (ZSK)
  - Sign the RRsets within the zone
  - Public key of ZSK is defined by a DNSKEY RR
- Key Signing Key (KSK)
  - Signed the keys which includes ZSK and KSK and may also be used outside the zone
- Trusted anchor in a security aware server
- Part of the chain of trust by a parent name server
- Using a single key or both keys is an operational choice (RFC allows both methods)

Creation of keys

- Zones are digitally signed using the private key
- Can use RSA-SHA-1, DSA-SHA-1 and RSA-MD5 digital signatures
- The public key corresponding to the private key used to sign the zone is published using a DNSKEY RR

Chain of Trust

- DNSSEC is based on trust
- Root is on top of the chain of trust.
  - Root servers were signed on July 15, 2010.

APNIC

Implementing DNSSEC

APNIC

DNSSEC - Setting up a Secure Zone

- Enable DNSSEC in the configuration file (named.conf)
  - `dnssec-enable yes; dnssec-validation yes;`
- Create key pairs (KSK and ZSK)
  - `dnssec-keygen -a rsasha1 -b 1024 -n zone champika.net`
- Publish your public key
- Signing the zone
- Update the config file
  - Modify the zone statement, replace with the signed zone file
- Test with dig

Updating the DNS Configuration

- Enable DNSSEC in the configuration file (named.conf)

```
options {
        directory "…."
        dnssec-enable yes;
        dnssec-validation yes;
 };
```

- Other options that can be added later
  - `auto-dnssec { off | allow | maintain} ;`
  - These options are used to automate the signing and key rollover

Creating key pairs

- To create ZSK

```
dnssec-keygen -a rsasha1 -b 1024 -n zone
<myzone>
```

- To create KSK

```
dnssec-keygen -a rsasha1 -b 1400 -f KSK -n
zone champika.net
```


Generating Key Pair

- Generate ZSK and KSK

```
dnssec-keygen -a rsasha1 -b 1024 -n zone <myzone>
```

Default values are RSASHA1 for algorithm, 1024 bits for ZSK and 2048 bits for KSK

The command above can be simplified as:

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

This generates four files.

Note: There has to be at least one public/private key pair for each DNSSEC zone

Publishing your public key

- Using $INCLUDE you can call the public key (DNSKEY RR) inside the zone file

```
$INCLUDE /path/Kchampika.net.+005+33633.key ; ZSK
$INCLUDE /path/Kchampika.net.+005+00478.key ; KSK
```

- You can also manually enter the DNSKEY RR in the zone file

Signing the Zone

- Sign the zone using the secret keys:

```
dnssec-signzone –o <zonename> -N INCREMENT -f
<output-file> -k <KSKfile> <zonefile> <ZSKfile>

dnssec-signzone –o champika.net db.champika.net
Kchampika.net.+005+33633
```

- Once you sign the zone a file with a .signed extension will be created
  - `db.champika.net.signed`

Signing the Zone

- Note that only authoritative records are signed
  - NS records for the zone itself are signed
  - NS records for delegations are not signed
  - DS RRs are signed!
  - Glue is not signed
- Difference in the file size
  - db.champika.net vs. db.champika.net.signed

Smart Signing

- Searches the key repository for any keys that will match the zone being signed

```
options {
keys-directory { "path/to/keys"; };
```

- Then the command for smart signing is

```
dnssec-signzone -S db.myzone.net
```

Publishing the Zone

- Reconfigure to load the signed zone. Edit named.conf and point to the signed zone.

```
zone "<myzone>" {
     type master;
     # file "db.myzone.net";
     file "db.myzone.net.signed";
};
```


Pushing the DS record

- The DS record must now be published by the parent zone.
- Contact the parent zone to communicate the KSK to them.

KSK Key Rollover

- Perform scheduled zone maintenance.
- KSK rollover using Double signing
- When you change the KSK keys, the DS record in the parent zone must also be updated.

```
dnssec-signzone -o myzone.net -N increment -f <output- \
file> -k Kmyzone.net.+005+11111 db.myzone.net \
Kmyzone.net.+005+67890
```

- Send the new DS record to the parent, and wait for it to propagate.
- Remove the old key and resign.

KSK Key Rollover

- Using Pre-publication
- In this method, the new key will be published but will not be used for signing yet.

```
dnssec-keygen -K keydir -f ksk -A none <myzone.net>
rndc loadkeys <myzone.net>
```

- Publish both keys, but use only the old one for signing
- Wait for the propagation time and TTL of the DNSKEY RR to expire.

KSK Key Rollover

- Then use dnssec-settime once you are ready to sign the zone. Use the new key for zone signing, leaving the old one published.

```
dnssec-settime -K keydir -A now Kexample.com.+005+12345
rndc loadkeys example.com
```

- Wait for the propagation and TTL in the old zone. Set the old key to no longer sign with the key, but leaves it in the zone.

```
dnssec-settime -K keydir -I now Kexample.com.+005+12345
rndc loadkeys example.com
```

- Now remove the old keys. This completely removes the keys.

```
dnssec-settime -K keydir -D now Kexample.com.+005+12345
rndc loadkeys example.com
```


Automated Signing

- Using RNDC
- Add the option to named.conf

```
auto-dnssec allow;
```

- Then you can use the commands:

```
rndc loadkeys zone
rndc sign zone
```

Testing the server

- Ask a dnssec enabled question from the server and see whether the answer contains dnssec-enabled data
  - Basically the answers are signed

```
dig @localhost www.champika.net +dnssec
+multiline
```


Testing with dig: an example

Registries / Hosting Providers – Sign your Zones

- All the zones you serve can be signed
  - think about key rollover
  - think about key compromise scenarios, and what processes you will follow when you detect them
- Think about how you can detect compromises, and monitor signatures

For ISPs – Validate!

- The most effective step you can take to encourage DNSSEC uptake as an ISP is to validate responses
- DNSSEC-signed zones are fairly new, so expect this to cause some non-zero (but manageable) amount of helpdesk load
- Comcast is an example of a large ISP (in the US) who has taken this step

DNSSEC Validation - Tools

- Use a validating resolver
  - From your ISP
  - From Google Public DNS – 8.8.8.8
  - From DNS-OARC - 149.20.64.20, 149.20.64.21
- DNS/TLSA Validator
  - https://www.dnssec-validator.cz/
- Web tools
  - DNSSEC Failed - http://www.dnssec-failed.org/
  - DNSSEC Debugger (Verisign) - http://dnssec-debugger.verisignlabs.com/
  - DNSSEC or NOT - http://test.dnssec-or-not.com/

APNIC

DNSSEC Key Management

APNIC

Ways to Deploy DNSSEC

- As part of the DNS software used
  - Manual key management
  - Can be quite complex
  - For static environment
  - Some means of automation using
    - option commands and scripts

DNSSEC tools for BIND, NSD, PowerDNS, etc

- Use with a hardware security module (HSM)
  - Semi-automatic
  - Good for dynamic environment

HSM, OpenDNSSEC

- Using an external appliance
  - 'dnssec-in-a-box'
  - Fully automates key generation, signing and rollover

DNS Appliance

Hardware Security Module

- Cryptographic devices used for storage of the encryption keys
  - Smart cards, PCI cards, USB tokens
- It also speeds up the cryptographic key generation
- Implements PKCS#11 (Cryptographic Token Key Interface)
  - A standard interface or API to cryptographic tokens

DNSSEC Signer Appliance

- Can be a pure signer or packaged with an IPAM or a DNS server
- In pure signer, the hardware appliance interfaces between the master/slave servers
- Examples: Secure64, Xelerance, SolidDNS, etc

APNIC

Questions

APNIC

Further Readings

- DNS Operational Practices
  - http://tools.ietf.org/html/rfc6781
  - Informational, published December 2012
- A Framework for DNSSEC Policies and DNSSEC Practice Statements
  - http://tools.ietf.org/html/rfc6841
  - Informational, published January 2013

APNIC

Thank you!

End of Session

APNIC