Introduction to DNS and its vulnerabilities

Olaf M. Kolkman
olaf@nlnetlabs.nl
The DNS is highly distributive.
The DNS is highly distributive.

Device queries Recursive Nameserver.

Recursive Nameserver Recurses over Authoritative nameservers.

Results are cached.

The DNS is highly distributive.

DNS is implemented through 100s of thousands of machines.
Stub Resolver -> Recursive Nameservers

www.nlnetlabs.nl A

Authoritative Nameservers

ROOT

NL

NLnetLabs.NL

Thursday, February 16, 12
Stub Resolver

Recursive Nameservers

Authoritative Nameservers

root.hints: location of the root servers

www.nlnetlabs.nl A
Stub Resolver

Recursive Nameservers

www.nlnetlabs.nl A

referral: nl NS

Authoritative Nameservers

www.nlnetlabs.nl A

root.hints: location of the root servers
Stub Resolver

Recursive Nameservers

Authoritative Nameservers

www.nlnetlabs.nl A

referral: nl NS

www.nlnetlabs.nl A

root.hints: location of the root servers
Stub Resolver

Recursive Nameservers

www.nlnetlabs.nl A

referral: nl NS

www.nlnetlabs.nl A

referral: nlnetlabs.nl NS

Authoritative Nameservers

ROOT

NL

root.hints: location of the root servers
Stub Resolver

Recursive Nameservers

www.nlnetlabs.nl A

referral: nl NS

www.nlnetlabs.nl A

referral: nlnetlabs.nl NS

www.nlnetlabs.nl A

referral: nl NS

www.nlnetlabs.nl A

referral: nlnetlabs.nl NS

www.nlnetlabs.nl A

Authoritative Nameservers

www.nlnetlabs.nl A

root.hints: location of the root servers
Stub Resolver

Recursive Nameservers

www.nlneatls.nl A

referral: nl NS

www.nlneatls.nl A

referral: nlnetlabs.nl NS

Authoritative Nameservers

www.nlneatls.nl A

Answer: www.nlneatls.nl A 213.154.224.1

root.hints: location of the root servers
Stub Resolver

Recursive Nameservers

www.nlnetlabs.nl A 213.154.224.1

referral: nl NS

www.nlnetlabs.nl A

referral: nlnetlabs.nl NS

www.nlnetlabs.nl A

Answer: www.nlnetlabs.nl A 213.154.224.1

Authoritative Nameservers

root.hints: location of the root servers
Attack Surface
Attack Surface

- Bugs and implementation mistakes
- On the Wire or through Compromise
- Compromise of systems

Thursday, February 16, 12
Attack Surface

- Bugs and implementation mistakes
- On the Wire or through Compromise
- Compromise of systems

Whoa, that looks bad!!! Who Uses This System?
Recursive Nameserver Query:
<qname, qtype, qclass, id>

STUB Resolver

Authoritative Nameserver

Atacker

Response:
<name, type, class, id>

Cache hit
Recursive Nameserver
Query: `<qname, qtype, qclass, id>`

STUB Resolver
Response: `<name, type, class, id>`

Authoritative Nameserver
Query: `<qname, qtype, qclass, id>`

Atacker
Success depends on legacy and speed of network.

Cache hit
Response: `<name, type, class, id>`

Response: `<name, type, class, id>`
And on various properties that the attacker needs to match

Success depends on legacy and speed of network.
Recursive Nameserver Query:
\(<qname, qtype, qclass, id>\)

STUB Resolver

Authoritative Nameserver

Atacker

Query:
\(<qname, qtype, qclass, id>\)

Response:
\(<name, type, class, id>\)

Cache hit

Query ID

Success depends on legacy and speed of network.

And on various properties that the attacher needs to match
And on various properties that the attacker needs to match

Success depends on legacy and speed of network.

Recursive Nameserver

Authoritative Nameserver

Atacker

Query:

<qname, qtype, qclass,id>

Response:

<name, type, class,id>

Cache hit

Query ID

Source Port

STUB Resolver
And on various properties that the attacher needs to match

Query ID 0x20

Success depends on legacy and speed of network.
TTL saves you?!?
I don’t think so....
Recursive Nameserver

Query: asdf23sadf.webcam.com

Response: asdf23sadf.webcam.com

Query: www.webcam.com

Response: www.webcam.com

Response: webcam.com NS ns1.webcam.com
ns1.webcam.com A 10.6.6.6

Query: asdf23sadf.webcam.com

Response: asdf23sadf.webcam.com

Query to 10.6.6.6

Query to 10.6.6.6

Try Delegations

Abuse a 25 year old protocol requirement
Do attacks happen in practice?

Would you notice?

Would you tell?
Do attacks happen in practice?

Why would one attack the DNS?

While one could be doing other things.
How to Protect?
Follow the Money

- Why would one attack the DNS?
- Organizing your life
- Paying your Tax
- Short-selling your stock
- Your weekly security update
Money

Don’t all these transactions use SSL and Certificates?
The role of a CA

3rd party trust broker
The role of a CA

3rd party trust broker

Subject Requests
The role of a CA

3rd party trust broker

Subject Requests

RA performs checks
The role of a CA

3rd party trust broker

Subject Requests

RA performs checks

RA tells CA to sign
The role of a CA

3rd party trust broker

Subject Requests

RA performs checks

RA tells CA to sign

Browser trusts CA signed certificates
The role of a CA

3rd party trust broker

Subject Requests

RA performs checks

RA tells CA to sign

Browser trusts CA signed certificates

EV

Extended validation

Thursday, February 16, 12
However all these little men are a wee bit expensive
However all these little men are a wee bit expensive

AUTOMATE THE LOT
DV
Domain Validation

Subject: Please sign certificate for Example.com

RA sends a mail to well known address @example.com

When mail returned CA will sign
Domain Validation

All these checks are based on information fetched from the DNS

Hold that thought for Jakob’s presentation
DNS System Vulnerabilities

Provisioning Vulnerabilities

Registrars & Registrants

Server vulnerability

Man in the Middle

Primary DNS

Secondary DNS

Secondary DNS Server Vulnerability

Man in the Middle Spoofing & Man in the Middle

Thursday, February 16, 12
What can one do to protect...(skipping DNSSEC)
Taking Unbound as example

Other servers might make other choices, but any modern resolver takes similar approaches.
Security Choices in Unbound

• In general, a modern paranoid resolver

• DNSSEC support.

• RFC 2181 support completely
  • Fine grained. Keeps track of where RRSets came from and won't upgrade them into answers.

• Does not allow RRSets to be overridden by lower level rrsets
Filtering

- Scrubber:
  - Only in-bailiwick data is accepted in the answer
    - The answer section must contain only answer
  - CNAME, DNAME checked that chain is correct
    - CNAME cut off and only the first CNAME kept
      - Lookup rest yourself do not trust other server
  - DNAME synthesize CNAME by unbound do not trust other server. Also cut off like above.
  - DNAME from cache only used if DNSSEC-secure.
Filtering II

- No address records in authority, additional section unless relevant – i.e. mentioned in a NS record in the authority section.

- Irrelevant data is removed

- When the message only had preliminary parsing and has not yet been copied to the working region of memory
Entropy

- Randomness protects against spoof
- `Arc4random()` (OpenBSD): crypto strong. May not be perfectly random, but predicting it is a cryptographical breakin.
- Real entropy from OS as seed
- Query id – all 16 bits used.
- Port randomisation – uses all 16 bits there, goes out of its way to make sure every query gets a fresh port number
Entropy II

- Destination address, and ipv4/ipv6. RTT band of 400msec (=everything).
  - Its not the timewindow but the randomness
- Query aggregation – same queries are not sent out – unless by different threads
- Qname strict match checked in reply
- 0x20 option
- Harden-referral-path (my draft) option
- Can use multiple source interfaces!
  - 4 outgoing IP address add +2 bits
Other measures

- Not for the wire itself
  - Heap function pointer protection (whitelisted)
  - Chroot() by default
  - User privileges are dropped (lots of code!)
  - ACL for recursion
  - No detection of attacks – assume always under attack
  - version.bind hostname.bind can be blocked or configured what to return (version hiding)
  - Disprefer recursion lame servers – they have a cache that can be poisoned
Arms Race...

Introducing DNSSEC
End to End Security

Registrars & Registrants

Registry

Primary DNS

Secondary DNS

Secondary DNS

End to End Security
All done using Public Key crypto

DNSKEY: public key from the keypair

RRSIG: Signatures made with a private key from the keypair

NSEC and NSEC3 For pre-calculated Denial of Existence

DS For delegating Security
But more on that later

Let us have a look at another cryptographic DNS protection mechanism.
Securing Host-Host Communication
Data flow through the DNS

What should you protect...

- Registrars & Registrants
- Host Security
- TSIG
- Secondary DNS
- Primary DNS

TSIG (rarely)
Data flow through the DNS

- Registrars & Registrants
- Registry
- Primary DNS
- Secondary DNS
- HOST Security
- TSIG
- TSIG (rarely)
Transaction Signature: TSIG

- TSIG (RFC 2845)
  - Authorising dynamic updates and zone transfers
  - Authentication of caching forwarders
  - Independent from other features of DNSSEC
- One-way hash function
  - DNS question or answer and timestamp
- Traffic signed with “shared secret” key
- Used in configuration, **NOT** in zone file
TSIG Example

Query: AXFR

Slave

KEY: $h@r3dS3cr3t

Master

Thursday, February 16, 12
TSIG Example

Slave

KEY: $h@r3dS3cr3t

Master

KEY: $h@r3dS3cr3t

Query: AXFR
Master

KEY: $h@r3dS3cr3t

Slave

KEY: $h@r3dS3cr3t

Query: AXFR

AXFR

AXFR
TSIG Example

Master

KEY: $h@r3ds3cr3t

Slave

KEY: $h@r3dS3cr3t

Query: AXFR

AXFR

Sig: B1@F00
TSIG Example

Slave

KEY: $h@r3dS3cr3t

Query: AXFR

AXFR

Sig: B1@F00

Master

KEY: $h@r3dS3cr3t
TSIG Example

Query: AXFR

Slave
KEY: $h@r3dS3cr3t

Master
KEY: $h@r3dS3cr3t

AXFR
Sig: B1@F00

verification

Thursday, February 16, 12
TSIG Example

Slave

KEY: $h@r3dS3cr3t

Query: AXFR

AXFR

Sig: B1@F00

verification

Response: Zone

Master

KEY: $h@r3dS3cr3t
TSIG Example

Query: AXFR

Slave

KEY: $h@r3dS3cr3t

Master

KEY: $h@r3dS3cr3t

Response: Zone

AXFR
Sig: B1@F00

verification

SOA
... SOA
TSIG Example

Slave

KEY: $h@r3dS3cr3t

Query: AXFR

Master

KEY: $h@r3dS3cr3t

AxFR

Sig: B1@F00

Verification

Response: Zone

Query: AXFR

Response: Zone

AxFR

Sig: B1@F00

SOA ...

SOA

SIG: FOOB@R

Thursday, February 16, 12
**TSIG Example**

**Query: AXFR**

**Slave**
- KEY: $h@r3dS3cr3t

**Master**
- KEY: $h@r3dS3cr3t
  - SOA
  - ... 
  - SOA
  - SIG: FOOB@R

**Response: Zone**

**AXFR**
- Sig: B1@F00

**verification**
TSIG Example

Slave

KEY: $h@r3dS3cr3t

SOA...
SOA
SIG: FOOB@R

Response: Zone

Query: AXFR

Master

KEY: $h@r3dS3cr3t

SOA...
SOA
SIG: FOOB@R

AXFR
Sig: Bl@F00

verification

Query: AXFR
Response: Zone
TSIG Example

Query: AXFR

Slave
KEY: $h@r3dS3cr3t

SOA...
SOA
SIG: FOOB@R

verification

Master
KEY: $h@r3dS3cr3t

SOA...
SOA
SIG: FOOB@R

verification

Response: Zone

AXFR
Sig: B1@F00
TSIG for Zone Transfers

1. Generate secret
2. Communicate secret
3. Configure servers
4. Test
Importance of the Time Stamp

• TSIG/SIG(0) signs a complete DNS request / response with time stamp
  – To prevent replay attacks
  – Currently hardcoded at five minutes

• Operational problems when comparing times
  – Make sure your local time zone is properly defined
  – `date -u` will give UTC time, easy to compare between the two systems
  – Use NTP synchronisation!
Authenticating Servers Using SIG(0)

• Alternatively, it is possible to use SIG(0)
  – Not yet widely used
  – Works well in dynamic update environment

• Public key algorithm
  – Authentication against a public key published in the DNS

• SIG(0) specified in RFC 2931
Cool Application

• Use TSIG-ed dynamic updates to configure your laptops name

• My laptop is known by the name of aagje.secret-wg.org
  - Mac OS users: there is a bonjour based tool.

  • www.dns-sd.org