The Domain Name System (DNS) and its security

CSU CS557 - Fall 2017
Partly based on the CS457 slides by Indrajit Ray

Lorenzo De Carli
What is the domain name system?

- Internet use 4-byte **binary addresses**
  - Humans are not very good at remembering those
- Solution: allow pesky humans to use **strings**
  - E.g. “www.cs.colostate.edu"
- Let the **domain name system** convert **strings** to **IP addresses**
Domain Name System

- The Domain System consist of:
  - An application-layer protocol to translate names to IP addresses (“resolve names”)
  - A distributed database to perform the translation (consisting of name servers)
  - Almost every networked communication is preceded by a DNS resolution to determine the IP that the user/application intends to reach
DNS resolution

Multiple servers for each zone in case any one server fails
13 root servers
13 edu servers
5 colostate.edu servers

End-user

www.colostate.edu A?
www.colostate.edu A is 129.82.64.100

Caching DNS Server

Root DNS Server
edu DNS Server
colostate.edu DNS Server
DNS resolution

“A” is one type of DNS record, but DNS servers can return several different types of records:
- A -> IP address
- AAAA -> IPv6 address
- ...

Multiple servers for each zone in case any one server fails
- 13 root servers
- 13 edu servers
- 5 colostate.edu servers

End-user
www.colostate.edu A?
www.colostate.edu A is 129.82.64.100

Root DNS Server
edu DNS Server
colostate.edu DNS Server

Caching DNS Server
Advantages of separating naming and addressing

- Main advantage: names are easier to remember
  - cnn.com VS 64.236.16.20

- Names make a certain endpoint independent from the IP address
  - If cnn.com moves, it can simply change the mapping of cnn.com to a different IP address

- Names can map to multiple/different IP addresses (used for load balancing, localization)

- Enable aliasing (multiple names for same address)
Other solutions

• Initial approach: static mapping between names and IPs, stored in a text file (/etc/hosts)
  • Master kept at SRI, Menlo Park, CA
• Obviously does not scale well
Other solutions - II

- Can I get by using a single central server?
- Yeah, right!
  - Think that more or less every connection…
  - …on every host…
  - is preceded by a DNS request/response
  - No server, no matter how powerful, could sustain this load!
- Also, it would constitute SOP for the whole Internet!
The DNS infrastructure is a distributed system

• Names are hierarchical

  • E.g. www.colostate.edu

  • Servers are organized hierarchically too!

• Name space divided in zones
DNS server hierarchy

- Data organized as tree structure
  - Each zone is authoritative for its local data.
- Each zone operates a set of name servers that contain the zone data
  - Change to host.cs.colostate.edu is entered at cs.colostate.edu servers.
- Tree structure directs queries to the appropriate name server
  - Root knows how to reach edu
  - Edu knows how to reach colostate.edu
  - Etc.
DNS server hierarchy

**DNS Root Servers**

- Labeled A through M
  - B USC - ISI Marina del Rey, CA
  - L ICANN Los Angeles, CA
  - E NASA Mt View, CA
  - F Internet Software C. Palo Alto, CA (and 17 other locations)
  - I Autonomica, Stockholm (plus 3 other locations)
  - K RIPE London (also Amsterdam, Frankfurt)
  - m WIDE Tokyo
  - A Verisign, Dulles, VA
  - C Cogent, Herndon, VA (also Los Angeles)
  - D U Maryland College Park, MD
  - G US DoD Vienna, VA
  - H ARL Aberdeen, MD
  - J Verisign, (11 locations)

**Types of DNS Servers**

- **Authoritative DNS servers:**
  - Provide authoritative records for a particular zone (e.g., colostate.edu, cisco.com, edu, uk, etc)
  - Can be maintained locally or by a service provider
- **Top-level domain (TLD) servers:**
  - Authoritative servers responsible for com, org, net, edu, etc, and all top-level country domains (uk, fr, ca, jp).
  - Typically managed professionally
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD

- **Caching Servers**
  - Accept queries for end hosts, lookup requested data, and cache answers for later replies.
  - Data organized as a tree structure
  - Each zone is authoritative for its local data
  - Each zone operates a set of name servers that contain the zone data
  - Change to host.cs.colostate.edu is entered at cs.colostate.edu servers.
  - Tree structure directs queries to the appropriate name server
  - Root knows how to reach edu
  - Edu knows how to reach colostate.edu
  - Etc.
DNS server hierarchy

11 root servers (see http://www.root-servers.org/)

- Labeled A through M
  - B USC - ISI Marina del Rey, CA
  - ICANN Los Angeles, CA
  - E NASA Mt View, CA
  - F Internet Software C. Palo Alto, CA (and 17 other locations)
  - I Autonomica, Stockholm (plus 3 other locations)
  - K RIPE London (also Amsterdam, Frankfurt)
  - M WIDE Tokyo

Types of DNS Servers

- Authoritative DNS servers:
  - provide authoritative records for a particular zone (e.g., colostate.edu, cisco.com, edu, uk, etc)
  - Can be maintained locally or by a service provider

- Top-level domain (TLD) servers:
  - Authoritative servers responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Typically managed professionally
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD

- Caching Servers
  - Accept queries for end hosts, lookup requested data, and cache answers for later replies.

- Data organized as tree structure
  - Each zone is authoritative for its local data.
  - Each zone operates a set of name servers that contain the zone data
  - Change to host.cs.colostate.edu is entered at cs.colostate.edu servers.
  - Tree structure directs queries to the appropriate name server
  - Root knows how to reach edu
  - Edu knows how to reach colostate.edu
  - Etc.

DNS Organization

Authoritative servers (maintain records for their respective zones)
DNS server hierarchy

Types of DNS Servers

- **Authoritative DNS servers:**
  - Provide authoritative records for a particular zone (e.g., colostate.edu, cisco.com, edu, uk, etc)
  - Can be maintained locally or by a service provider
- **Top-level domain (TLD) servers:**
  - Authoritative servers responsible for com, org, net, edu, etc, and all top-level country domains (uk, fr, ca, jp).
  - Typically managed professionally
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD

**Caching Servers**

- Accept queries for end hosts, lookup requested data, and cache answers for later replies.
- Data organized as a tree structure
- Each zone is authoritative for its local data
- Each zone operates a set of name servers that contain the zone data
- Change to host.cs.colostate.edu is entered at cs.colostate.edu servers
- Tree structure directs queries to the appropriate name server
- Root knows how to reach edu
- Edu knows how to reach colostate.edu
- Etc.
DNS server hierarchy

- **Root servers**
  - 13 root servers (see http://www.root-servers.org/)
  - Labeled A through M
  - B USC- ISI Marina del Rey, CA
  - L ICANN Los Angeles, CA
  - E NASA Mt View, CA
  - F Internet Software C. Palo Alto, CA (and 17 other locations)
  - I Autonomica, Stockholm (plus 3 other locations)
  - K RIPE London (also Amsterdam, Frankfurt)
  - m WIDE Tokyo
  - A Verisign, Dulles, VA
  - C Cogent, Herndon, VA (also Los Angeles)
  - D U Maryland College Park, MD
  - G US DoD Vienna, VA
  - H ARL Aberdeen, MD
  - J Verisign, 11 locations

- **Types of DNS Servers**
  - **Authoritative DNS servers**:
    - Provide authoritative records for a particular zone (e.g., colostate.edu, cisco.com, edu, uk, etc)
    - Can be maintained locally or by a service provider
  - **Top-level domain (TLD) servers**:
    - Authoritative servers responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
    - Typically managed professionally
    - Network solutions maintains servers for com TLD
    - Educause for edu TLD

- **Caching Servers**
  - Accept queries for end hosts, lookup requested data, and cache answers for later replies.
  - Data organized as tree structure
  - Each zone is authoritative for its local data.
  - Each zone operates a set of name servers that contain the zone data
  - Change to host.cs.colostate.edu is entered at cs.colostate.edu servers.
  - Tree structure directs queries to the appropriate name server
    - Root knows how to reach edu
    - Edu knows how to reach colostate.edu
    - Etc.
DNS server hierarchy

- **Root servers:**
  - Provide authoritative records for TLD servers

- **TLD servers:**
  - Provide authoritative records of zones in their subdomains (e.g. the TLD server for .edu has a record for colostate.edu)
Root servers

• 13 root servers (see http://www.root-servers.org/)
• Labeled A through M

A Verisign, Dulles, VA
B USC-ISI Marina del Rey, CA
C Cogent, Herndon, VA (also Los Angeles)
D U Maryland College Park, MD
E NASA Mt View, CA
F Internet Software C. Palo Alto, CA (and 17 other locations)
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign, (11 locations)
K RIPE London (also Amsterdam, Frankfurt)
I Autonomica, Stockholm (plus 3 other locations)
L ICANN Los Angeles, CA
m WIDE Tokyo
DNS Caches

• Most users do not directly contact authoritative servers for any resolution

• Instead, they contact a local DNS cache (e.g. provided by their ISP)

• The local DNS takes care of contacting the appropriate servers in the network, cache the resolved IP address, and forward it to the requesting client
DNS resolution

• Recursive/iterative queries

• Recursive query: the contacted DNS resolver takes the burden of performing the resolution process

• Iterative query: the contacted DNS server replies with name of another server to contact

  • “I don’t know this name, but ask this server”
DNS resolution

- Recursive/iterative queries

  - Recursive query: the contacted DNS resolver takes the burden of performing the resolution process

  - Iterative query: the contacted DNS server replies with name of another server to contact

  - “I don’t know this name, but ask this server”
DNS resolution

- Recursive/iterative queries

- Recursive query: the contacted DNS resolver takes the burden of performing the resolution process

- Iterative query: the contacted DNS server replies with name of another server to contact

  - “I don’t know this name, but ask this server”

Who has edu?

Who has cs.ucla.edu?
DNS resolution

- Recursive/iterative queries

- Recursive query: the contacted DNS resolver takes the burden of performing the resolution process

- Iterative query: the contacted DNS server replies with name of another server to contact
  - “I don’t know this name, but ask this server”

- Types of Queries
  - Recursive query: the contacted DNS resolver takes the burden of performing the resolution process
  - Iterative query: the contacted DNS server replies with name of another server to contact
    - “I don’t know this name, but ask this server”

- DNS Caching
  - Performing all these queries takes time
    - And all this before the actual communication takes place
      - E.g., 1-second latency before starting Web download
  - Caching can substantially reduce overhead
    - The top-level servers very rarely change
    - Popular sites (e.g., www.cnn.com) visited often
      - Local DNS server often has the information cached
  - How DNS caching works
    - DNS servers cache responses to queries
    - Responses include a “time to live” (TTL) field
    - Server deletes the cached entry after TTL expires

Who has edu?
Who has ucla.edu?
Who has cs.ucla.edu?
DNS resolution

• Recursive/iterative queries

• Recursive query: the contacted DNS resolver takes the burden of performing the resolution process

• Iterative query: the contacted DNS server replies with name of another server to contact

  • “I don’t know this name, but ask this server”
Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do `gethostbyname()` to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional `gethostbyaddr()` to translate into name

Types of Queries

- Recursive query:
  - Puts burden of name resolution on contacted name server
  - Heavy load?
  - Query 1 is recursive
- Iterated query:
  - Contacted server replies with name of server to contact
  - "I don't know this name, but ask this server"
  - Queries 2, 4, and 6 are iterative

DNS Caching

- Performing all these queries takes time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached

How the heck do I get to this guy? Who should I ask?

A slight issue
A slight issue

How the heck do I get to this guy? Who should I ask?

Solution: the addresses of the 13 root servers are typically hardcoded in name resolution software
DNS caching

- Performing all these queries takes time
  - And all this is **before** the communication even takes place

- In order to reduce overhead, **caching** is used
  - Local DNS server caches responses to queries
  - Responses include a time to live (TTL) field
  - Cached entries are deleted when TTL expires
DNS records

• What are these records anyway?

• The DNS system is a massive distributed database, and the records are the entries of the database

• Record are in the generic format (name, value, type, ttl)

• Records are not just for translating names to IPv4 addresses
DNS record types

- **Type A**: name is hostname, value is IPv4 address
- **Type AAAA**: name is hostname, value is IPv6 address
- **Type CNAME**: name is hostname, value is a canonical name for the same host
  - E.g. `www.ibm.com`'s canonical name is `servereast.backup2.ibm.com`
- **Type MX**: name is a domain name, value is a mail server for that domain
- **Type NS**: name is a domain name, value is the name of the authoritative DNS server for that domain
Glue records

• Parent zone (e.g. .com) stores authoritative name server for a child zone (e.g. "myzone.com NS ns.myzone.com")

• However, in order to reach ns.myzone.com I would need to query the authoritative server for myzone.com, which is ns.myzone.com!

• Catch-22

• The solution is to store a glue record in the authoritative name server for the .com zone: “ns.myzone.com A 129.12.37.1”
DNS queries

**DNS protocol**: *query* and *reply* messages, both with same *message format*

**msg header**
- **identification**: 16 bit # for query, reply to query uses same #
- **flags**:  
  - query or reply  
  - recursion desired  
  - recursion available  
  - reply is authoritative

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
<tr>
<td>questions</td>
<td>answers</td>
</tr>
<tr>
<td></td>
<td>authority</td>
</tr>
<tr>
<td></td>
<td>additional information</td>
</tr>
</tbody>
</table>

12 bytes
DNS queries - II

Name, type fields for a query

RRs in response to query

Records for authoritative servers

Additional "helpful" info that may be used

<table>
<thead>
<tr>
<th>Identification</th>
<th>flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of questions</td>
<td>number of answer RRs</td>
</tr>
<tr>
<td>number of authority RRs</td>
<td>number of additional RRs</td>
</tr>
</tbody>
</table>

- Questions (variable number of questions)
- Answers (variable number of resource records)
- Authority (variable number of resource records)
- Additional information (variable number of resource records)

Reliability

- DNS servers are replicated – Name service available if at least one replica is up – Queries can be load balanced between replicas
- UDP used for queries – Need reliability: must implement this on top of UDP
- Try alternate servers on timeout – Exponential backoff when retrying same server
- Same identifier for all queries – Don’t care which server responds

Inserting Resource Records into DNS

- Example: just created startup “FooBar”
- Register foobar.com at Network Solutions – Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
- Registrar inserts two RRs into the com TLD server:
  - (foobar.com, dns1.foobar.com, NS)
  - (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
- Type A record for www.foobar.com
- Type MX record for foobar.com
DNS reliability

• DNS servers are replicated
  • Name service available if at least one replica is up
  • Queries can be load-balanced between replicas

• **UDP** used for queries
  • What if message lost? RFC 1536 recommends retransmission timer w/ RTT estimation and exponential back-off…
  • … or try a different server/replica
  • Retransmitted queries all carry the same identifier
How are DNS records created?

• Example: company “foobar” wants to register its domain

• Register foobar.com at Network Solutions
  
  • Provide registrar of name and IP address of authoritative name server

  • Registrar inserts two records into com TLD server:

    • (foobar.com, dns1.foobar.com, NS)

    • (dns1.foobar.com, 212.212.212.1, A)

• Company add two records to authoritative server dns1.foobar.com:

  • (www.foobar.com, 212.212.212.10, A)

  • (foobar.com, mail.foobar.com, MX)
DNS security
Security in standard DNS

- Standard DNS messages have limited protection against spoofing
  - 16-bit query ID must be repeated in the reply
  - The source UDP port is chosen randomly
  - Not a very strong protection
Why is the query ID not enough?

- Attack #1: *sniffing*+*spoofing*

- As no encryption is applied to DNS messages, any on-path attacker can sniff a query and try to beat the legitimate server by quickly crafting a spoofed answer
Why is the query ID not enough?

- Attack #1: *sniffing*+*spoofing*

- As no encryption is applied to DNS messages, any on-path attacker can sniff a query and try to beat the legitimate server by quickly crafting a spoofed answer.
Why is the query ID not enough?

- Attack #1: **sniffing+spoofing**

- As no encryption is applied to DNS messages, any on-path attacker can sniff a query and try to beat the legitimate server by quickly crafting a spoofed answer.
Why is the query ID not enough?

- Attack #1: **sniffing+spoofing**

As no encryption is applied to DNS messages, any on-path attacker can sniff a query and try to beat the legitimate server by quickly crafting a spoofed answer.
Why is the query ID not enough?

- Attack #1: **sniffing+spoofing**

- As no encryption is applied to DNS messages, any on-path attacker can sniff a query and try to beat the legitimate server by quickly crafting a spoofed answer.
Why is the query ID not enough? - II

- What if the attacker can’t observe the query?

- Attack #2: fragmentation

- DNS messages can be larger than a typical MTU (maximal transmission unit)

- If that is the case, they cause IP fragmentation (i.e. the original IP packet is broken into multiple ones)

- Challenge information (UDP port + query ID) is carried in the first fragment

- **Attack**: force DNS resolver to send requests that typically elicit a large answer; inject packet fragments that are likely to override the actual 2nd fragment of the response
Why is the query ID not enough? - III

- Attack #3: prediction

- Historically, certain DNS resolver implementations used easily predictable source ports and query IDs

- E.g. Bind 9 uses a weak PRNG (LFSR generator): when LSB of transaction ID is 0, there are only 10 possible values for the ID which is going to be generated for the next query

- Enables spoofing DNS replies by sending a burst of fake replies with guessed UDP port/query IDs
DNS cache poisoning

- All attacks above are used to inject fake records into DNS caches
- Every client which request those records will receive poisoned data
- What is the impact of this?
DNS cache poisoning - II
DNS cache poisoning - II

- Can redirect a client to the wrong IP address for a certain domain
DNS cache poisoning - II

- Can redirect a client to the wrong IP address for a certain domain

- Not so useful if the user expects HTTPS (which is now common)
DNS cache poisoning - II

- Can redirect a client to the wrong IP address for a certain domain
  - Not so useful if the user expects HTTPS (which is now common)
- Can redirect a client to a different domain using poisoned CNAME records
DNS cache poisoning - II

- Can redirect a client to the wrong IP address for a certain domain
  - Not so useful if the user expects HTTPS (which is now common)
- Can redirect a client to a different domain using poisoned CNAME records
  - Works as long as the user does not realize the domain she reached is not the one she requested
Internet-wide study of DNS Cache Injections
Viability of cache poisoning attacks

• We have seen broadly how poisoned DNS replies can be successfully submitted

• We have seen the possible consequences of successful DNS cache poisoning

• A relevant question is: if a poisoned DNS reply is successfully submitted, will DNS software accept it?

• This is what the assigned reading focuses on
Discussion of the paper

- Overall idea: establish whether popular DNS caches accept and use poisoned DNS replies

- **Observation #1:** whether replies are accepted or not may depend on the configuration of the cache software

- **Observation #2:** even when poisoned records are accepted, they may not be used due to *ranking*
Experimental methodology
Experimental methodology

Direct prober: used when target cache is directly reachable
Experimental methodology

Indirect probers: used when access to target cache is restricted

Direct prober: used when target cache is directly reachable
Experimental methodology

Indirect probers: used when access to target cache is restricted

Direct prober: used when target cache is directly reachable

Two sets of DNS servers: (i) legitimate authoritative servers, (ii) attack servers (sets are on network blocks w/ different IP prefixes)
Experimental methodology - II

• Phases:
  
  • 1: **Seeding**: creation of genuine record in target cache
  
  • 2: **Poisoning**: attempt to inject poisoned records in target cache
  
  • 3: **Verification**: check if records of interest contain genuine or injected values
Attack payload example

<table>
<thead>
<tr>
<th>16. ak1</th>
<th>Q An</th>
<th>Poisoned (spoofed) answer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A? zweil.test-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>zweil.test-ak1.TAIL CNAME one1.test-ak1.TAIL;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11bis.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-w11.TAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one1.test-ak1.TAIL CNAME one1.magic-ak1.TAIL</td>
</tr>
</tbody>
</table>

Record planted during seeding phase
Attack payload example

16. ak1

Q An Au/Ad

A? zweil.test-ak1.TAIL
zweil.test-ak1.TAIL CNAME one1.test-ak1.TAIL;
onel.test-ak1.TAIL CNAME one1.magic-ak1.TAIL

one1.test-ak1.TAIL

Record planted during seeding phase

Poisoned (spoofed) answer

• Broadly, this attack spoofs a CNAME record redirecting the target test-ak1 subdomain to the attack subdomain magic-ak1

• The record is not used until a valid A record for one1.test-ak1.TAIL exists; once that record expires, the poisoned CNAME record becomes active
### Results

- **Vulnerabilities of popular DNS cache softwares**

<table>
<thead>
<tr>
<th>Name</th>
<th>BIND 9.10.2-P2</th>
<th>BIND 9.4.1</th>
<th>Unbound 1.5.4</th>
<th>MaraNDS 3.2.07 Deadwood</th>
<th>PowerDNS 3.7.3</th>
<th>MS DNS 6.1 Win Server’08 R2 6.1.7601</th>
<th>MS DNS 6.2 Win Server’12 R2 6.2.9200</th>
<th>MS DNS 6.3 Win Server’12 R2 6.3.9600</th>
<th>Google Public DNS</th>
<th>Open DNS</th>
<th>BIND 9.10.2-P2 w/DNSSEC</th>
<th>Nominum Vantio CacheServe v5</th>
<th>Nominum Vantio CacheServe v7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ns0</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2 ns0-auth</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3 ns</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4 ns-auth</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>5 ns2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>6 ns2-auth</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>7 b4</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>8 u1-auth</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>9 u3-2</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>10 u3-3</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>11 u3-4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>12 w-dname</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>13 w7</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>14 w8</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>15 dname</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>16 ak1</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>17 w11</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>18 w11bis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Results - II

- Percentage of vulnerable networks per category

![Graph showing the percentage of vulnerable networks per category for different network categories: Popular Networks, Enterprise Networks, ISP Networks. The x-axis represents different tests, and the y-axis represents the percentage.]
Can we do anything about DNS security?

- We have seen that:
  - Poisoned DNS replies can be successfully submitted
  - DNS caches oftentimes accept and use such requests
  - DNS poisoning can have real and damaging consequences for users

- It’s 2017: can we come up with a more secure way to do name resolutions?
DNSSEC

• Goal: secure DNS communications to prevent spoofing

• Broad design principles:
  
  • Provide **data integrity** (responses cannot be altered in transit)

  • Provide **proof of authority** (the response come from an authoritative server for the domain of interest)
Securing DNS record with DNSSEC

- Example (from Microsoft Technet): DNS records for “contoso.com” domain before and after enabling DNSSEC:
Securing DNS record with DNSSEC

- Example (from Microsoft Technet): DNS records for “contoso.com" domain before and after enabling DNSSEC:

Every record has now been signed
Securing DNS record with DNSSEC

- Example (from Microsoft Technet): DNS records for “contoso.com” domain before and after enabling DNSSEC:

  - Every record has now been signed
  - A DNSKEY record (and other related data) has been created
Validation process in a nutshell

- Example (from Microsoft Technet):
Validation process in a nutshell

- Example (from Microsoft Technet):

When replying to a DNS query, an authoritative server sends both the requested record and a signature record.
Validation process in a nutshell

- Example (from Microsoft Technet):

When replying to a DNS query, an authoritative server sends both the requested record and a signature record. The DNS cache validates the record against the signature before forwarding the reply to the user.
Roots of trust
Roots of trust

• How do we know we can trust DNSKEY records?
  
  • DNSKEY records are signed themselves with a key-signing key (KSK)
    
    • How do we trust KSK records?
      
      • Hash of the KSK is provided to the authoritative nameserver for the parent zone (requester can use the hash to validate the KSK)
        
        • Root of trust goes all the way to the root DNS servers, and…
          
          • … DNSKEY used to sign KSK of these servers is signed in a public ceremony
Key-signing ceremony

Source: https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/
DNSSEC issues

• Broken chain of trust may prevent verification (nearly 20% of popular domain, Dai et al., CANS 2016)

  • Wrong or missing records in parent zones

  • Hash in parent zone does not match

• DNSSEC records are large and can be used in amplification denial-of-service attacks (not really an issue of DNSSEC, but an issue created by DNSSEC)