The Domain Name System

3035/GZ01 Networked Systems
Kyle Jamieson
Lecture 10

Department of Computer Science
University College London
Today

1. The Domain Name System (DNS)
2. DNS security: Cache poisoning attacks of ’08
3. Coursework 2 Intro: Ben’s local DNS server
The Domain Name System (DNS)

**People** have many identifiers
- Name, passport number, mailing address

**Internet hosts and routers**
1. IP address (32 bit), used for addressing datagrams
2. “Name” (e.g. www.yahoo.com) used by humans

**Question:** *how to map between IP address and name?*

\[ IP = \text{DNS}_\text{RESOLVE}(\text{domain\_name}) \]
Naming in computer systems

- **Flat namespace**: no context in name
  - microsoft, ucl, google

- **Path name**: includes context in name
  - ginger.pedantic.edu.
  - /usr/bin/vi

K&R, Chapter 2, Section B, second paragraph
DNS names are resolved recursively

Absolute path name: /usr/bin/vi
1. name = usr, context = / (root)
2. name = bin, context = /usr
3. name = vi, context = /usr/bin

Absolute DNS name: sipb.mit.edu.
1. name = edu, context = . (root)
2. name = mit, context = .edu
3. name = sipb, context = mit.edu.
Would-be design #1: the directory

\[ IP = \text{DNS\_RESOLVE}(\text{domain\_name}) \]

ftp hosts.txt user1.isp.net
ftp hosts.txt user2.isp.net

hosts.txt
web.mit.edu: 18.12.1.12
lxr.linux.no: 23.2.33.12
d.ja.net: 198.22.11.1
Would-be design #2: centralized server

\[ IP = \text{DNS\_RESOLVE}(\text{domain\_name}) \]

Could this work?
The Domain Name System

- **A database** implemented by many *name servers (NS)*
  - Distributed
  - Replicated
  - Hierarchical

- Delegation follows namespace hierarchy

```
. com. uk. edu.
  ac.uk.
 ucl.ac.uk.
  cmu.edu.
  ee.ucl.ac.uk.
  cs.ucl.ac.uk.
```
The DNS namespace is hierarchical.

Root:

Top-level domain (TLD):
- com.
- uk.
- edu.

- ac.uk.
- cmu.edu.
- mit.edu.

- ucl.ac.uk.

- ee.ucl.ac.uk.
- cs.ucl.ac.uk.

www.cs.ucl.ac.uk.
DNS resource record (RR): Overview

DNS is a distributed database storing **resource records**

RR includes: *(name, type, value, time-to-live)*

- **Type = A** (address)
  - name is hostname
  - value is IP address
- **Type = NS** (name server)
  - name is domain (e.g. cs.ucl.ac.uk)
  - value is hostname of authoritative name server for this domain
- **Type = CNAME**
  - name is an alias for some “canonical” (real) name
  - e.g. www.cs.ucl.ac.uk is really haig.cs.ucl.ac.uk
  - value is canonical name
- **Type = MX** (mail exchange)
  - value is name of mail server associated with domain name
  - pref field discriminates between multiple MX records
Many uses of DNS

• Hostname to IP address translation
• IP address to hostname translation ("reverse lookup")
• Host name aliasing
  – Canonical hostname, alias host names point to canonical
  – Can be arbitrarily many aliases
• Mail server location
  – Lower pref fields are preferred
  – Handles mail server primary-backup arrangement
• Content distribution networks
  – Load balancing among servers with different IP addresses
  – Complex, hierarchical arrangements are possible
Root name servers

• 13 root name servers worldwide
• Responsible for the root of the DNS namespace
• Well-known to all NS’s:

A.ROOT-SERVERS.NET. IN A 198.41.0.4
B.ROOT-SERVERS.NET. IN A 192.228.79.201
C.ROOT-SERVERS.NET. IN A 192.33.4.12
...
M.ROOT-SERVERS.NET. IN A 202.12.27.33

- a Verisign, Dulles, VA
- b USC-ISI Marina del Rey, CA
- c Cogent, Herndon, VA (also LA)
- d U Maryland College Park, MD
- e NASA Mt View, CA
- f Internet Software C. Palo Alto, CA (and 36 other locations)
- g US DoD Vienna, VA
- h ARL Aberdeen, MD
- i Verisign (21 locations)
- j Autonomica, Stockholm (plus 28 other locations)
- k RIPE London (also 16 other locations)
- l WIDE Tokyo (also Seoul, Paris, SF)
- m ICANN Los Angeles, CA
- n ICANN Los Angeles, CA
TLD and Authoritative Servers

• Top-level domain (TLD) servers
  – Responsible for com, org, net, edu, etc, and all top-level country domains: uk, fr, ca, jp
  – *Network Solutions* maintains servers for com TLD
  – *Educause* for edu TLD

• Authoritative DNS servers
  – An organization’s DNS servers, providing authoritative information for organization’s servers
  – Can be maintained by organization or service provider
Local name servers

- Do not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one
  - Also called default or caching name server

- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Does work for the client
DNS in operation

- Most queries and responses are UDP datagrams
- Two types of queries:
  - **Recursive:**
  - **Iterative:**
A recursive DNS lookup (simplified)

. (root) authority 198.41.0.4
  
edu.: NS 192.5.6.30
  	no.: NS 158.38.8.133
  
uk.: NS 156.154.100.3

Client
  
  . (root): NS 198.41.0.4
  
edu.: NS 192.5.6.30
  
scholarly.edu.: NS 12.35.1.1

www.scholarly.edu?

Contact 192.5.6.30 for edu.

www.scholarly.edu?

Contact 12.35.1.1 for scholarly.edu.

www.scholarly.edu?

www.scholarly.edu?:
  
  A 12.35.2.30
  
imap.scholarly.edu.: A 12.35.2.31

edu. authority 192.5.6.30
  
scholarly.edu.: NS 12.35.1.1
  
pedanNc.edu.: NS 19.31.1.1

Contact 12.35.1.1 for scholarly.edu.

scholarly.edu. authority 12.35.1.1
  
www.scholarly.edu.: A 12.35.51.30
Local NS does clients’ work

1. Clients make **recursive** queries to local NS
2. Local NS processing:
   – Local NS sends **iterative** queries to other NS’s
   – or, finds answer in cache
3. Local NS responds with an answer to the client’s request
## Recursive versus iterative queries

<table>
<thead>
<tr>
<th>Recursive query</th>
<th>Iterative query</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Less burden on client</td>
<td>• More burden on client</td>
</tr>
<tr>
<td>• More burden on server (has to resolve the query)</td>
<td>• Less burden on server (refers to another)</td>
</tr>
<tr>
<td>• Most root and TLD servers will not answer (reduce load)</td>
<td></td>
</tr>
<tr>
<td>• Local NS answers recursive query</td>
<td></td>
</tr>
</tbody>
</table>
A recursive query “in the wild”

```bash
$ dig @a.root-servers.net www.freebsd.org +norecurse
;; ;<<><> DiG 9.4.3-P3 <<><> @a.root-servers.net www.freebsd.org
+norecurse
;; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57494
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 12

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; AUTHORITY SECTION:
org. 172800 IN NS b0.org.afilias-nst.org.
org. 172800 IN NS d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
#b0.org.afilias-nst.org. 172800 IN A 199.19.54.1
#d0.org.afilias-nst.org. 172800 IN A 199.19.57.1

;; Query time: 177 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; MSG SIZE  rcvd: 435
```

“Glue” record
A recursive query “in the wild” (2)

```
$ dig @199.19.54.1 www.freebsd.org +norecurse
; <<>> DiG 9.4.3-P3 <<>> @a0.org.afilias-nst.org www.freebsd.org
+norecurse
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39912
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 0

;; QUESTION SECTION:
www.freebsd.org. IN A

;; AUTHORITY SECTION:
freebsd.org. 86400 IN NS ns1.isc-sns.net.
freebsd.org. 86400 IN NS ns2.isc-sns.com.
freebsd.org. 86400 IN NS ns3.isc-sns.info.

;; Query time: 128 msec
;; SERVER: 199.19.56.1#53(199.19.56.1)
;; MSG SIZE  rcvd: 121
```

- No glue record provided for ns1.isc-sns.net, so need to go off and resolve, then restart the query
A recursive query “in the wild” (3)

```
$ dig @ns1.isc-sns.net www.freebsd.org +norecurse
; <<>> DiG 9.4.3-P3 <<>> @ns1.isc-sns.net www.freebsd.org +norecurse
;(1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17037
;; flags: qr aa QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;www.freebsd.org. IN A

;; ANSWER SECTION:
www.freebsd.org. 3600 IN A 69.147.83.33

;; AUTHORITY SECTION:
freebsd.org. 3600 IN NS ns2.isc-sns.com.
freebsd.org. 3600 IN NS ns1.isc-sns.net.
freebsd.org. 3600 IN NS ns3.isc-sns.info.

;; ADDITIONAL SECTION:
ns1.isc-sns.net. 3600 IN A 72.52.71.1
ns2.isc-sns.com. 3600 IN A 38.103.2.1
ns3.isc-sns.info. 3600 IN A 63.243.194.1
```
DNS: caching and updating records

• Once any name server learns mapping, it **caches** mapping
  – Subsequent requests are served from the cache
  – Cached entries timeout after some time (TTL field)
  – TLD servers typically cached in local name servers, thus root name servers are not often visited
• Update/notify mechanisms under design by IETF
• Caching interacts with load balancing
• Caching has security implications, as we will see later
Inserting records into DNS

• Example: new startup “Network Utopia”
• Register name networkutopia.com at DNS registrar (e.g., Network Solutions)
  – Provide names, IP addresses of authoritative name server (primary and secondary)
  – Registrar inserts two RRs into .com TLD server:

  A.GTLD-SERVERS.NET
  (networkutopia.com, dns1.networkutopia.com, NS)
  (dns1.networkutopia.com, 212.212.212.1, A)

• Network Utopia inserts authoritative A, MX records:

  dns1.networkutopia.com
  (networkutopia.com, mail.networkutopia.com, MX)
  (www.networkutopia.com, 212.212.212.3, A)
• Most queries and responses via UDP, server port 53
Local NS at least needs to keep state associating Query ID → which query (if any)
**DNS resource record (RR): Detail**

- **rdata**: data associated with the RR
- **type**: determines the meaning of rdata
- **class**: always IN (Internet)
DNS protocol message

- Message formats completely specified in RFC 1035
- Query and reply messages have identical format
- **Question section**: query for name server
- **Answer section**: RRs answering the question
- **Authority section**: RRs that point to an authoritative NS
- **Additional section**: “glue” RRs
DNS protocol header

- **Query ID**: 16-bit identifier shared between query, reply
- **Flags word**
  - QR: query (0) or response (1)
  - opcode: standard query (0)
  - AA: authoritative answer
  - TC: truncation
  - RD: Recursion desired
  - RA: Recursion available
  - Z: (reserved and zeroed)
  - rcode: response code; ok (0)
- **qdcound**: number of question entries (QEs) in message
- **ancount**: number of RRs in the answer section
- **nscound**: number of RRs in the authority section
- **arcount**: number of RRs in the additional section
Today

1. The Domain Name System (DNS)
2. DNS security: Cache poisoning attacks of ’08
3. Coursework 2 Intro: Ben’s local DNS server
DNS security

• Why do you trust your web browser?

• Summer 2008: Kaminsky DNS vulnerability

Vulnerability Note VU#800113

Multiple DNS implementations vulnerable to cache poisoning

Overview

Deficiencies in the DNS protocol and common DNS implementations facilitate DNS cache poisoning attacks.
2005 DNS cache poisoning: Impact

• 500-1000 medium-to-large size organizations were affected
• 10 large (1000+) person organizations
• Affected domain names:
  – americanexpress.com
  – citicards.com
  – billpay.quickbooks.com
  – adp.com
  – hrblockemail.com
  – orbitz.com
  – sabre.com
  – tickets.com

[Source: Internet Storm Center]
Implications of subverting DNS

1. Redirect victim’s web traffic to rogue servers
2. Redirect victim’s email to rogue email servers (MX records in DNS)
   • Does Secure Sockets Layer (SSL) provide protection?
     – Yes and No
     – Yes—user will get “wrong certificate warnings” if SSL is enabled
     – No—SSL not enabled or user ignores warnings
     – No—how is SSL trust established? By email!
Predicting the next query ID

- Next query ID = Query ID + 1

<table>
<thead>
<tr>
<th>Source port</th>
<th>Dest port</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP length</td>
<td>UDP checksum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query ID</th>
<th>Opcode</th>
<th>RR</th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>A</th>
<th>Z</th>
<th>rcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DNS cache poisoning, version 1

Clients -> Victim NS

IP = Bad guy's IP, QID = 5001

Victim NS -> Root NS
Root NS -> TLD NS

IP = Bad guy's IP, QID = 5001

TLD NS -> Barclay's NS

IP = Barclay's IP, QID = 5001

Barclay's NS -> www.barclays.co.uk?

www.barclays.co.uk? -> Clients

IP = Bad guy's IP, QID = 5001

IP = Bad guy's IP, QID = 5002

IP = Bad guy's IP, QID = 5003

...
DNS cache poisoning, version 1

• Requirements for successful exploit
  – Attacker has to know the UDP source port the victim NS sent the query on (otherwise UDP drops the forged reply)
    • ca. 2008, NS’s used a well-known source port!
  – Attacker has to guess the Query ID
    • Countermeasure: name servers now use random query IDs
    • Older servers used an easily-guessable pseudorandom number generator
  – Attacker’s forged replies have to arrive first
  – Name can’t already be in victim’s cache
Kaminsky DNS cache poisoning step
Attacking authority records one level higher in the DNS hierarchy

barclays.co.uk NS ns1.barclays.co.uk
ns1.barclays.co.uk A 10.0.0.1

ns.co.uk

ns1.barclays.co.uk
10.0.0.1

www123.barclays.co.uk?
QID = 3817

ns1.barclays.co.uk
10.2.0.2

www123.barclays.co.uk?
QID = 39183

ns1.badguy.com
10.2.0.2

Bad guy’s network

Clients
Kaminsky DNS cache poisoning

• Compromise the **entire domain** instead of just an IP
• Repeat the step $K$ times to make success more likely
• Use fresh uncached query each time
  – www123.barclays.co.uk
  – www1234.barclays.co.uk
  – www12345.barclays.co.uk

• How likely is the attack to work?

  \[
  \Pr(\text{guess correct query id}) = \frac{1}{65,535}
  \]

  \[
  \Pr(\text{guess wrong query id } K \text{ times}) = \left(1 - \frac{1}{65,535}\right)^K
  \]

<table>
<thead>
<tr>
<th>$K$</th>
<th>$\Pr(\text{Wrong } K \text{ times})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.99994</td>
</tr>
<tr>
<td>40</td>
<td>0.9994</td>
</tr>
<tr>
<td>400</td>
<td>0.994</td>
</tr>
<tr>
<td>4,000</td>
<td>0.94</td>
</tr>
<tr>
<td>40,000</td>
<td>0.54</td>
</tr>
</tbody>
</table>
Cache poisoning: increasing the odds

Client

Root NS

Victim NS

ns.co.uk
	ns1.barclays.co.uk A 10.0.0.1

barclays.co.uk NS ns1.barclays.co.uk
ns1.barclays.co.uk A 10.0.0.1

www123.barclays.co.uk?
QID = 3817
www1234.barclays.co.uk?
QID = 48917

ns1.barclays.co.uk
10.0.0.1

Bad guy’s network
ns1.badguy.com
10.2.0.2

www123.barclays.co.uk?
QID = 39183

www1234.barclays.co.uk?
QID = 715
Cache poisoning: increasing the odds

- Send a flurry of $L$ queries and $L$ forged responses
  - Random query IDs everywhere

\[
\Pr(\text{one query/response pair matches}) = \frac{1}{65,535}
\]

\[
\Pr(\text{guess wrong query id } L \text{ times}) = \left(1 - \frac{1}{65,535}\right)^\binom{L}{2}
\]

\[
= \left(1 - \frac{1}{65,535}\right)^{\frac{L(L-1)}{2}}
\]

- In practice, takes about 10 minutes

<table>
<thead>
<tr>
<th>$L$</th>
<th>$\Pr(\text{Every forgery wrong})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.9994</td>
</tr>
<tr>
<td>100</td>
<td>0.926</td>
</tr>
<tr>
<td>290</td>
<td>0.54</td>
</tr>
</tbody>
</table>
Mitigating DNS cache poisoning

• Randomize UDP source port of local NS query
• Reply checking:
  1. Kernel network stack matches destination port of TLD server’s reply with UDP source port of local NS’s query
  2. DNS server matches query ID of reply with query id of request
• MS DNS server pre-allocates 2,500 UDP ports for requests

$$\Pr(\text{correct guess}) = \left( \frac{1}{65,000} \right) \left( \frac{1}{2,500} \right) \approx 6 \times 10^{-9}$$
Today

1. The Domain Name System (DNS)
2. DNS security: Cache poisoning attacks of ’08
3. Coursework 2 Intro: Ben’s local DNS server
Ben’s local name server: Structure

- Single-threaded, only one client active at a time
- Two sockets: one for incoming recursive queries (ss); the other for outgoing iterative queries (cs)
17th November: Midterm exam

NEXT TIME