Computer Networks II
Domain Name System
Domain Name System

• The domain name system (DNS) is an application-layer protocol for mapping domain names to IP addresses.
Domain Name System

- DNS provides a distributed database over the internet that stores various resource records, including:
  - **Address (A) record**: IP address associated with a host name
  - **Mail exchange (MX) record**: mail server of a domain
  - **Name server (NS) record**: authoritative server for a domain

Example DNS entries from http://www.maradns.org/tutorial/recordtypes.html

```
For example, if example.com wishes to sub-delegate "john.example.com" to John who works at Example, Inc., lines like this can be added to the example.com zone file:

john.example.com. NS nsl.john.example.com.
john.example.com. NS nss.john.example.com.
# It's important to provide "glue"; in other words, let the world know
# the IPs for these name servers.
nsl.john.example.com. 10.9.8.7
nss.john.example.com. 10.5.77.65

John, who is running his own nameservers with the IPs 10.9.8.7 and 10.5.77.65 then has a zone file for john.example.com that looks something like this

# It is best if the NS records for a subzone agree with the delegation
# records above
john.example.com. NS nsl.john.example.com.
john.example.com. NS nss.john.example.com.

nsl.john.example.com. 10.9.8.7
nss.john.example.com. 10.5.77.65

# Now that that is out of the way, here is the rest of the zone
john.example.com. 10.9.8.7
www.john.example.com. 10.5.77.65
john.example.com. MX 10 mail.john.example.com.
mail.john.example.com. 10.9.8.7
```

Example DNS entries from http://www.maradns.org/tutorial/recordtypes.html
Name Servers

• Domain names:
  – Two or more labels, separated by dots (e.g., cs166.net)
  – Rightmost label is the top-level domain (TLD)

• Hierarchy of authoritative name servers
  – Information about root domain
  – Information about its subdomains (A records) or references to other name servers (NS records)

• The authoritative name server hierarchy matches the domain hierarchy: root servers point to DNS servers for TLDs, etc.

• Root servers, and servers for TLDs change infrequently

• DNS servers refer to other DNS servers by name, not by IP: sometimes must bootstrap by providing an IP along with a name, called a glue record
Name Resolution

- **Zone**: collection of connected nodes with the same authoritative DNS server
- Resolution method when answer not in cache:

  - Where is www.example.com?
  - Try com nameserver
    - Where is www.example.com?
    - Try example.com nameserver
      - Where is www.example.com?
  - 208.77.188.166

Client → ISP DNS Server → Root DNS Server → Com DNS Server → Example.com DNS Server
DNS packet structure

- DNS queries and replies are transmitted via a single UDP packet

**Query**
- 16-bit query identifier
- Domain name and type of record

**Response**
- NAME
- Type
- Class
- TTL
- RDLENGTH
- RDATA
- A
- NS
- MX

- DNS response consist of a sequence of DNS records

- DNS queries are typically issued over UDP on port 53
  - 16-bit request identifier in payload
DNS Caching

• There would be too much network traffic if a path in the DNS tree would be traversed for each query
  – Root zone would be rapidly overloaded
• **DNS servers** cache results for a specified amount of time
  – Specified by ANS reply's time-to-live field
• **Operating systems and browsers** also maintain resolvers and DNS caches
  – View in Windows with command `ipconfig /displaydns`
  – Associated privacy issues
DNS Caching

Step 1: query yourdomain.org

Step 2: receive reply and cache at local NS and host
DNS Caching (con'd)

Step 3: use cached results rather than querying the ANS

Local Machine 1
- Application
- Resolver
- cache

Local Machine 2
- Application
- Resolver
- cache

Local NS
- Resolver
- cache

query
answer

Step 4: Evict cache entries upon ttl expiration
Recursive Name Resolution

Local Machine
- Application
- Resolver
- cache

ISP Server
- Resolver
- cache

root name server
- Resolver
- cache

query → answer
referral
query
answer
Iterative Name Resolution

Local Name Server

- Application
- Resolver
- cache

Resolver

- cache
- google.com
- query
- answer

- (root)
- Resolver
- cache
- query
- answer

- .com
- Resolver
- cache
- query
- answer

- answer
- query
- 2

- answer
- query
- 3
Pharming: DNS Hijacking

- Changing IP associated with a server maliciously:

  ![Diagram showing Normal DNS, IP address 74.208.31.63, My Premium Blog Spot with UserID and Password, IP address 208.77.188.166, and Pharming attack with IP address 74.208.31.63.]

**Phishing:** the different web sites look the same.
DNS Cache Poisoning

(a) Evil Client

ISP DNS Server

DNS Request
Where is www.example.com?

DNS Cache

DNS Request
Where is www.example.com? Query ID = x

DNS Lookup

(b) Evil Client

ISP DNS Server

DNS Reply
www.example.com is 1.1.1.1 Query ID = x

DNS Cache

www.example.com 1.1.1.1

(c) Victim Client

ISP DNS Server

DNS Request
Where is www.example.com?

DNS Reply
www.example.com is 1.1.1.1

DNS Cache

www.example.com 1.1.1.1
DNS Cache Poisoning

- Basic idea: give DNS servers false records and get it cached
- DNS uses a 16-bit request identifier to pair queries with answers
- Cache may be poisoned when a name server:
  - Disregards identifiers
  - Has predictable ids
  - Accepts unsolicited DNS records
DNS Cache Poisoning and the Birthday Paradox

• Attacker issuing fake response will guess a transaction ID equal to one of \( n \) different 16-bit real ID with a probability \( n/2^{16} \)

• The probability of an attacker fail to match one will be \( (1-n/2^{16}) \).

• With \( n \) fake responses the probability of failure will be \( (1-n/2^{16})^n \)
DNS Cache Poisoning and the Birthday Paradox

Figure 6.7: A DNS cache poisoning attack based on the birthday paradox: (a) First, an attacker sends $n$ DNS requests for the domain she wishes to poison. (b) The attacker sends $n$ corresponding replies for her own request. If she successfully guesses one of the random query IDs chosen by the ISP DNS server, the response will be cached.
Subdomain DNS Cache Poisoning

Attacker

- aaac.example.com
- bbb.example.com
- aaa.example.com

Queries

Local Name Server

- Application
- Resolver
- cache

Glue record of example.com

Responses

Example.com

- Resolver
- cache

.query

1

answer

Resolver

.cache

.com

.query

2

answer

Resolver

.cache

.(root)

.query

3

answer

Resolver

.cache
Figure 6.8: A DNS cache poisoning attack against a client: (a) On visiting a malicious web site, the victim views a page containing many images, each causing a separate DNS request to be made to a nonexistent subdomain of the domain that is to be poisoned. (b) The malicious web server sends guessed responses to each of these requests. On a successful guess, the client’s DNS cache will be poisoned.
DNS Cache Poisoning Prevention

• **Use random identifiers** for queries
• Always check identifiers
• **Port randomization** for DNS requests
• **Deploy DNSSEC**
  – Challenging because it is still being deployed and requires reciprocity
DNSSEC

• Guarantees:
  – Authenticity of DNS answer origin
  – Integrity of reply
  – Authenticity of denial of existence

• Accomplishes this by signing DNS replies at each step of the way

• Uses public-key cryptography to sign responses

• Typically use trust anchors, entries in the OS to bootstrap the process
Figure 6.9: A DNSSEC response and the chain of trust that validates it. In this case, book.example.com returns a signed DNS response along with its public key, example.com sends its public key and a signed DS record validating the public key of book.example.com, and .com sends its public key and a signed DS record validating the public key of example.com. The client can trust this chain, since it knows the public key of .com.
DNSSEC Deployment

- As the internet becomes regarded as critical infrastructure there is a push to secure DNS
- NIST is in the process of deploying it on root servers now
- May add considerable load to DNS servers with packet sizes considerably larger than 512 byte size of UDP packets
- There are political concerns with the US controlling the root level of DNS