The Domain Name System
“Words have meaning and names have power.”

– Network identities are integral to authentication and authorization
– On the Internet, names are IP addresses
– Raw addresses are not always the right abstraction
– Alternatively, we use *domain names*
Domain Names

- Domain names map memorable labels to IP addresses
- Can map one name to many addresses
- How are names resolved?
  - Static mappings (e.g., /etc/hosts)
  - NIS, LDAP, AD
  - The DNS
The Domain Name System [1]

- The domain name system (DNS) is a distributed, fault-tolerant database
- Composed of hierarchical administrative zones starting from the root (.), then gTLDs/ccTLDs
- Each zone is responsible for some namespace
- Zones can delegate to sub-zones
Name Servers

- Each zone has at least one *authoritative name server* that is the trusted source for that zone’s DNS information.
- A zone’s DNS information is comprised of *resource records* (RRs) that maps from a key to a value:
  - **A** IPv4 address
  - **AAAA** IPv6 address
  - **CNAME** Canonical name
  - **MX** Mail exchanger
  - **NS** Authoritative name server
Address Resolution

- Address resolution is performed by walking the DNS hierarchy until an authoritative response is received.
- *Recursive resolvers* perform queries on a *stub resolver’s* behalf.
- A resolver can also directly walk the hierarchy itself.
- *Caching resolvers* temporarily store responses to reduce network load according to an RR time-to-live (TTL).
- Clients and servers can assume multiple logical roles.
Recursive Resolution

- Resolver
- Root NS
- TLD NS
- Auth NS
Stub Resolution

Stub ─── Recursor ─── Root NS
            │              └── TLD NS
            │              └── Auth NS
# drill -T seclab.cs.ucsb.edu

. 518400 IN NS a.root-servers.net.
.

[...]

. 518400 IN NS m.root-servers.net.
edu. 172800 IN NS a.edu-servers.net.
edu. 172800 IN NS b.edu-servers.net.

[...]
edu. 172800 IN NS m.edu-servers.net.
ucsb.edu. 172800 IN NS ns1.ucsb.edu.
ucsb.edu. 172800 IN NS ns2.ucsb.edu.
ucsb.edu. 172800 IN NS bru-ns2.brown.edu.
seclab.cs.ucsb.edu. 1200 IN A 128.111.48.75
**DNS Messages**

```
struct DnsHeader {
    identifier: u16,
    query_or_response: bool,
    opcode: u4,
    authoritative_answer: bool,
    response_truncated: bool,
    recursion_desired: bool,
    recursion_available: bool,
    question_count: u16,
    answer_count: u16,
    authority_count: u16,
    additional_count: u16,
}
```

- **identifier** used to match queries, responses; also called Transaction ID (TX-ID)
- **opcode** is the request type
- **authoritative_answer** indicates whether response was cached
- Header followed by question, answer vectors
Resource Records

```c
struct ResourceRecord {
    name: String,
    rr_type: u16,
    rr_class: u16,
    ttl: u32,
    rd_len: u16,
    rd_data: [u8; rd_len],
}
```

- `rr_type` denotes the kind of record (e.g., A, NS)
- `rr_class` denotes the mapping scope (e.g., IN for Internet)
- `ttl` denoted in seconds
- RR-specific data given in `rd_data`
DNS Security Issues

- DNS on its own does not provide confidentiality, integrity, or authenticity in transit
- Lack of security opens the door to DNS hijacking
- Attack where an adversary claims to own a domain when in actuality they do not
- Caching behavior can greatly amplify the effects of an attack
DNS Hijacking

Figure 1: DNS hijacking example using UDP spoofing
Figure 2: DNS cache poisoning example
DNS Cache Poisoning

DNS cache poisoning is devastating if successful, but it is difficult to pull off.

1. Client must have requested the target name
2. Addressing and DNS TX-ID must match (Probability?)
3. Any authority and additional records must match the queried domain (Why?)
4. Target name must not be cached
5. Attacker must win the race
DNS Cache Poisoning

DNS cache poisoning is devastating if successful, but it is difficult to pull off
(1) Client must have requested the target name
(2) Addressing and DNS TX-ID must match (Probability?)
(3) Any authority and additional records must match the queried domain (Why?)
(4) Target name must not be cached
(5) Attacker must win the race
Can this basic attack be improved?
The Kaminsky Attack

(1) *Client must have requested the target name*
(2) Addressing and DNS TX-ID must match (Probability?)
(3) *Any authority and additional records must match the queried domain (Why?)*
(4) *Target name must not be cached*
(5) Attacker must win the race

Is such a thing possible?
Removing Attack Constraints

Client must have requested the target name
Removing Attack Constraints

Client must have requested the target name
What if the attacker requests names directly from the victim nameserver?
Removing Attack Constraints

Target name must not be cached
Target name must not be cached
If the attacker directly requests names, it can use fresh names for each query.
Any authority and additional records must match the queried domain
Removing Attack Constraints

Any authority and additional records must match the queried domain

In the response to a query, an authoritative nameserver can update the NS entry for the domain.

This can be exploited by an attacker to poison the NS entry and take over the entire domain (zone)!
The Kaminsky Attack

```
 ┏━━━━━━━━━━┓ ┏━━━━━━━━━━┓
┃ Root/GTLD NS ┃ ┃ Auth NS ┃
┗━━━━━━━━━━┛ ┗━━━━━━━━━━┛

 ┏━━━━━━━━━━┓ ┏━━━━━━━━━┓
┃ Resolver ┃ ┃ Evil Cl. ┃
┗━━━━━━━━━━┛ ┗━━━━━━━━━┛

 ┏━━━━━━━━━━┓ ┏━━━━━━━━━━━━━━┓
┃ Evil Auth NS ┃ ┃ Evil Auth NS ┃
┗━━━━━━━━━━┛ ┗━━━━━━━━━━━━━━┛
```
DNS Hijacking Defenses

- The attacker needs to predict several pieces of information
  (i) Queried domain (non-Kaminsky attack)
  (ii) DNS query ID (TX-ID)
  (iii) Network ports
- Does increasing the entropy of the system work?
- DNSSEC, DNSCurve, DNSCrypt
DNSSEC specifies DNS security extensions, adding several properties:

(i) Origin authentication
(ii) Data integrity
(iii) Authenticated denial of existence

DNSSEC does not provide:

(i) Data confidentiality
(ii) Service availability
DNSSEC Chain of Trust

- DNSSEC uses public-key cryptography to establish a *chain of trust* from the DNS root zone to authoritative nameservers.
- Root zone is a trusted third party and extremely powerful (thus, a point of political contention).
- DNSSEC-aware resolvers obtain root zone public keys (trust anchors) through an out-of-band channel.
DNSSEC Resource Records

**RRSIG** Resource record signature

**DNSKEY** Zone public key, used to verify RRSIG records

**DS** Delegation signer; denotes a zone → sub-zone delegation relation

**NSEC** Denotes next record name in a zone; used to verify record non-existence
DNSSEC Resolution

- Root NS
- Stub
- Recursor
- TLD NS
- Auth NS
chris:~$ cat ksk-as-ds.txt
  . IN DS 20326 8 2 E06D44B80B8F1D39A95C0B0D7C65D08458E880409BBC683457104237C7F8EC8D
chris:~$ drill -TD -k ksk-as-ds.txt www.cloudflare.com
;; Number of trusted keys: 1

[ ] . 172800 IN DNSKEY 256 3 8 ;{id = 48903 (zsk), size = 2048b}
  . 172800 IN DNSKEY 257 3 8 ;{id = 20326 (ksk), size = 2048b}
Checking if signing key is trusted:
New key: . 172800 IN DNSKEY 256 3 8 AwEAAac4qsciJ5MdMU...
  Trusted key: . 3600 IN DS 20326 8 2 e06d44b80b8f1d3...
  Trusted key: . 172800 IN DNSKEY 256 3 8 AwEAAac4qsciJ5MdMU...
Key is now trusted!
[ ] com. 86400 IN DS 30909 8 2 e2d3c916f6deeac73294e8268fb5885044a833fc54...
DNSSEC Example

;; Domain: com.
[T] com. 86400 IN DNSKEY 256 3 8 ;{id = 56311 (zsk), size = 1280b}
com. 86400 IN DNSKEY 256 3 8 ;{id = 39844 (zsk), size = 1280b}
com. 86400 IN DNSKEY 257 3 8 ;{id = 30909 (ksk), size = 2048b}
Checking if signing key is trusted:
New key: com. 86400 IN DNSKEY 256 3 8 AwEAAcpiO...
   Trusted key: . 3600 IN DS 20326 8 2 e06d44b80b8f1d3...
   Trusted key: . 172800 IN DNSKEY 256 3 8 AwEAAc4qsciJ5MdMU...
   Trusted key: com. 86400 IN DNSKEY 256 3 8 AwEAAcpiO...
Key is now trusted!
[T] cloudflare.com. 86400 IN DS 2371 13 2 32996839a6d808afe3eb4a795a0e6a7...
DNSSEC Example

[T] cloudflare.com. 3600 IN DNSKEY 257 3 13 ;{id = 2371 (ksk), size = 256b}
cloudflare.com. 3600 IN DNSKEY 256 3 13 ;{id = 34505 (zsk), size = 256b}
Checking if signing key is trusted:
New key: cloudflare.com. 3600 IN DNSKEY 256 3 13 oJMRESz5...
   Trusted key: . 3600 IN DS 20326 8 2 e06d44b80b8f1d3...
   Trusted key: . 172800 IN DNSKEY 256 3 8 AwEAAc4qsciJ5MdMU...
   Trusted key: com. 86400 IN DNSKEY 256 3 8 AwEAAcpi0...
   Trusted key: cloudflare.com. 3600 IN DNSKEY 256 3 13 oJMRESz5...
Key is now trusted!
[T] www.cloudflare.com. 300 IN DS 2371 13 2 56171d8070a7e777c000824b52799...
[T] www.cloudflare.com. 3600 IN DNSKEY 257 3 13 ;{id = 2371 (ksk), size = 256b}
www.cloudflare.com. 3600 IN DNSKEY 256 3 13 ;{id = 34505 (zsk), size = 256b}
www.cloudflare.com. 300 IN A 104.17.209.9
;;[S] self sig OK; [B] bogus; [T] trusted
Domain Name Registration
Registries and Registrars

http://www.example.com

Second Level Domain (SLD)  Top Level Domain (TLD)
Managed by a registrar under contract with a registry  Managed by a registry

- ICANN/IANA manages the root zone
- Registries manage TLDs (e.g., VeriSign $\rightarrow$.com)
- Registrars manage SLDs (e.g., MarkMonitor $\rightarrow$ google.com)
$ whois google.com

Domain Name: GOOGLE.COM
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.markmonitor.com
Registrar URL: http://www.markmonitor.com
Updated Date: 2019-09-09T15:39:04Z
Creation Date: 1997-09-15T04:00:00Z
Registry Expiry Date: 2028-09-14T04:00:00Z
Registrar: MarkMonitor Inc.

- Domain registration information provided in WHOIS databases
- Thick WHOIS servers: Information for entire TLDs
- Thin WHOIS servers: Pointers to managing registrars
Domain Management

- Domains have value, and therefore are subject to attacks
  - Trademarks, ad revenue, phishing, general abuse of trust
  - e.g., Google owns google.com for a reason
- Domain squatting: Purchasing a domain with the intent to profit off of another entity’s trademark
- Domain drop catching: Instant re-registration of expiring domains
Border Gateway Protocol
The Internet is composed of autonomous systems (AS) corresponding (roughly) to distinct administrative entities:
- ICANN/IANA delegates to regional Internet registries (RIR)
- Five RIRs (AFRINIC, ARIN, APNIC, LACNIC, RIPE) assign ASes
Routing protocols establish how network traffic moves within and between ASes:
- Interior gateway protocols (IGP) establish intra-AS routes
- Exterior gateway protocols (EGP) establish inter-AS routes
The Border Gateway Protocol (BGP) is the *de facto* EGP.
The Border Gateway Protocol (BGP) is the *de facto* EGP.

- BGP allows complex policy specification that can take political, economic, and security concerns into account:
  - “Don’t provide transit for this untrusted country’s traffic”
  - “Prefer routes through this low-cost link”
BGP Entities

**Stub**  Single link to the global AS graph

**Multihomed**  Multiple links, but no transit

**Transit**  Networks that forward traffic between links

- BGP reasons about the Internet as a graph of ASes
- BGP “speakers” (routers) communicate via TCP links to exchange routing information
BGP Routing Information

\[\langle \text{prefix}, \text{next-hop}, \langle \text{AS}_1, \ldots, \text{AS}_n \rangle, \{\text{attr}_1, \ldots, \text{attr}_m\} \rangle\]

- BGP is a distance-vector protocol
  - Router announce network reachability to neighbors
  - Network prefix, next hop, AS path, attributes tuples
- Routes are selected to be installed in the local routing table using a scoring function (only partially standardized)
- Network topology changes are announced in UPDATE messages
Route Selection

(1) Next hop must be reachable
(2) Highest WEIGHT (iBGP, non-standard)
(3) Highest LOCAL_PREFERENCE (iBGP)
(4) Shortest AS path
(5) Lowest ORIGIN
(6) Lowest MULTI_EXIT_DISC
(7) Longest prefix
What guarantees that a BGP announcement is legitimate?
What guarantees that a BGP announcement has not been tampered with?
BGP supports several (weak) authentication mechanisms

- BGP TTL check
- TCP MD5 authentication
- Peer and announcement filtering

But, BGP security is heavily reliant on trust
BGP Hijacking

- Malicious announcements can trick victims into selecting malicious routes
- Attackers can drop traffic (“blackholing”) or hijack traffic by redirecting it through their own AS
- Requires other ASes to select and forward their announcements
  - Shorter AS paths
  - Prefix deaggregation
  - Forged path attributes
Dear All,

I would like to sincerely apologize to everyone everwhere who experienced problems yesterday due to the 7007 AS announcements.

If anyone cares to know, here is what happened:

At 11:30AM, EST, on 25 Apr 1997, our border router, stamped with AS 7007, recieved a full routing view from a downstream ISP (well, a view contacting 23,000 routes anyway).
“Accidental” announcement of 50 prefixes not managed by Rostelecom

Many of these prefixes belonged to financial institutions
  - e.g., MasterCard, Visa, Fortis, Alfa-Bank

Indications that routes were injected into routing tables for “traffic engineering” purposes (i.e., redirection)
  - Some routes were more specific than official routes (e.g., /24 vs. /23)
80 high-traffic prefixes announced by Russian AS 39523
  - Google, Apple, Facebook, Microsoft, Twitch, NTT, Riot Games

Why was this incident suspicious?
  - No (general) announcements seen from this AS in years
  - All hijacked prefixes belonged to important organizations
  - Very specific prefixes as would be used for redirection

Only active once before (August 2017)
  - 701 (Verizon) → 15169 (Google) → 31007 (Equinox) → 39523 (DV-LINK-AS) → 66.232.224.0/24 (Kohls)
Conclusions
In this module, we covered:

- The Domain Name System
- The Border Gateway Protocol
References

