HELLO,
MY NAME IS

Naming and Security

With material from:
Dave Levin
• Naming and its problems
  • DHCP: Getting an IP address
  • DNS: Finding an IP address
Naming

- IP addresses: Global namespace
  - Important: No conflicts allowed!

- Humans aren’t very good at IP addresses
  - How to pick one that isn’t taken?
  - How to remember someone else’s?
DHCP: Address assignment
DHCP
Dynamic Host Configuration Protocol

New host

- Doesn’t have an IP address yet (can’t set src addr)
- Doesn’t know who to ask for one
- Solution: Discover one on the local subnet

DHCP discover (L2 broadcast)

DHCP offer

DHCP request (L2 broadcast)

DHCP ACK

DHCP server

- offer includes: IP address, DNS server, gateway router, and duration of this offer (“lease” time)
- request asks for the offered IP address
DHCP attacks

- Flooding attack: Fake requests to starve real hosts
  - Fake release messages to DoS host
- Discover/request are broadcast: attacker can overhear
- Race actual DHCP server to replace:
  - DNS server: Redirect victim’s lookups at will
  - Gateway server:
    - Where victim sends all outgoing traffic
    - Intercept, relay to real gateway: MITM
DHCP defense: Snooping

• Subnet switch acts like a firewall
  • “Snoop” on L3 information from L2

• Drop offers from untrusted sources before they reach end host

• Bind incoming IP addresses, ports, MACs to avoid MAC spoofing
DNS: Name resolution
Hostnames & IP addresses

gold:~ dml$ ping google.com
PING google.com (74.125.228.65): 56 data bytes
64 bytes from 74.125.228.65: icmp_seq=0 ttl=52 time=22.330 ms
64 bytes from 74.125.228.65: icmp_seq=1 ttl=52 time=6.304 ms
64 bytes from 74.125.228.65: icmp_seq=2 ttl=52 time=5.186 ms
64 bytes from 74.125.228.65: icmp_seq=3 ttl=52 time=12.805 ms

google.com is easy to remember, but not routable

74.125.228.65 is routable

Name resolution:
The process of mapping from one to the other
Domain names

- **www.cs.umd.edu** = “domain name”
  - www.cs.umd.edu is a “subdomain” of cs.umd.edu

- Domain names can map to a set of IP addresses

```
gold:~ dml$ dig google.com

;; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>>HEADER<<- opcode: QUERY, status: NOERROR, id: 35815
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.   IN A

;; ANSWER SECTION:
google.com.   105 IN A  74.125.228.70
google.com.   105 IN A  74.125.228.66
google.com.   105 IN A  74.125.228.64
google.com.   105 IN A  74.125.228.69
google.com.   105 IN A  74.125.228.78
google.com.   105 IN A  74.125.228.73
google.com.   105 IN A  74.125.228.68
google.com.   105 IN A  74.125.228.65
```

We’ll understand this more in a bit; for now, note that **google.com** is mapped to many IP addresses
Terminology

• “zone” = a portion of the DNS namespace, divided up for administrative reasons
  • Think of it like a collection of hostname/IP address pairs that happen to be lumped together
    • www.google.com, mail.google.com, dev.google.com, …

• Subdomains do not need to be in the same zone
  • Allows the owner of one zone (umd.edu) to delegate responsibility to another (cs.umd.edu)
Namespace hierarchy

Zones

.com
.edu
.net

.umd.edu
.cs.umd.edu

.duke.edu

.www.cs.umd.edu
Name server

• Answers queries of the form “What is the IP address for foo.bar.com?”
  • Every zone must run ≥2 nameservers
  • Several very common nameserver implementations: BIND, PowerDNS (more popular in Europe)
Authoritative Name Server

- Every zone has to maintain a file that maps IP addresses and hostnames ("www.cs.umd.edu is 128.8.127.3")
- One of the name servers in the zone has the master copy of this file. It is the authority on the mapping.
Resolver

• While name servers *answer* queries, resolvers *ask* queries.

• Every OS has a resolver. Typically small and pretty dumb. All it typically does is forward the query to a local…
Recursive name server

• Does the heavy lifting: issues queries on behalf of the client resolver until authoritative answer returns

• Prevalence
  • Almost always a local (private) recursive name server
  • Very rare to support recursive queries otherwise
Resource records

• Usually: Mapping IP address to hostname

• But can be other things:
  • (A)ddress records (IP <-> hostname)
  • Mail server (MX, mail exchanger)
  • SOA (start of authority, delineate different zones)
  • Others (e.g., share keys for DNSSEC)

• Records are the unit of information
Name servers in zone must:

• Give **authoritative answers** for hostnames in zone

  “A” record: umd.edu = 54.84.241.99
  54.84.241.99 is a valid IP address for umd.edu

• Give pointers to **name servers for subdomains**
  • umd.edu zone’s nameservers tell us the name and IP address of the cs.umd.edu zone’s nameservers

  Ask ipa01.cs.umd.edu for all cs.umd.edu subdomains
DNS
Domain Name Service at a very high level

What is an IP address for cs.umd.edu?

Caching responses is critical to DNS’s success.

Every response (3,5,7,8) has a time-to-live (TTL). TTLs should be reasonably long (days), but some are minutes.
How do they know these IP addresses?

- Local DNS server: host learned this via DHCP
- Parent knows its children: registration process
- Root nameserver: *hardcoded* into the local DNS server (and every DNS server)
  - 13 root servers (logically): A-root, B-root, …, M-root
  - These IP addresses change *very* infrequently
- **UMD runs D-root.**
  - IP address changed beginning of 2013
Caching

- Central to DNS’s success
- Also central to attacks
- “Cache poisoning”: filling cache with false info
What is an IP address for cs.umd.edu?

Every query (2,4,6) has the same request in it (“what is the IP address for cs.umd.edu?”)

But different:
- dst IP (port = 53)
- query ID
What’s in a response?

• Many things, but for attacks we care about …

• A Record: gives “the authoritative response for the IP address of this hostname”

• NS record: describes “this is the name of the nameserver who should know more about how to answer this query than I do”
  • Often also “glue” records (IP addresses of those name servers to avoid chicken and egg problems)
  • Resolver will generally cache all of this information
Query IDs

- Local resolver has a lot of incoming/outgoing queries at any point in time.
- To determine which response maps to which queries, it uses a query ID.
- Query ID: 16-bit field in the DNS header
  - Requester sets to anything
  - Responder must provide same value in its response

How would you implement query IDs at a resolver?
Query IDs used to increment

- Global query ID value
- Map outstanding query ID to local state of who to respond to (the client)
- Basically: new Packet(queryID++)

How would you attack this?
Cache poisoning

How do you guess this?

Authoritative DNS server ("bank.com")

Local nameserver (recursive)

Bad guy

www.bad.com

16322

16322

16321

16322

A www.bank.com = 6.6.6.6

Will cache www.bank.com = 6.6.6.6 and ignore authority’s answer

www.bank.com

Next is likely 16322

6.6.6.6
Details: Getting it to work

- Must guess query ID: ask for it, and go from there
  - Partial fix: randomize query IDs
  - Problem: small space
  - Attack: issue a lot of query IDs

- Must guess source port number
  - Typically constant for a given server (often always 53)

- The answer must not already be in the cache
  - It will avoid issuing a query in the first place
Cache poisoning
Can we do more harm than a single record?

Will cache “the person to ask for ALL bank.com queries is 6.6.6.6”
Solutions?

- Randomizing query ID?
  - Not sufficient alone: only 16 bits of entropy

- Randomize source port, as well
  - There’s no reason for it stay constant
  - Gets us another 16 bits of entropy

- DNSSEC?
DNSSEC

www.cs.umd.edu?

Ask “.edu”
.edu’s public key = PK_{edu} (Plus “.”’s sig of this zone-key binding)

Root DNS server “.”

www.cs.umd.edu?

Ask “umd.edu”
umd.edu’s public key = PK_{umd} (Plus “edu”’s sig of this zone-key binding)

TLD DNS server (“.edu”)

www.cs.umd.edu?

IN A www.cs.umd.edu 128.8.127.3 (Plus “umd.edu”’s signature of the answer)

Authoritative DNS server (“umd.edu”)

Only the authoritative answer is signed
Properties of DNSSEC

- If everyone has deployed it, and if you know the root’s keys, prevents spoofed responses
  - Very similar to PKIs in this sense

- Unlike PKIs, we still want authenticity even if not everyone has deployed DNSSEC
  - What if someone replies back without DNSSEC?
  - Ignore = secure but you can’t connect to a lot of hosts
  - Accept = can connect but insecure

- Back to our notion of incremental deployment
  - DNSSEC is not all that useful incrementally