Core Protocols: DNS

Michalis Polychronakis

Stony Brook University
Domain Name System

DNS maps domain names to IP addresses

“Phonebook” for the internet

Client: I want to connect to: www.cs.stonybrook.edu

DNS server: here is its IP address: 130.245.27.2

Distributed, hierarchical, reliable database

Replaced the manually maintained /etc/hosts file

Domain names are assigned by registrars accredited by ICANN

Not always a one-to-one mapping

Virtual hosting: many names hosted on a single IP address

Load balancing/fault tolerance: single name hosted on many IP addresses
DNS Server Hierarchy

Hierarchically divided name space

.\edu \rightarrow \text{stonybrook.edu} \rightarrow \text{cs.stonybrook.edu} \rightarrow \text{www.cs.stonybrook.edu}

\textbf{Root} name servers

Responsible for top-level domains (TLDs): .\com, .\edu, .\net, ...

Point to the authoritative name server of each TLD \rightarrow managed by governments or commercial organizations

\begin{verbatim}
$ curl http://data.iana.org/TLD/tlds-alpha-by-domain.txt | wc -l
1451
\end{verbatim}

\textbf{Authoritative} name servers are responsible for a set of names belonging into a \textit{zone}

A leaf node in the DNS hierarchy manages the zone of a single domain
Domain Name

A string that identifies a realm of administrative autonomy, authority, or control
The part of a URL after the protocol (e.g., https://) and before the next slash
The text that a user types into a browser window to reach a particular website

Fully Qualified Domain Name (FQDN)

A specific and complete name that provides an absolute path in the DNS hierarchy
Host name + subdomain + domain name (e.g., www.cs.stonybrook.edu)

Effective Top-level Domain (eTLD)

Depending on the TLD, not all second-level domains can be registered
Example: the .uk name space is sub-divided into different categories controlled by the registrar (.ac.uk for colleges, .co.uk for companies, etc.)
Not only country level: .github.io, .dyndns.org, …
AKA public suffix: a suffix under which users can register names

eTLD+1

eTLD + next level: indicative of the actual domain registrant
Example: stonybrook.edu, bbc.co.uk

DNS Resolvers

Query DNS servers and resolve the requested resource

Main query types:

**Recursive:** query a single server, which may then query (as a client itself) other DNS servers on behalf of the requester

Has to reply with the requested response or “doesn’t exist” (cannot refer the client to a different DNS server)

**Iterative:** query a chain of one or more DNS servers

Each server returns the best answer it has

If the server cannot find an exact match, it returns a referral: a pointer to a server authoritative for a lower level of the domain namespace
Walking the Tree: End User

Applications place resolution requests to the *stub resolver* of the OS

The stub resolver then typically sends DNS queries to a *recursive resolver*

Caches responses for future queries (TTL specified by the domain owner)

Negative responses are cached as well → save time (e.g., misspellings, expired domains)

---

Walking the Tree: Recursive Resolver

Hosts know at least one local DNS recursive resolver

- Usually specified by the ISP or organization through DHCP – users can manually override it
- Uses the hierarchy of zones and delegations to respond to queries for which it is not authoritative
- Caches responses as well

\[ \text{styx.cs.stonybrook.edu?} \quad \text{IP} = 130.245.42.22 \]
DNS message

+---------------------+      
|        Header       |      
+---------------------+      
|       Question      |  the question for the name server      
|       Answer       |  RRs answering the question      
|      Authority      |  RRs pointing toward an authority      
|     Additional     |  RRs holding additional information      
+---------------------+      

DNS header

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Mostly uses UDP ([port 53](https://en.wikipedia.org/wiki/User_Datagram_Protocol)); TCP sometimes is used for long responses and zone transfers
DNS resource record

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| |                                               |
| |                      NAME                     |
| |                                               |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| |                      TYPE                     |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| |                     CLASS                     |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| |                      TTL                      |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| |                   RDLENGTH                    |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
| |                     RDATA                     |
| +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|

**NAME:** Name of the node to which this record pertains

**TYPE:** RR type in numeric form (e.g., 15 for MX RRs)

**CLASS:** Class code

**TTL:** Count of seconds the RR stays valid

**RDLENGTH:** Length of RDATA field

**RDATA:** Additional RR-specific data
Types of Resource Records

Besides translating host addresses, DNS is in essence a generic “directory” service for other host-related information.

- **A**: host IPv4 address
- **AAAA**: host IPv6 address
- **NS**: authoritative name server
- **MX**: mail server of domain
- **CNAME**: aliases for other names (not IP addresses)
- **PTR**: map IP addresses to names (reverse lookup)
- **TXT**: arbitrary data associated with the domain
- **HINFO**: host information
DNS Spoofing/Cache Poisoning

No authentication: responses can be spoofed!
Point to a different address of the attacker’s choosing
Phishing, malware infection, MitM, …
Subverting Name-based Authentication  [Bellovin 1990]

Trusted access based on host names *(not a good idea)*

Server performs reverse DNS lookup to check if a client’s host name is contained in a list of authorized host names

Example: “r-utilities” perform name-based authentication (e.g., permit all hosts in `.rhosts` to `rsh/rlogin` on the server)

Attack: fake a PTR record for an attacker-controlled IP address to return a trusted hostname

When `rsh/rlogin` receives the connection, the reverse lookup using the attacker’s originating IP will return a trusted name…

Fix: cross-check the returned name by performing a name lookup

The returned IP address will not match the attacker’s IP address (IP₁ ➔ name ➔ IP₂)
DNS Poisoning: Different Vantage Points

**Off-path:** attackers cannot observe the victim’s DNS messages (*blind*)

- **Blind injection:** must *guess* the proper values in the forged response fields according to the victim’s query
- **Race condition:** forged response must arrive before the real one

**On-path:** attackers can passively observe the victim’s traffic (DNS queries) and inject properly forged responses (*MotS*)

- Easy to mount in WiFi networks, by ISPs, …
- Race condition: forged response must arrive before the real one

**In-path:** attackers can block responses from reaching the victim, and inject forged ones instead (*MitM*)

- But at this point the attacker can do so much more…

---

[65x468] DNS Poisoning: Different Vantage Points

**Off-path:** attackers cannot observe the victim’s DNS messages (*blind*)

- **Blind injection:** must *guess* the proper values in the forged response fields according to the victim’s query
- **Race condition:** forged response must arrive before the real one

**On-path:** attackers can passively observe the victim’s traffic (DNS queries) and inject properly forged responses (*MotS*)

- Easy to mount in WiFi networks, by ISPs, …
- Race condition: forged response must arrive before the real one

**In-path:** attackers can block responses from reaching the victim, and inject forged ones instead (*MitM*)

- But at this point the attacker can do so much more…
Countering Blind Injection: DNS TXID

Synchronization mechanism between clients and servers

16-bit transaction identifier

Randomly chosen for each query*
Response accepted only if its TXID matches the one used in the request

Attacker has to guess right and win a race

Guess the correct TXID
Response src IP and dst port should match query dst IP and src port

It’s possible! Kaminsky attack

(*) Initial implementations would simply increment the TXID for each new request, which quickly led to successful attacks.
“What's the IP for example.com?”

“Hey, I am an authoritative nameserver. IP address is 192.0.0.17”

“What's the IP for example.com?”

192.0.0.16

192.0.0.17

DNS server

Authoritative nameserver

Attacker

Kaminsky Attack (Dan Kaminsky, 2008)

Goal: poison a DNS server’s cache entry for example.com

The victim server will only accept responses to pending queries it has previously sent itself
Unsolicited responses will be rejected

Requirements for a successful forged response:

Matching question section ➔ trivial: attacker queries the victim server for an A record, which forces the victim to send a controlled query to the respective authoritative server

Matching source and destination IP address ➔ trivial: IP address of the authoritative is known

Matching source and destination UDP port ➔ trivial: pre-Kaminsky, DNS servers would use 53 for source port too (even if different, can be easily inferred if it changes predictably)

Matching TXID ➔ tough: 16 bits of randomness

Additional issue: www.example.com may already be in the recursive’s cache

In that case the recursive will not ask the authoritative ➔ no opportunity for poisoning
**Kaminsky Attack** (Dan Kaminsky, 2008)

The attacker queries the recursive with a subdomain not in the cache

Non-existent subdomains are fine! $\text{foo1.example.com}$

Sidesteps the caching TTL issue (e.g., as would be the case for $\text{www.example.com}$)

Causes the victim resolver to in turn query the authoritative server for the requested subdomain → *the race begins!*

The attacker then floods the resolver with forged responses

Each containing a different guess of the query’s TXID

`;\text{ ANSWER SECTION:}
\text{ foo1.example.com.} \hspace{1em} 120 \hspace{1em} \text{IN} \hspace{1em} \text{A} \hspace{1em} 10.0.0.10
`;\text{ AUTHORITY SECTION:}
\text{ example.com.} \hspace{1em} 86400 \hspace{1em} \text{IN} \hspace{1em} \text{NS}
\text{ ns1.example.com.}
`;\text{ ADDITIONAL SECTION:}
\text{ ns1.example.com.} \hspace{1em} 604800 \hspace{1em} \text{IN} \hspace{1em} \text{A} \hspace{1em} 10.6.6.6

*Fake referral as “glue” record*

**If the race is lost, just repeat with a different subdomain!**
Kaminsky Attack: Key Insights

The recursive will always contact the authoritative of `example.com` for any lookup of a non-existent domain

`foo1.example.com, foo2.example.com, ...`

DNS responses may contain more than a single answer

- Additional responses can contain any type of record (AKA “glue” records)
- The attacker can poison the cache with values in the additional RR field

Today’s internet speeds allow flooding the server with thousands of packets before the real response arrives

- Allows for more than enough TXID guesses

Fix: `source UDP port randomization`

- Orders of magnitude higher entropy by combining TXID + source port number
Even More Entropy: 0x20 Randomization

Domain names are case-insensitive

The request’s question section is always copied verbatim into the response

Opportunity: use the 0x20 bit of ASCII letters to encode an additional value

\[
\begin{align*}
A–Z: & \ 0x41–0x5A, \ a–z: \ 0x61–0x7A \\
A: & \ 01000001 \ (0x41) \\
a: & \ 01100001 \ (0x61)
\end{align*}
\]

Example: the following names will be treated as equal by a responder (for cache matching), but can be treated as unique by a requestor

\[
\begin{align*}
\text{www.example.com} & \quad 1111111111111 \\
\text{WWW.EXAMPLE.COM} & \quad 0000000000000 \\
\text{WwW.eXaMpLe.CoM} & \quad 0101010101010 \\
\text{wWw.ExAmPlE.cOm} & \quad 1010101010101
\end{align*}
\]

Encoded value
Pharming

Traffic redirection at the client side by malware that alters DNS settings

- Change the system’s (or the local router’s) DNS server entry
- Add or override entries in `/etc/hosts`

Example: DNSChanger: estimated 4M infected computers, US$14M profit
(FBI’s “Operation Ghost Click”)

Drive-by pharming

- A malicious web page contains JavaScript code that alters the local router’s DNS
  server from inside the LAN

Dynamic pharming (aka DNS rebinding)

- Quickly switch mapping of `bank.com` between a malicious and a real IP
- Serve malicious script, then switch to the real site ➔ same origin policy is bypassed
More Ways to Intercept Traffic using DNS

Hijacking of existing names by going after registrars
  Using social engineering, stolen credentials, insider attacks, exploits, …

Typosquatting, combosquatting, re-registering expired domains, …

Various attack goals
  Phishing
  Hijack scripts hosted on expired domains still in use by other web pages
  Hijack third-party libraries/modules of popular language repositories (NPM, PyPI, …)
PyPI Python repository hit by typosquatting sneak attack

19 SEP 2017

Security threats, Vulnerability

by John E Dunn

Somebody with time on their hands has tested out a devious new form of typosquatting targeting developers installing Python packages from the PyPI (Python Package Index) repository.

According to an advisory posted to the Slovak National Security Office (NBU), ten packages for Python 2.x were removed from the site after their setup.py files were found to contain malicious code. The bad code was hiding in plain site in the repository, using filenames either nearly identical to, or which could be mistaken for, legitimate ones.
```
crossenv` malware on the npm registry
```

On August 1, a user notified us via Twitter that a package with a name very similar to the popular cross-env package was sending environment variables from its installation context out to npm.hacktask.net. We investigated this report immediately and took action to remove the package. Further investigation led us to remove about 40 packages in total.

On July 19 a user named hacktask published a number of packages with names very similar to some popular npm packages. We refer to this practice as "typo-squatting". In the past, it's been mostly accidental. In a few cases we've seen deliberate typo-squatting by authors of libraries that compete with existing packages. This time, the package naming was both deliberate and malicious—the intent was to collect useful data from tricked users.

All of hacktask's packages have been removed from the npm registry.

Adam Baldwin of Lift Security also looked into this incident to see if there were any other packages, not owned by hacktask, with the same package setup code. He has every file in the public registry indexed by content hash to make scans like this possible. He did not find any other instances of that specific file with those contents.

**exposure**

Following is a list of hacktask's packages, with a count of total downloads from 7/19 to 7/31.
Cybersecurity Incident & Important Consumer Information

Equifax Announces Cybersecurity Incident Involving Consumer Information

No Evidence of Unauthorized Access to Core Consumer or Commercial Credit Reporting Databases

Company to Offer Free Identity Theft Protection and Credit File Monitoring to All U.S. Consumers

September 7, 2017 — Equifax Inc. (NYSE: EFX) today announced a cybersecurity incident potentially impacting approximately 143 million U.S. consumers. Criminals exploited a U.S. website application vulnerability to gain access to certain files. Based on the company’s investigation, the unauthorized access occurred from mid-May through July 2017. The company has found no evidence of unauthorized activity on Equifax’s core consumer or commercial credit reporting databases.
Equifax Seized 138 Scammy Lookalike Domains Instead of Just Changing Its Dumb 'Security' Site

Late last month, Equifax secured control over 138 domains mimicking a website that the company launched in September in the wake of its massive data breach.

Subject to a cybersquatting complaint, the domains were originally purchased through GoDaddy by a Hong Kong company called China Capital Investment Limited. Even now, the domains redirect to placeholder pages full of ads labeled “Identity Theft Protection” and “Protect My Credit” that link to commercial products such as Lifelock.

This summer, after learning that criminal hackers had pilfered the personal and financial information of 143 million Americans, Equifax quickly launched its “Security” site, which is still up. It’s the main focus of the company’s legal response to the breach, but it’s also a front for a new round of cybercrime.

Throughout the summer, the company repeatedly reassured consumers via social media that the breach was over and the site was a secure one. Now that InformationWeek has learned that the domains are still in operation, questions are sure to arise about the company’s claims that it is actively working to stop the cyber gefion.

It’s unclear who is behind the cybersquatting. It could be an innocent purchaser, or perhaps someone is using the site to scam even more people.
Someone Made a Fake Equifax Site. Then Equifax Linked to It.

By Maggie Astor  
Sept. 20, 2017

People create fake versions of big companies’ websites all the time, usually for phishing purposes. But the companies do not usually link to them by mistake.

Equifax, however, did just that after Nick Sweeting, a software engineer, created an imitation of equifaxsecurity2017.com, Equifax’s page about the security breach that may have exposed 143 million Americans’ personal information. Several posts from the company’s Twitter account directed consumers to Mr. Sweeting’s version, securityequifax2017.com. They were deleted after the mistake was publicized.

By Wednesday evening, the Chrome, Firefox and Safari browsers had blacklisted Mr. Sweeting’s site—and he took it down. By that point, the damage was done: some people had already clicked the fake link and were directed to a site that asked for their personal information.
Other DNS Attacks

DoS on root/critical servers
  Or other targets ➔ DNS amplification attacks

Covert DNS communication
  Data exfiltration, C&C, …

Zone transfers
  Reconnaissance

Server bugs
  System compromise

Censorship
  Block websites at the domain level
DNSSEC

Goal: authenticate and ensure the integrity of DNS requests and responses
Non-goals: availability, confidentiality

Cryptographically signed resource records: resolvers can verify the signature

Two new resource types:

**DNSKEY**: creates a hierarchy of trust within each zone  
Name = Zone domain name; Value = Public key for the zone

**RRSIG**: Prevents hijacking and spoofing  
Name = (type, name) tuple (the query itself); Value = Signature of the results

Not a complete solution

- Enables DoS amplification/CPU exhaustion attacks
- Forgery of delegation records still possible
- No “last mile” protection (between the DNS client and its local DNS server)
**DoH/DoT** (DNS over HTTPS/TLS)

Both protocols use end-to-end encryption between the client and the DoH/DoT-based DNS resolver

- **Privacy**: DNS requests cannot be monitored (e.g., nosy ISPs, censorship)
- **Security**: DNS responses cannot be manipulated (e.g., MitM/MotS)

**DoH**: queries and responses are transferred over HTTPS ([RFC8484](https://tools.ietf.org/html/rfc8484))

**DoT**: queries and responses are transferred over TLS ([RFC7858](https://tools.ietf.org/html/rfc7858))

Main difference: DoT uses its own standard port (853) → can be trivially blocked

Better for corporate environments where administrators need to maintain control

Since February 2020, Firefox uses DoH by default for users in the USA

- With Cloudflare as the default provider (NextDNS or a custom server can be selected)
DoH/DoT Privacy Concerns

Trust is shifted to a different entity
   The ISP cannot monitor the traffic, but the DoH server provider now can

Mozilla’s Trusted Recursive Resolver program: Cloudflare has committed to
   i) throwing away all PII after 24 hours
   ii) never provide that data to third parties
   iii) regular audits

Mitigation
   Spread requests across multiple DoH vendors (K-resolver)
   Introduce intermediate proxies (Oblivious DoH)
DoH/DoT Security Concerns

Reduced visibility and control

Real-time DNS monitoring is invaluable for threat detection
Analysis of logged DNS data is invaluable for incident response and forensics

DNS-level enterprise policies (filtering) becomes challenging

The local DNS resolver is sidestepped: the web browser itself securely connects to the remote DoH server

Mitigation

Use endpoint monitoring software (attackers can still tamper with it, not possible for BYOD environments)
Intercept HTTPS (some organizations do that anyway)
It's not DNS
There's no way it's DNS
It was DNS