Network Attacks & Control

CS 161: Computer Security

Prof. Vern Paxson

TAs: Paul Bramsen, Apoorva Dornadula, David Fifield, Mia Gil Epner, David Hahn, Warren He, Grant Ho, Frank Li, Nathan Malkin, Mitar Milutinovic, Rishabh Poddar, Rebecca Portnoff, Nate Wang

http://inst.eecs.berkeley.edu/~cs161/

March 16, 2017
`dig eecs.mit.edu A`

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
; ; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

; QUESTION SECTION:
;eecs.mit.edu.  IN  A

; ANSWER SECTION:
;eecs.mit.edu. 21600 IN A 18.62.1.6

; AUTHORITY SECTION:
mit.edu. 11088 IN NS BITSY.mit.edu.
mit.edu. 11088 IN NS W20NS.mit.edu.
mit.edu. 11088 IN NS STRAWB.mit.edu.

; ADDITIONAL SECTION:
STRAWB.mit.edu. 126738 IN A 18.71.0.151
BITSY.mit.edu. 166408 IN A 18.72.0.3
W20NS.mit.edu. 126738 IN A 18.70.0.160
```

"Answer" tells us the IP address associated with eecs.mit.edu is 18.62.1.6 and we can cache the result for 21,600 seconds.
In general, a single *Resource Record* (RR) like this includes, left-to-right, a DNS name, a *time-to-live*, a family (*IN* for our purposes - ignore), a type (*A* here, which stands for “Address”), and an associated value.
Authority tells us the name servers responsible for the answer. Each RR gives the hostname of a different name server ("NS") for names in mit.edu. We should cache each record for 11,088 seconds.

If the "Answer" had been empty, then the resolver's next step would be to send the original query to one of these name servers.
dig eecs.mit.edu A

;; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160

“Additional” provides extra information to save us from making separate lookups for it, or helps with bootstrapping. Here, it tells us the IP addresses for the hostnames of the name servers. We add these to our cache.
DNS Protocol

Lightweight exchange of *query* and *reply* messages, both with same message format.

Primarily uses **UDP** for its transport protocol, which is what we’ll assume.

Frequently, both clients and servers use port 53.

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SRC port</strong></td>
<td><strong>DST port</strong></td>
</tr>
<tr>
<td><strong>checksum</strong></td>
<td><strong>length</strong></td>
</tr>
</tbody>
</table>

**UDP Header**

**UDP Payload**

**IP Header**

**DNS Query** or **Reply**

- Identification
- Flags
- # Questions
- # Answer RRs
- # Authority RRs
- # Additional RRs
- Questions (variable # of resource records)
- Answers (variable # of resource records)
- Additional information (variable # of resource records)

**UDP Payload**

**UDP Header**

**IP Header**

**DNS Query** or **Reply**

- Identification
- Flags
- # Questions
- # Answer RRs
- # Authority RRs
- # Additional RRs
- Questions (variable # of resource records)
- Answers (variable # of resource records)
- Additional information (variable # of resource records)
DNS Protocol

Lightweight exchange of *query* and *reply* messages, both with same message format.

Primarily uses **UDP** for its transport protocol, which is what we’ll assume.

Frequently, both clients and servers use port 53.
DNS Protocol, con’t

Message header:

- **Identification**: 16 bit # for query, reply to query uses same #
- Along with repeating the Question and providing Answer(s), replies can include “**Authority**” (name server responsible for answer) and “**Additional**” (info client is likely to look up soon anyway)
- Each **Resource Record** has a **Time To Live** (in seconds) for **caching** *(not shown)*
dig eecs.mit.edu A

; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
; ; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

; ; QUESTION SECTION:
eecs.mit.edu.

; ; ANSWER SECTION:
eecs.mit.edu. 21600 IN A 18.62.1.6

; ; AUTHORITY SECTION:
mit.edu. 11088 IN NS BITSY.mit.edu.
mit.edu. 11088 IN NS W20NS.mit.edu.
mit.edu. 11088 IN NS STRAWB.mit.edu.

; ; ADDITIONAL SECTION:
STRAWB.mit.edu. 126738 IN A 18.71.0.151
BITSY.mit.edu. 166408 IN A 18.72.0.3
W20NS.mit.edu. 126738 IN A 18.70.0.160

What if the mit.edu server is untrustworthy? Could its operator steal, say, all of our web surfing to Facebook?
Let’s look at a flaw in the original DNS design (since fixed)
dig eecs.mit.edu A

; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu. 21600 IN A 18.62.1.6

;; AUTHORITY SECTION:
mit.edu. 11088 IN NS BITSY.mit.edu.
mit.edu. 11088 IN NS W20NS.mit.edu.

;; ADDITIONAL SECTION:
www.facebook.com 30 IN A 18.6.6.6
BITSY.mit.edu. 166408 IN A 18.72.0.3
W20NS.mit.edu. 126738 IN A 18.70.0.160

What could happen if the mit.edu server returns the following to us instead?
dig eecs.mit.edu A

; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
; ; global options: +cmd
; ; Got answer:
; ; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
; ; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

; ; QUESTION SECTION:
; eecs.mit.edu.

; ; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

; ; AUTHORITY SECTION:
mit.edu.                11088   IN      NS       BITSY.mit.edu.
mit.edu.                11088   IN      NS       W20NS.mit.edu.

; ; ADDITIONAL SECTION:
www.facebook.com        30      IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160

We’d dutifully store in our cache a mapping of
www.facebook.com to an IP address under
MIT’s control. (It could have been any IP
address they wanted, not just one of theirs.)
In this case they chose to make the mapping disappear after 30 seconds. They could have made it persist for weeks, or disappear even quicker.
dig eecs.mit.edu A

; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.

;; ADDITIONAL SECTION:
www.facebook.com        30      IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160

Next time one of our clients starts to connect to www.facebook.com, it will ask our resolver for the corresponding IP address. The resolver will find the answer in its cache and return 18.6.6.6 😯
dig eecs.mit.edu A

; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
eecs.mit.edu. IN A

;; ANSWER SECTION:
eecs.mit.edu. 21600 IN A 18.62.1.6

;; AUTHORITY SECTION:
mit.edu. 11088 IN NS BITSY.mit.edu.
mit.edu. 11088 IN NS W20NS.mit.edu.

;; ADDITIONAL SECTION:
www.facebook.com 30 IN A 18.6.6.6
BITSY.mit.edu. 166408 IN A 18.72.0.3
W20NS.mit.edu. 126738 IN A 18.70.0.160

How do we fix such cache poisoning?
dig eecs.mit.edu A

Don’t accept **Additional** records unless they’re for the domain of the name server we queried

E.g., contacting a name server for mit.edu ⇒ only accept additional records from *.*.mit.edu

No extra risk in accepting these since server could return them to us directly in an **Answer** anyway.

<table>
<thead>
<tr>
<th>Domain</th>
<th>TTL</th>
<th>Type</th>
<th>Class</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>eecs.mit.edu</td>
<td>21600</td>
<td>IN</td>
<td>A</td>
<td>18.62.1.6</td>
</tr>
<tr>
<td>mit.edu</td>
<td>11088</td>
<td>IN</td>
<td>NS</td>
<td>BITSY.mit.edu</td>
</tr>
<tr>
<td>mit.edu</td>
<td>11088</td>
<td>IN</td>
<td>NS</td>
<td>W20NS.mit.edu</td>
</tr>
<tr>
<td>mit.edu</td>
<td>30</td>
<td>IN</td>
<td>NS</td>
<td><a href="http://www.facebook.com">www.facebook.com</a></td>
</tr>
<tr>
<td><a href="http://www.facebook.com">www.facebook.com</a></td>
<td>30</td>
<td>IN</td>
<td>A</td>
<td>18.6.6.6</td>
</tr>
<tr>
<td>BITSY.mit.edu</td>
<td>166408</td>
<td>IN</td>
<td>A</td>
<td>18.72.0.3</td>
</tr>
<tr>
<td>W20NS.mit.edu</td>
<td>126738</td>
<td>IN</td>
<td>A</td>
<td>18.70.0.160</td>
</tr>
</tbody>
</table>
dig eecs.mit.edu A

Don’t accept **Additional** records unless they’re for the domain of the name server we queried

E.g., contacting a name server for mit.edu ⇒ only accept additional records from *.mit.edu

No extra risk in accepting these since server could return them to us directly in an **Answer** anyway.

This is called “bailiwick checking”.

---

**bailiwick**  | ˈbāləˌwɪk | noun

1 (one's **bailiwick**) one's sphere of operations or particular area of interest: you never give the presentations—that's my **bailiwick**.
The Many Moving Pieces
In a DNS Lookup of www.isc.org

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Resolver’s cache

Authority Server (the “root”)

? A www.isc.org
Answers:
Authority:
org. NS a0.afilias-nst.info
Additional:
a0.afilias-nst.info A 199.19.56.1

User’s ISP’s? A www.isc.org
Recursive Resolver

? A www.isc.org
# The Many Moving Pieces

In a DNS Lookup of www.isc.org

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.</td>
<td>NS</td>
<td>a0.afilias-nst.info</td>
<td>172800</td>
</tr>
<tr>
<td>a0.afilias-nst.info</td>
<td>A</td>
<td>199.19.56.1</td>
<td>172800</td>
</tr>
</tbody>
</table>

Resolver’s cache

Authority Server (the “root”)
The Many Moving Pieces
In a DNS Lookup of \texttt{www.isc.org}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{org.}</td>
<td>NS</td>
<td>a0.afilias-nst.info</td>
<td>172800</td>
</tr>
<tr>
<td>\texttt{a0.afilias-nst.info.}</td>
<td>A</td>
<td>199.19.56.1</td>
<td>172800</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Resolver’s cache

User’s ISP’s? \texttt{A www.isc.org}

Recursive Resolver

Authority Server

\texttt{Answers:}

\texttt{Authority:}

\texttt{isc.org. NS sfba.sns-pb.isc.org.}

\texttt{isc.org. NS ns.isc.afilias-nst.info.}

\texttt{Additional:}

\texttt{sfba.sns-pb.isc.org. A 199.6.1.30}

\texttt{ns.isc.afilias-nst.info. A 199.254.63.254}
The Many Moving Pieces In a DNS Lookup of **www.isc.org**

User’s ISP’s? A **www.isc.org**
Recursive Resolver

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.</td>
<td>NS</td>
<td>a0.afilias-nst.info</td>
<td>172800</td>
</tr>
<tr>
<td>a0.afilias-nst.info</td>
<td>A</td>
<td>199.19.56.1</td>
<td>172800</td>
</tr>
<tr>
<td>isc.org.</td>
<td>NS</td>
<td>sfba.sns-pb.isc.org.</td>
<td>86400</td>
</tr>
<tr>
<td>isc.org.</td>
<td>NS</td>
<td>ns.isc.afilias-nst.info</td>
<td>86400</td>
</tr>
<tr>
<td>sfbay.sns-pb.isc.org.</td>
<td>A</td>
<td>199.6.1.30</td>
<td>86400</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Resolver’s cache

isc.org. Authority Server

? A **www.isc.org**

Answers:
www.isc.org. A 149.20.64.42

Authority:
isc.org. NS ns.isc.afilias-nst.info.

Additional:
sfbay.sns-pb.isc.org. A 199.6.1.30
ns.isc.afilias-nst.info. A 199.254.63.254
The Many Moving Pieces
In a DNS Lookup of www.isc.org

User's ISP's Recursive Resolver Answers: www.isc.org A 149.20.64.42

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.</td>
<td>NS</td>
<td>a0.afilias-nst.info</td>
<td>172800</td>
</tr>
<tr>
<td>a0.afilias-nst.info.</td>
<td>A</td>
<td>199.19.56.1</td>
<td>172800</td>
</tr>
<tr>
<td>isc.org.</td>
<td>NS</td>
<td>sfba.sns-pb.isc.org.</td>
<td>86400</td>
</tr>
<tr>
<td>isc.org.</td>
<td>NS</td>
<td>ns.isc.afilias-net.info</td>
<td>86400</td>
</tr>
<tr>
<td>sfbay.sns-pb.isc.org.</td>
<td>A</td>
<td>199.6.1.30</td>
<td>86400</td>
</tr>
<tr>
<td><a href="http://www.isc.org">www.isc.org</a></td>
<td>A</td>
<td>149.20.64.42</td>
<td>600</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Resolver's cache
What about **blind spoofing**?

- Say we look up mail.google.com; how can an **off-path** attacker feed us a **bogus A answer** before the legitimate server replies?

- How can such a **remote** attacker even know we are looking up mail.google.com?

Suppose, e.g., we visit a web page under their control:

```html
...<img src="http://mail.google.com" ...> ...
```
DNS Threats, con’t

What about *blind spoofing*?

- Say we look up `mail.google.com`; how can an off-path attacker feed us a bogus A answer before the legitimate server replies?
- How can such an attacker even know we are looking up `mail.google.com`?
  
  Suppose, e.g., we visit a webpage under their control:
  
  ```html
  ...<img src="http://mail.google.com" ...> ...
  ```

This HTML snippet causes our browser to try to fetch an image from `mail.google.com`. To do that, our browser first has to look up the IP address associated with that name.
DNS Blind Spoofing, con’t

Once they know we’re looking it up, they just have to guess the Identification field, and reply before legit server.

How hard is that?

Originally, identification field incremented by 1 for each request. How does attacker guess it?

They observe ID k here

So this will be k+1

<img src="http://badguy.com" …>

<img src="http://mail.google.com" …>

Fix?
DNS Blind Spoofing, con’t

Once we **randomize** the Identification, attacker has a 1/65536 chance of guessing it correctly.

*Are we pretty much safe?*

Attacker can send *lots* of replies, not just one …

**However:** once a reply from legit server arrives (with correct Identification), it’s **cached** and no more opportunity to poison it. Victim is inoculated!

Unless attacker can send 1000s of replies before legit arrives, we’re likely safe - phew!?
DNS Blind Spoofing (Kaminsky 2008)

• Two key ideas:
  – Spoof uses **Additional** field (rather than **Answer**)
  – Attacker can get around caching of legit replies by generating a **series** of **different** name lookups:

```html
<img src="http://random1.google.com" ...>
<img src="http://random2.google.com" ...>
<img src="http://random3.google.com" ...>
...
<img src="http://randomN.google.com" ...>
```
Kaminsky Blind Spoofing, con’t

For each lookup of randomk.google.com, attacker spoofs a bunch of records like this, each with a different Identifier.

Once they win the race, not only have they poisoned mail.google.com ...

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

;; ANSWER SECTION:
randomk.google.com 21600 IN A doesn’t matter

;; AUTHORITY SECTION:
google.com. 11088 IN NS mail.google.com

;; ADDITIONAL SECTION:
mail.google.com 126738 IN A 6.6.6.6

6.6.6.6
Kaminsky Blind Spoofing, con’t

For each lookup of randomk.google.com, attacker spoofs a bunch of records like this, each with a different Identifier

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

; ANSWER SECTION:
randomk.google.com 21600 IN A doesn’t matter

; AUTHORITY SECTION:
google.com. 11088 IN NS mail.google.com

; ADDITIONAL SECTION:
mail.google.com 126738 IN A 6.6.6.6

Once they win the race, not only have they poisoned mail.google.com ... but also the cached NS record for google.com’s name server - so any future X.google.com lookups go through the attacker’s machine
Defending Against Blind Spoofing

Central problem: all that tells a client they should accept a response is that it matches the Identification field.

With only 16 bits, it lacks sufficient entropy: even if truly random, the search space an attacker must brute force is too small.

Where can we get more entropy?
Defending Against Blind Spoofing

Central problem: all that tells a client they should accept a response is that it matches the Identification field.

With only 16 bits, it lacks sufficient entropy: even if truly random, the search space an attacker must brute force is too small.

Where can we get more entropy? (Without requiring a protocol change.)
Defending Against Blind Spoofing

For requestor to receive DNS reply, needs both correct Identification and correct ports.

On a request, DST port = 53. SRC port usually also 53 - but not fundamental, just convenient.

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC=53</td>
<td>DST=53</td>
</tr>
<tr>
<td>checksum</td>
<td>length</td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td># Questions</td>
<td># Answer RRs</td>
</tr>
<tr>
<td># Authority RRs</td>
<td># Additional RRs</td>
</tr>
</tbody>
</table>

Questions (variable # of resource records)
Answers (variable # of resource records)
Authority (variable # of resource records)
Additional information (variable # of resource records)

Total entropy: 16 bits
Defending Against Blind Spoofing

“Fix”: client uses random source port ⇒ attacker doesn’t know correct dest. port to use in reply

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC=53</td>
<td>DST=rnd</td>
</tr>
<tr>
<td>checksum</td>
<td>length</td>
</tr>
</tbody>
</table>

Identification
Flags
# Questions  # Answer RRs
# Authority RRs  # Additional RRs
Questions (variable # of resource records)
Answers (variable # of resource records)
Authority (variable # of resource records)
Additional information (variable # of resource records)

Total entropy: ? bits
Defending Against Blind Spoofing

“Fix”: client uses random source port ⇒ attacker doesn’t know correct dest. port to use in reply

32 bits of entropy makes it orders of magnitude harder for attacker to guess all the necessary fields and dupe victim into accepting spoof response.

<table>
<thead>
<tr>
<th>16 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC=53</td>
<td>DST=rnd</td>
</tr>
<tr>
<td>checksum</td>
<td>length</td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
</tr>
<tr>
<td># Questions</td>
<td># Answer RRs</td>
</tr>
<tr>
<td># Authority RRs</td>
<td># Additional RRs</td>
</tr>
</tbody>
</table>

Questions (variable # of resource records)
Answers (variable # of resource records)
Authority (variable # of resource records)
Additional information (variable # of resource records)
"Fix": client uses random source port ⇒ attacker doesn’t know correct dest. port to use in reply

32 bits of entropy makes it orders of magnitude harder for attacker to guess all the necessary fields and dupe victim into accepting spoof response.

This is what primarily “secures” DNS against blind spoofing today. (Note: not all resolvers have implemented random source ports!)
Summary of DNS Security Issues

- DNS threats highlight:
  - Attackers can attack opportunistically rather than eavesdropping
    - Cache poisoning only required victim to look up some name under attacker’s control (*has been fixed*)
  - Attackers can often manipulate victims into vulnerable activity
    - E.g., IMG SRC in web page to force DNS lookups
  - Crucial for identifiers associated with communication to have sufficient entropy (= a lot of bits of unpredictability)
  - “Attacks only get better”: threats that appears technically remote can become practical due to unforeseen cleverness