Network Security
DNS, DDOS and Firewalls

April 15, 2015

Lecture by Kevin Chen

Slides credit: Vern Paxson, Dawn Song

DNS Background

Host Names vs. IP addresses

- Host names
  - Examples: `www.cnn.com` and `bbc.co.uk`
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provide little (if any) information about location
- IP addresses
  - Examples: `64.236.16.20` and `212.58.224.131`
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location

Mapping Names to Addresses

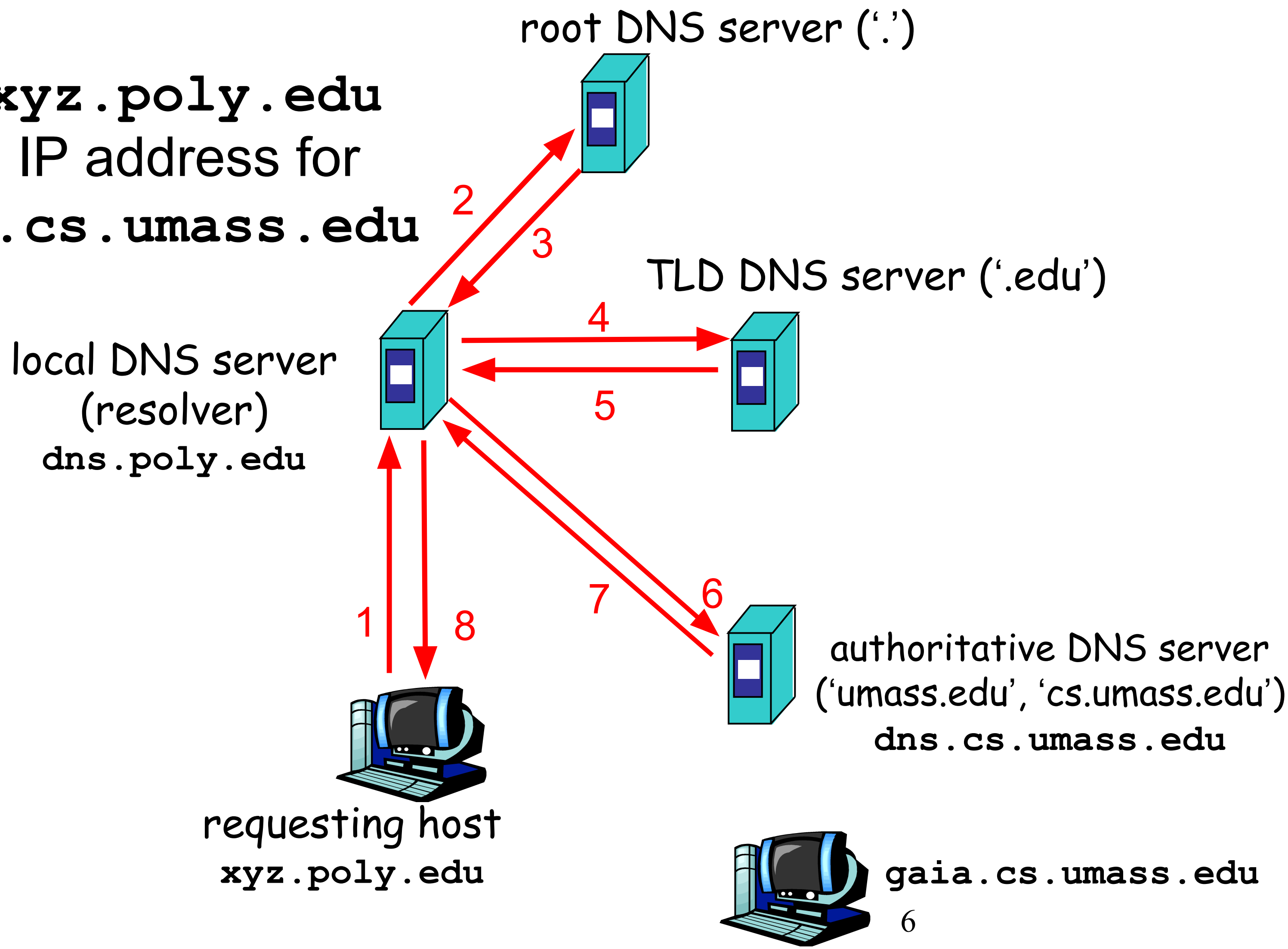
- Domain Name System (DNS)
  - Hierarchical name space divided into zones
  - Zones distributed over collection of DNS servers
  - (Also separately maps addresses to names)
- Hierarchy of DNS servers
  - Root (hardwired into other servers)
  - Top-level domain (TLD) servers
  - "Authoritative" DNS servers (e.g. for *berkeley.edu*)

Mapping Names to Addresses

- Domain Name System (DNS)
  - Hierarchical name space divided into zones
  - Zones distributed over collection of DNS servers
  - (Also separately maps addresses to names)
- Hierarchy of DNS servers
  - Root (hardwired into other servers)
  - Top-level domain (TLD) servers
  - “Authoritative” DNS servers (e.g. for *berkeley.edu*)
- Performing the translations
  - Each computer configured to contact a *resolver*

Example

Host at **`xyz.poly.edu`** wants IP address for **`gaia.cs.umass.edu`**



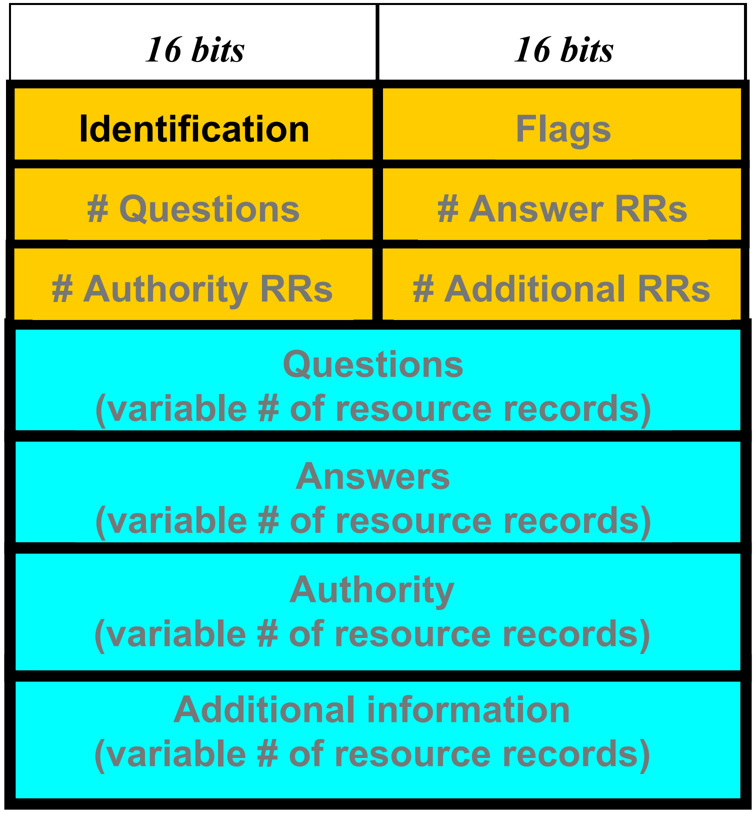
DNS Protocol

DNS protocol: *query* and *reply* messages, both with same *message format*

(Mainly uses UDP transport rather than TCP)

Message header:

- Identification: 16 bit # for query, reply to query uses same #
- Replies can include "Authority" (name server responsible for answer) and "Additional" (info client is likely to look up soon anyway)
- Replies have a Time To Live (in seconds) for **caching**



dig eecs.mit.edu A

Use Unix "`dig`" utility to look up DNS address ("A") for hostname `eecs.mit.edu`

```
; ; <<>> 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
```

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
```

These are just comments from `dig` itself with details of the request/response

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
```

Transaction identifier

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.                                   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
```

Here the server echoes back the question that it is answering

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; QU                                      IONAL: 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 its address is `18.62.1.6` and we can cache the result for 21,600 seconds

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
```

"Authority" tells us the *name servers* responsible for the answer. Each record gives the *hostname* of a different name server ("`NS`") for names in `mit.edu`. We should cache each record for 11,088 seconds.

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.
```

"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.

```
;; ANSWER SECTION
eecs.mit.edu.

;; 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
```

Non-Eavesdropping Threats: DNS

- DHCP attacks show brutal power of attacker who can eavesdrop
- Consider attackers who *can't* eavesdrop - but still aim to manipulate us via how protocols function
  - As a DNS resolver
  - Off-path DNS spoofing
- DNS: path-critical for just about everything we do
  - Maps hostnames ⇔ IP addresses
  - Design only **scales** if we can minimize lookup traffic
    - #1 way to do so: caching
    - #2 way to do so: return not only answers to queries, but additional info that will likely be needed shortly
- Directly interacting w/ DNS: `dig` program on Unix
  - Allows querying of DNS system
  - Dumps each field in DNS responses

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.               30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.     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 happens 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.                 IN        A

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.               11088   IN        NS        BITSY.mit.edu.
mit.edu.               11088   IN        NS        W20NS.mit.edu.
mit.edu.               30      IN        NS        eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.     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 dutifully store in our cache a mapping of `eecs.berkeley.edu` to an IP address under MIT's control. (It could have been any IP address they wanted, not just one of theirs.)

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.                                                6

;; AUTHORITY SECTION:
mit.edu.                11088   IN        NS       BITSY.mit.edu.
mit.edu.                11088   IN        NS       W20NS.mit.edu.
mit.edu.                30      IN        NS       eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.      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
```

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.                  IN      A

;; ANSWER SECTION
eecs.mit.edu.
```

How do we fix such ***cache poisoning***?

```
;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.      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
```

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +c
;; Got answer:
;; ->>HEADER<<- opcod
;; flags: qr rd ra; Q

;; 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.                 30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.       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
```

Don’t accept Additional records unless they’re for the domain we’re looking up

E.g., looking up `eecs.mit.edu` ⇒ only accept additional records from `*.mit.edu`

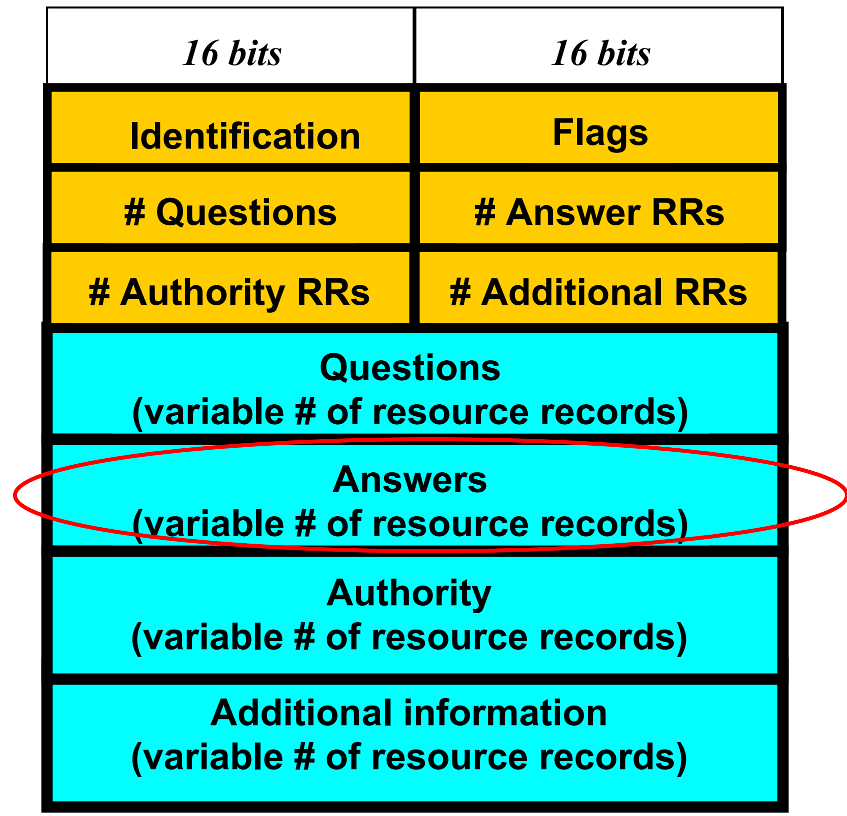
No extra risk in accepting these since server could return them to us directly in an Answer anyway.

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`?



The attacker can "force" it:

```
<img src="http://mail.google.com" ...>
```

DNS Blind Spoofing, con't

Fix?

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?

16 bits	16 bits
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)	

```
<img src="http://badguy.com" ...>
<img src="http://mail.google.com" ...>
```

They observe ID k here

So this will be k+1

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 reply from legit server arrives (with correct Identification), it's **cached** and no more opportunity to poison it. Victim is innoculated!

16 bits	*16 bits*
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)	

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:

```
<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 returns 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` …

Kaminsky Blind Spoofing, con’t

For each lookup of `randomk.google.com`, attacker returns 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? (*Without* requiring a protocol change.)

16 bits	*16 bits*
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)	

Defending Against Blind Spoofing

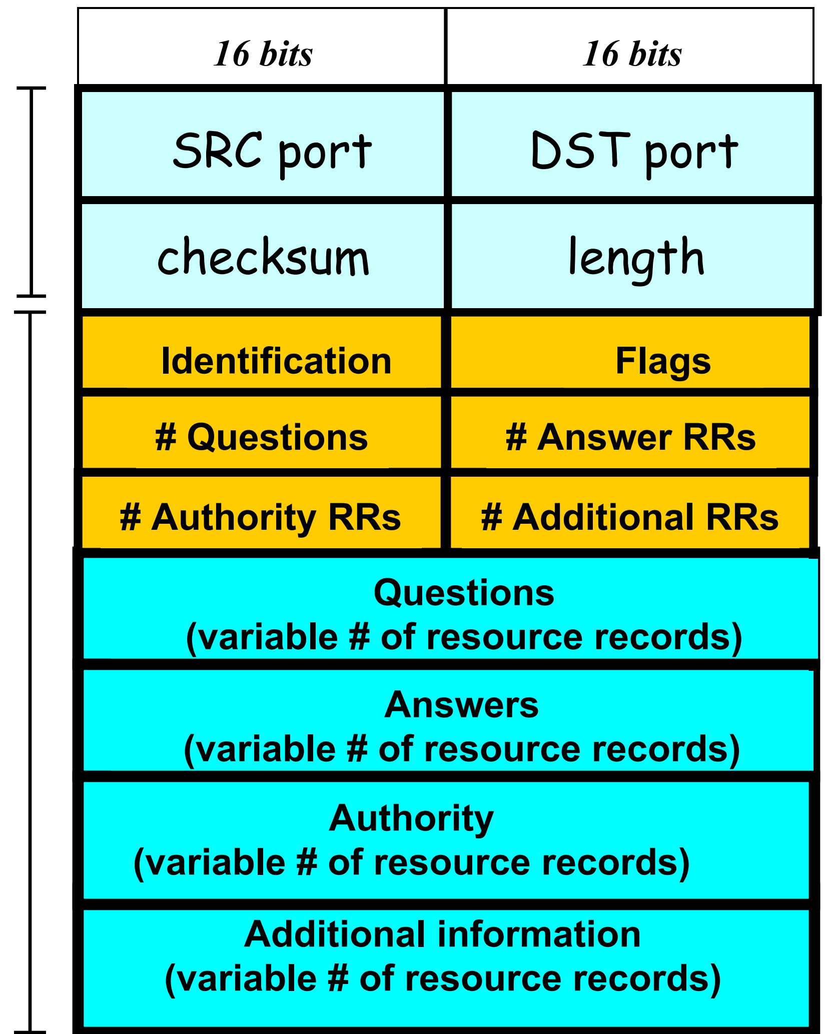
DNS (primarily) uses UDP for transport rather than TCP.

UDP Header

UDP header has:

- 16-bit Source & Destination ports (identify processes, like w/ TCP)
- 16-bit checksum, 16-bit length

UDP Payload



Defending Against Blind Spoofing

Total *entropy*: 16 bits

DNS (primarily) uses UDP for transport rather than TCP.

UDP header has:

- 16-bit Source & Destination ports (identify processes, like w/ TCP)
- 16-bit checksum, 16-bit length

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

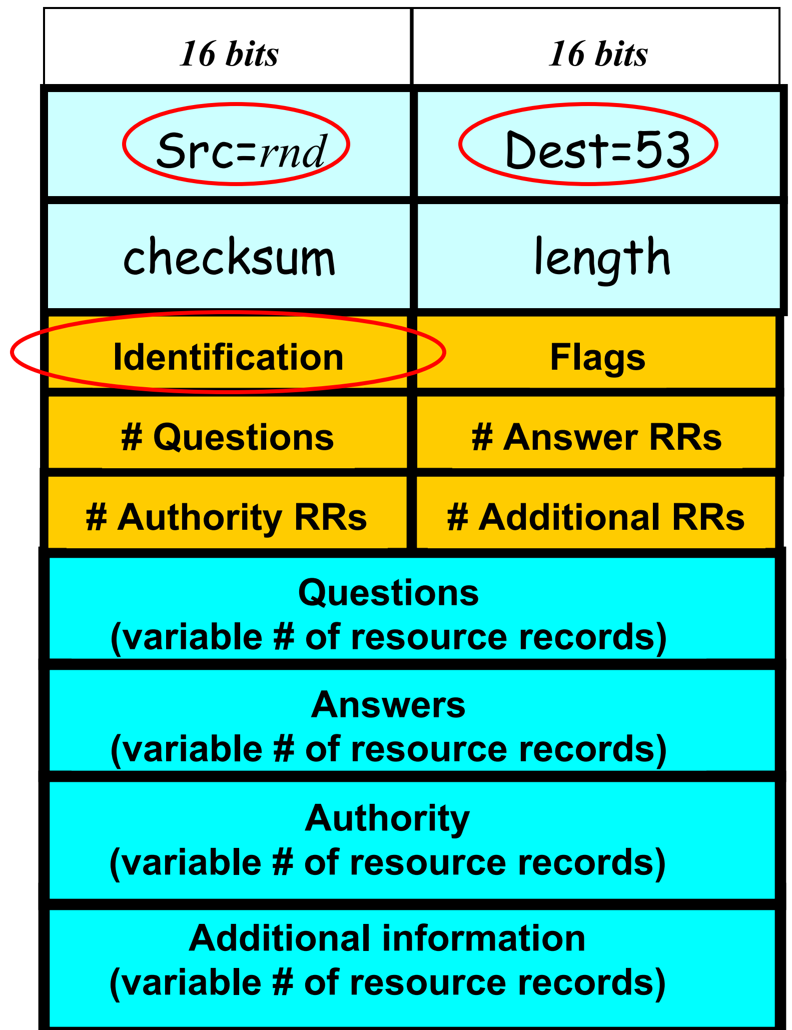
On a request, DST port = 53. SRC port usually also 53 - but not fundamental, just convenient

16 bits	*16 bits*
Src=53	Dest=53
checksum	length
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)	

Defending Against Blind Spoofing

“Fix”: use random source port

Total *entropy*: ? bits

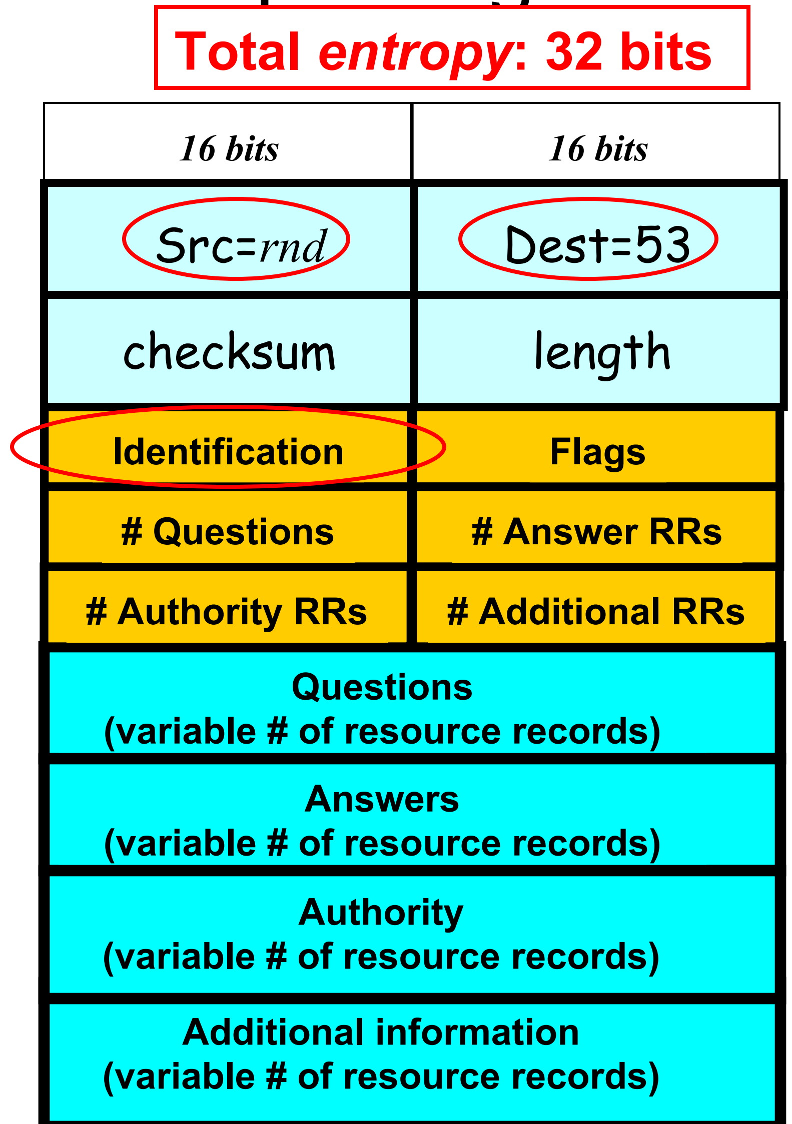


Defending Against Blind Spoofing

“Fix”: use random source port

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 today. (Note: not all resolvers have implemented random source ports!)



Denial-of-Service (DoS) Attacks

Attacks on Availability

- Denial-of-Service (DoS)
- Preventing legitimate users from using a service

- We need to consider our threat model
- What might motivate a DoS attack?

Botnets Beat Spartan Laser on *Halo 3*

By Kevin Poulsen   February 4, 2009 | 12:13 pm | Categories: Cybarmageddon!

What's the most powerful weapon you can wield when playing *Halo 3* online?

I know. You can control the entire map with a battle rifle and a couple of sticky grenades. But that teeny-bopper you just pwned has you beat with the tiny botnet he leased with his allowance money.

KrebsonSecurity
In-depth security news and investigation

There are dozens of underground forums where members advertise their ability to execute debilitating "distributed denial-of-service" or DDoS attacks for a price. DDoS attack services tend to charge the same prices, and the average rate for taking a Web site offline is surprisingly affordable: about $5 to $10 per hour; $40 to $50 per day; $350-$400 a week; and upwards of $1,200 per month.

An ad for a DDoS attack service.

Of course, it pays to read the fine print before you enter into any contract. Most DDoS services charge varying rates depending on the complexity of the target's infrastructure, and how much lead time the attack service is given to size up the mark. Still, buying in bulk always helps: One service advertised on several fraud forums offered discounts for regular and wholesale customers.

Extortion via DDoS on the rise

By Denise Pappalardo and Ellen Messmer, Network World, 05/16/05

Criminals are increasingly targeting corporations with distributed denial-of-service attacks designed not to disrupt business networks but to extort thousands of dollars from the companies.

Ivan Maksakov, Alexander Petrov and Denis Stepanov were accused of receiving $4 million from firms that they threatened with cyberattacks.

The trio concentrated on U.K. Internet gambling sites, according to the prosecution. One bookmaker, which refused to pay a demand for $10,000, was attacked and brought offline--which reportedly cost it more than $200,000 a day in lost business.

NOV 06
8

DDoS makes a phishing e-mail look real

Posted by Munir Kotadia @ 12:00

0 comments

Just as Internet users learn that clicking on a link in an e-mail purporting to come from their bank is a bad idea, phishers seem to be developing a new tactic -- launch a DDoS attack on the Web site of the company whose customers they are targeting and then send e-mails "explaining" the outage and offering an "alternative" URL.

November 17th, 2008

Anti fraud site hit by a DDoS attack

Posted by Dancho Danchev @ 4:01 pm

Categories: Botnets, Denial of Service (DoS), Hackers, Malware, Pen testing...
Tags: Security, Cybercrime, DDoS, Fraud, Bobbear...

9 TalkBacks ADD YOUR OPINION | SHARE | PRINT | E-MAIL | WORTHWHILE? +2 4 VOTES

The popular British anti-fraud site **Bobbear.co.uk** is currently under a DDoS attack (distributed denial of service attack) , originally launched last Wednesday, and is continuing to hit the site with 3/4 million hits daily from hundreds of thousands of malware infected hosts mostly based in Asia and Eastern Europe, according to the site's owner. Targeted DDoS attacks against anti-fraud and volunteer cybercrime fighting communities clearly indicate the impact these communities have on the revenue stream of scammers, and with Bobbear attracting such a high profile underground attention, the site is indeed doing a very good job.

---

9. In the past year, has your site been subjected to a denial of service attack, meaning an attacker prevented or attempted to prevent access to your site altogether?

#	Answer	Bar	Response	%
1	yes		21	62%
2	no		8	24%
3	not sure		5	15%
	Total		34	

Row over Korean election DDoS attack heats up

Ruling party staffer accused of disrupting Seoul mayoral by-election

By **John Leyden** • **Get more from this author**

Posted in Security, 7th December 2011 09:23 GMT

A political scandal is brewing in Korea over alleged denial of service attacks against the National Election Commission (NEC) website.

Police have arrested the 27-year-old personal assistant of ruling Grand National Party politician Choi Gu-sik over the alleged cyber-assault, which disrupted a Seoul mayoral by-election back in October.

However, security experts said that they doubt the suspect, identified only by his surname "Gong", had the technical expertise or resources needed to pull off the sophisticated attack.

Gong continues to protest his innocence, a factor that has led opposition politicians to speculate that he is covering up for higher-ranking officials who ordered the attack.

Democratic Party politician Baek Won-woo told *The HankYoreh*: “We need to determine quickly and precisely whether there was someone up the line who ordered the attack, and whether there was compensation.” ®

Russia accused of unleashing cyberwar to disable Estonia

- Parliament, ministries, banks, media targeted
- Nato experts sent in to strengthen defences

Ian Traynor in Brussels
The Guardian, Thursday 17 May 2007
Article history

Bronze Soldier, the Soviet war memorial removed from Tallinn. Nisametdinov/AP

A three-week wave of massive cyber-attacks on the small Baltic country of Estonia, the first known incidence of such an assault on a state, is causing alarm across the western alliance, with Nato urgently examining the offensive and its implications.

August 11th, 2008

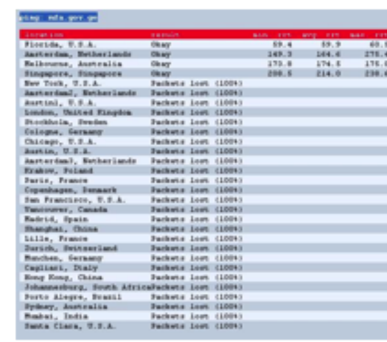
Coordinated Russia vs Georgia cyber attack in progress

Posted by Dancho Danchev @ 4:23 pm

Categories: Black Hat, Botnets, Denial of Service (DoS), Governments, Hackers...
Tags: Security, Cyber Warfare, DDoS, Georgia, South Osetia...

62 TalkBacks ADD YOUR OPINION | SHARE | PRINT | E-MAIL | WORTHWHILE? +18 24 VOTES

In the wake of the Russian-Georgian conflict, a week worth of speculations around Russian Internet forums have finally materialized into a coordinated cyber attack against Georgia's Internet infrastructure. The attacks have already managed to compromise several government web sites, with continuing DDoS attacks against numerous other Georgian government sites, prompting the government to switch to hosting locations to the U.S, with Georgia's Ministry of Foreign Affairs undertaking a desperate step in order to disseminate real-time information by moving to a Blogspot account



GitHub hit by Massive DDoS Attack From China

Friday, March 27, 2015 Mohit Kumar

8+1 249 | Like 3k | Share 2735 | Tweet 508 | Reddit 298 | Share 20 | ShareThis 4918

Github – a popular coding website used by programmers to collaborate on software development – was hit by a large-scale ***distributed denial of service (DDoS) attack*** for more than 24 hours late Thursday night.

It seems like when users from outside countries visit different websites on the Internet that serve advertisements and tracking code from Chinese Internet giant ***Baidu***, the assailants on Chinese border quietly inject malicious JavaScript code into the pages of those websites.

Credit: http://thehackernews.com/

Most Significant Operational Threats

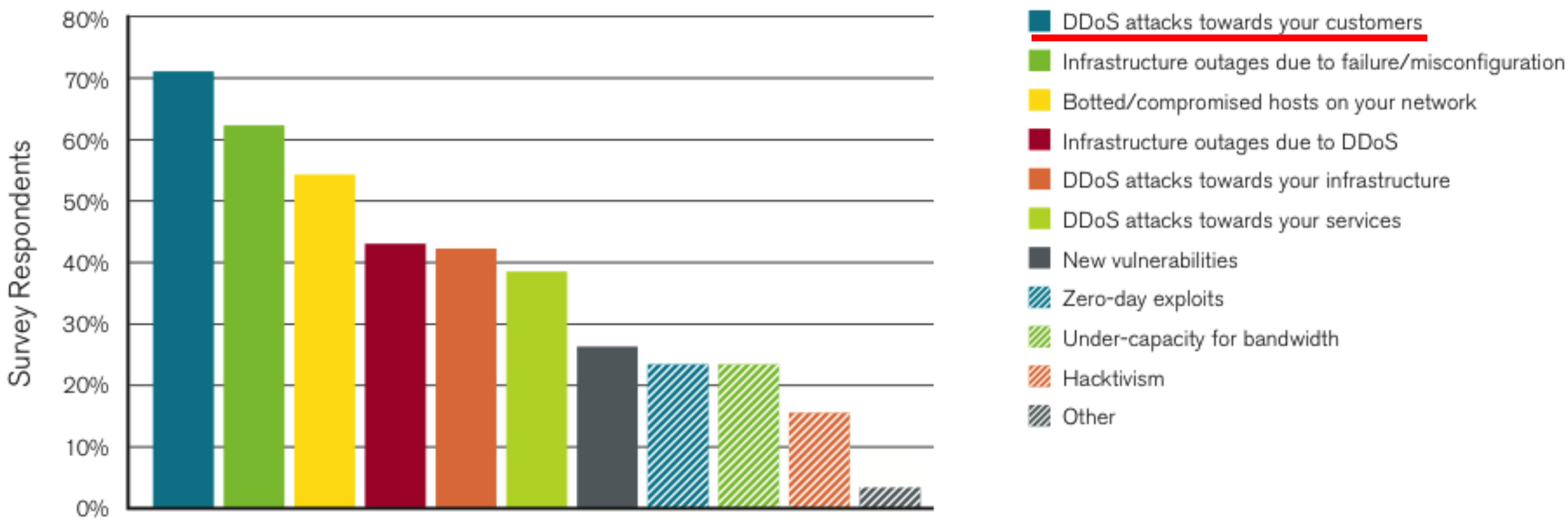
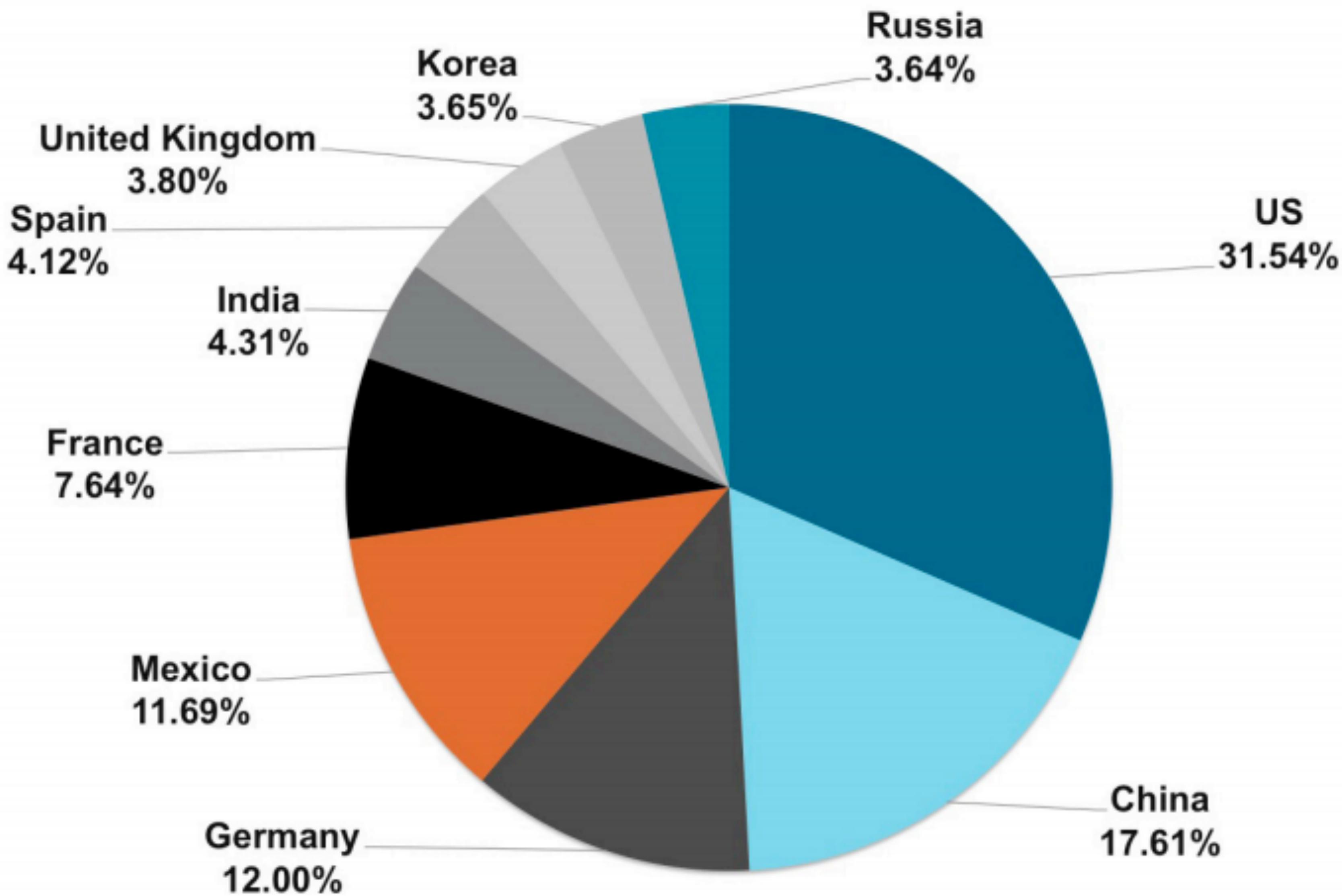


Figure 6 *Source: Arbor Networks, Inc.*

Top 10 source countries for DDoS attacks in Q4 2014

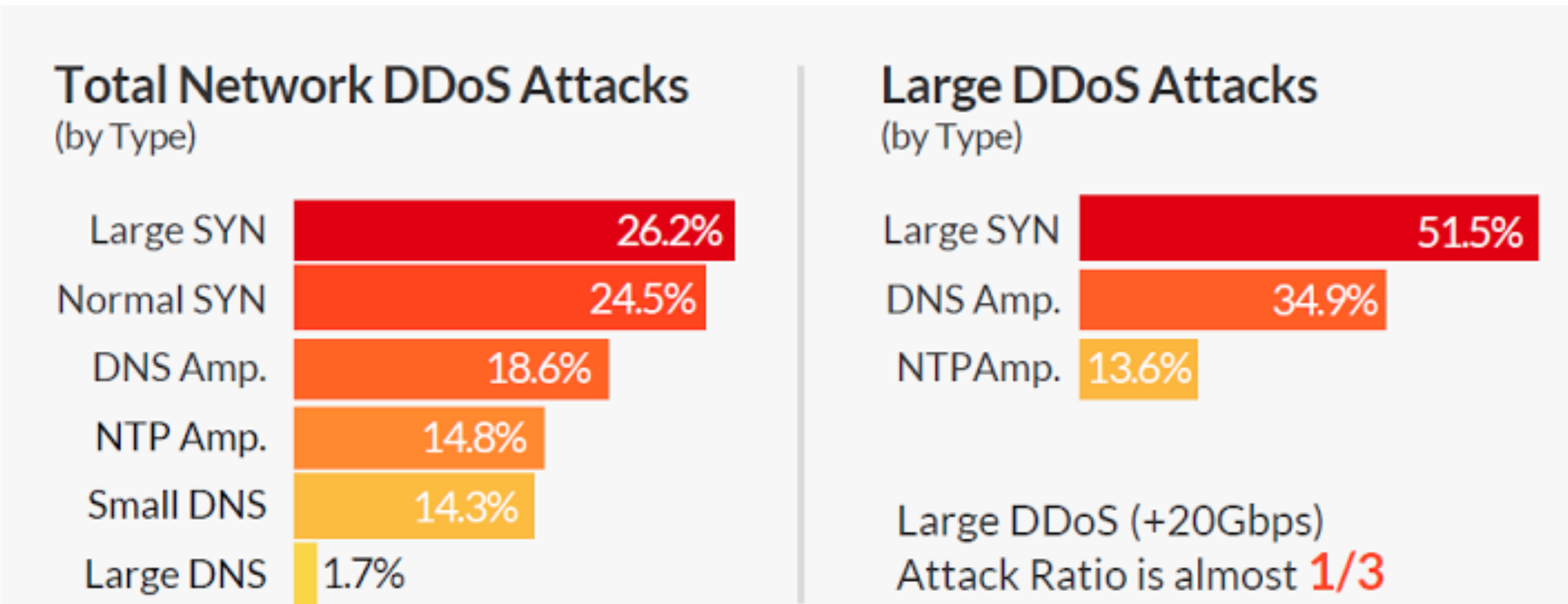


Credit: Akamai, The State of the Internet Q4 2014

Over 20Gbps DDoS attacks Now Become Common for Hackers

Sunday, March 30, 2014 Swati Khandelwal

The report site as "DDOS Threat Landscape", explains that almost one in every three DDoS attacks is above 20Gbps and 81% of attacks feature multiple vector threats.



Credit: http://thehackernews.com/

Motivations for DoS

- Showing off / entertainment / ego
- Competitive advantage
  - Maybe commercial, maybe just to win
- Vendetta / denial-of-money
- Extortion
- Political statements
- Impair defenses
- Espionage
- Warfare

Denial-of-Service Attacks

- Types of DoS
  - Network-level DoS
  - Application-level DoS
- DDoS: Distributed Denial-of-Service
- Amplification is key:
  - Traffic volume amplification
    - E.g. third parties amplify traffic
  - Computation resource amplification
    - E.g. memory consumption, CPU cycles

Network-level DoS

- How could you DoS a target's Internet access?
  - Flooding with lots of packets (brute-force)
  - DDoS: flood with packets from many sources
  - Amplification: Abuse patsies who will amplify your traffic
- What resources does attacker need?
  - At least as much sending capacity ("bandwidth") as the bottleneck link of the target's Internet connection
    - Attacker sends maximum-sized packets
  - Or: overwhelm the rate at which the bottleneck router can process packets
    - Attacker sends minimum-sized packets to maximize packet arrival rate

Defending Against Network DoS

- Suppose an attacker has high bandwidth (a “big pipe”)
- It sends packets to the target at a high rate
- How can the target defend against onslaught?
  - Install a network filter to discard any packets that arrive with attacker’s IP address as their source
    - E.g., `drop * 66.31.1.37:* -> *:*`
    - Or it can leverage any other pattern in the flooding traffic that’s not in benign traffic
    - Attacker’s IP address = means of identifying misbehaving user

Filtering sounds pretty easy…

… but DoS filters can be easily evaded

- Make traffic appear as though it's from many hosts
  - Spoof the source address so it can't be used to filter
    - Just pick a random 32-bit number of each packet sent
  - How does a defender filter this?
    - They don't!
    - Best they can hope for is that operators around the world implement anti-spoofing mechanisms (today about 75% do)

- Use many hosts to send traffic rather than just one
  - Requires defender to install complex filters
  - How many hosts is "enough" for the attacker?
    - Today they are very cheap to acquire … :-(

It’s not a “level playing-field”

- Asymmetries allow attackers to consume victim resources with little comparable effort
  - Makes DoS easier to launch
  - Defense costs much more than attack

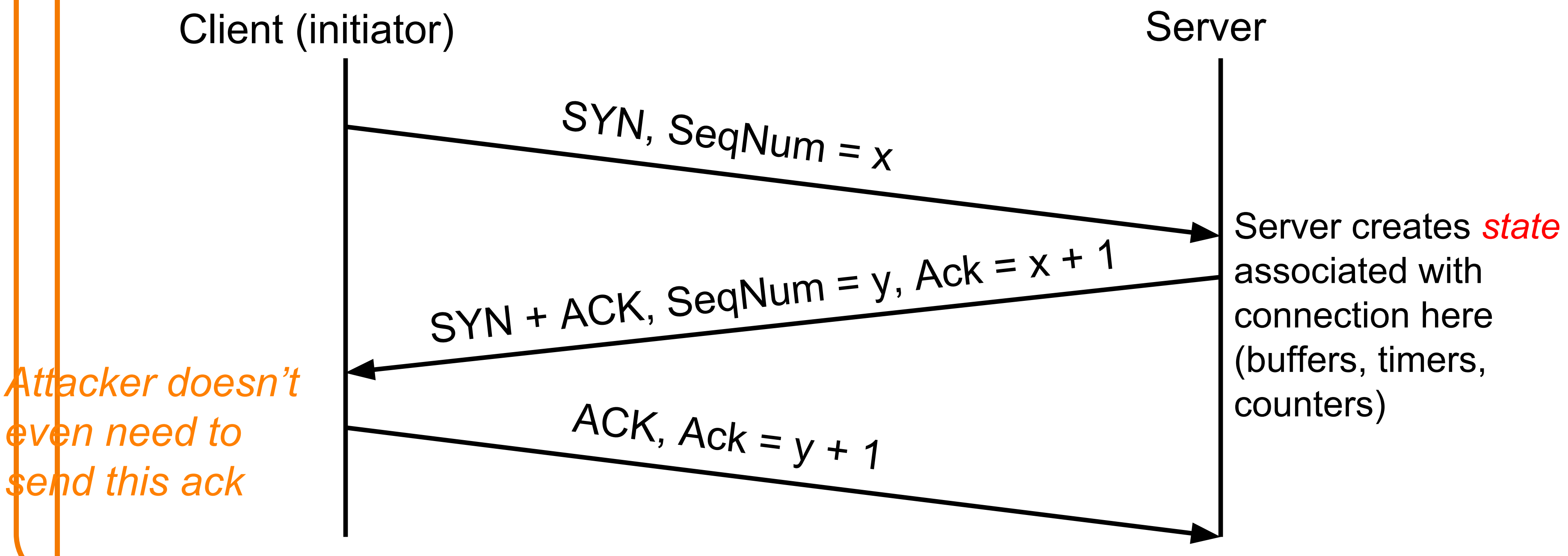
- Particularly dangerous form of asymmetry: amplification
  - Attacker leverages third party resources to increase workload

DNS Amplification

- Amplification example: DNS lookups
- Attacker spoofs DNS request from open DNS servers, seemingly from the target
  - Small attacker packets yield large flooding packets
    - Since the reply includes a copy of the query, plus the answer, etc
- Note #1: these examples involve blind spoofing
  - So for network-layer flooding, generally only works for UDP-based protocols (can't establish TCP conn.)
- Note #2: victim doesn't see spoofed source addresses
  - Addresses are those of actual intermediary systems

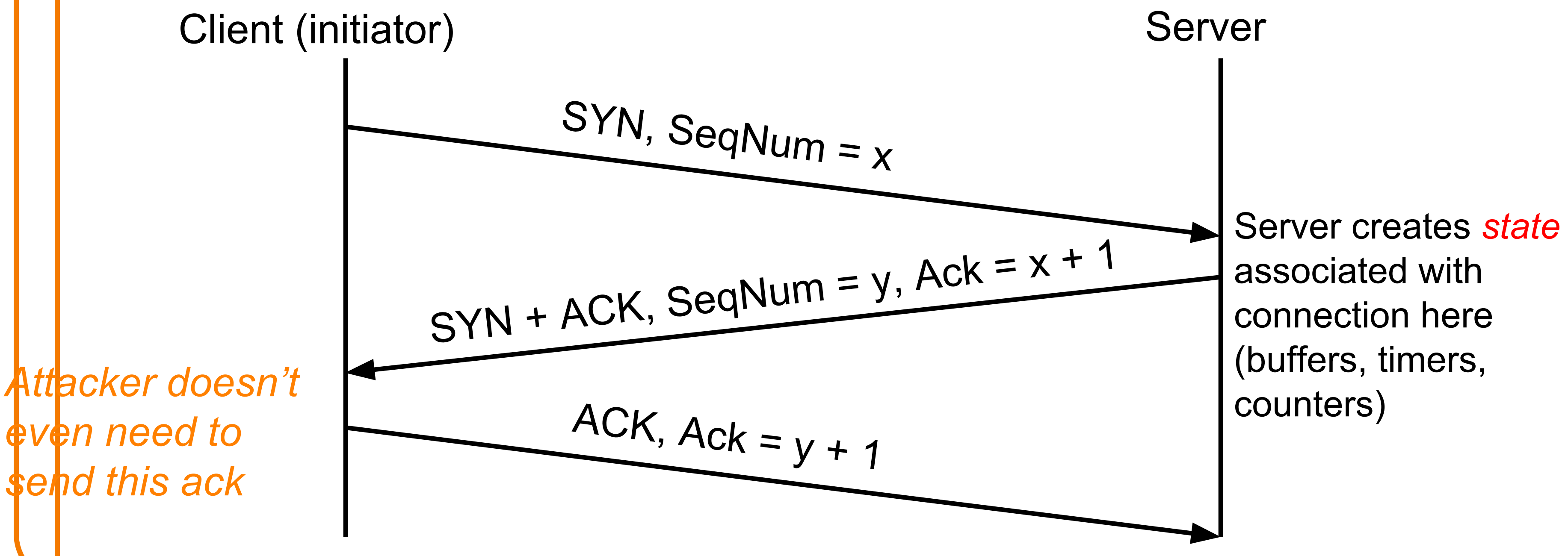
Transport-Level Denial-of-Service

- Recall TCP's 3-way connection establishment handshake
  - Goal: agree on initial sequence numbers



Transport-Level Denial-of-Service

- Recall TCP's 3-way connection establishment handshake
  - Goal: agree on initial sequence numbers
- So a single SYN from an attacker suffices to force the server to spend some memory



TCP *SYN Flooding*

- Attacker targets memory rather than network capacity
- Every (unique) SYN that the attacker sends burdens the target
- What should target do when it has no more memory for a new connection?
  - No good answer
  - Refuse new connection?
    - Legit new users can't access service
  - Evict old connections to make room?
    - Legit old users get kicked off

TCP SYN Flooding Defense

- How can the target defend itself?

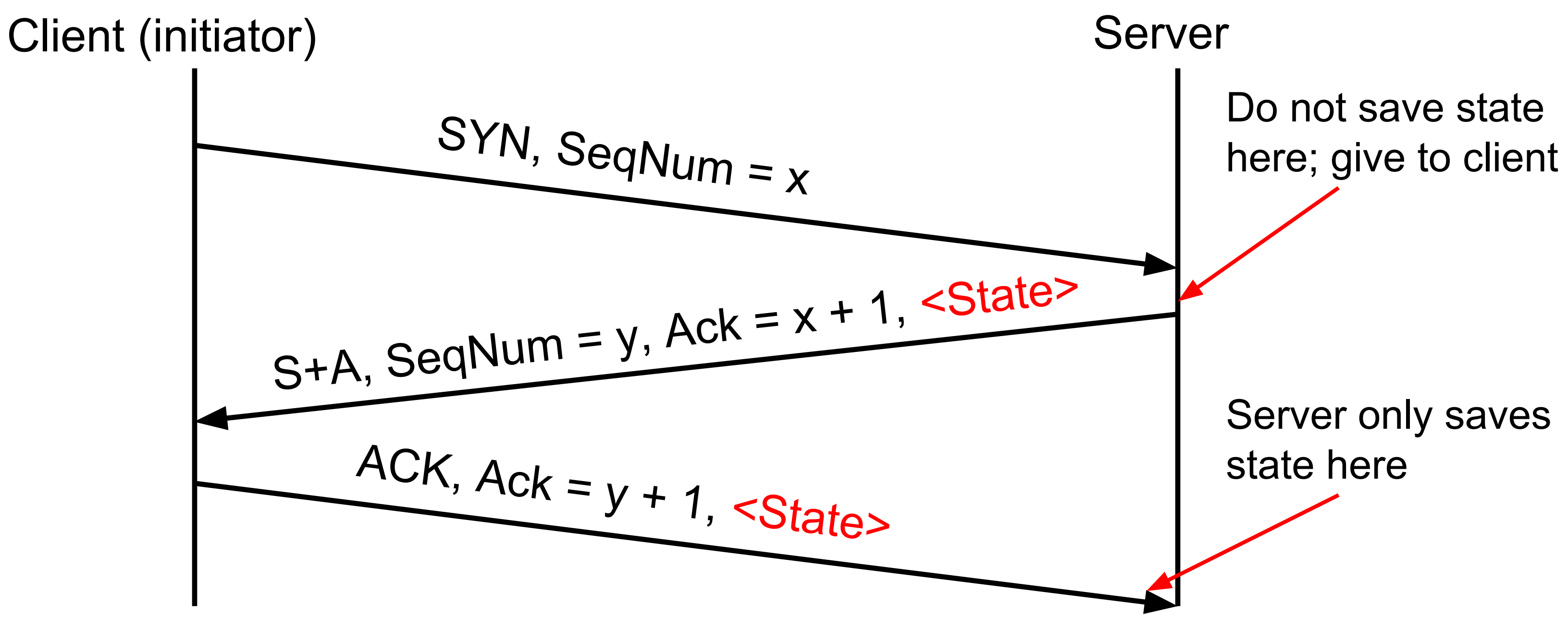
- Approach #1: tons of memory
  - How much is enough?
  - Depends on resources attacker can bring to bear (threat model), which might be hard to know

TCP SYN Flooding Defense

- Approach #2: identify bad actors & refuse connections
  - Hard because identification is on IP address
    - We cannot require a password because doing so requires an established connection!
  - For a public Internet service, who knows which addresses customers might come from?
  - Plus: attacker can spoof addresses since they don't need to complete TCP 3-way handshake
- Approach #3: don't keep state!
  - "SYN cookies"; only works for spoofed SYN flooding

SYN Flooding Defense: Idealized

- Server: when SYN arrives, rather than keeping state locally, send it to the client …
- Client needs to return the state in order to established connection



SYN Flooding Defense: Idealized

- Server: when SYN arrives, rather than keeping state locally, send it to the client
- Client
  establ

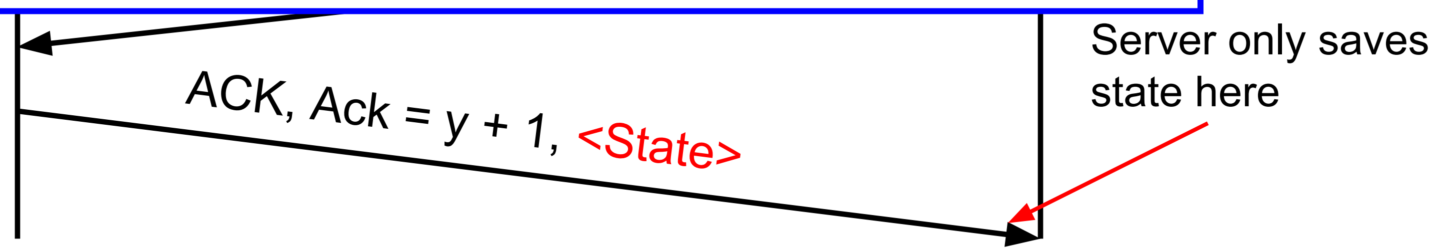
Client (

t save state
give to client

Problem: the world isn't so ideal!

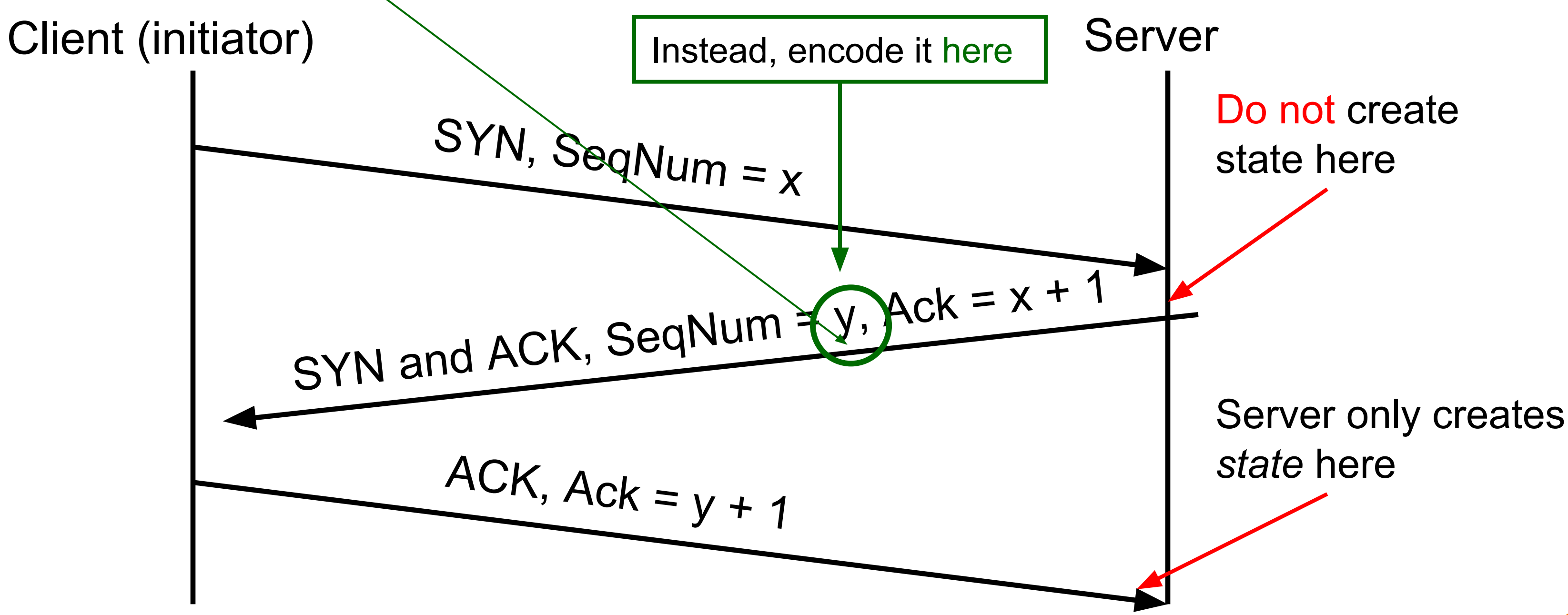
TCP doesn't include an easy way to add a new **<State>** field like this.

Is there any way to get the same functionality without having to change TCP clients?



Practical Defense: SYN Cookies

- Server: when SYN arrives, encode connection state entirely within SYN-ACK's sequence # y
  - y = encoding of necessary state, using server secret
- When ACK of SYN-ACK arrives, server only creates state if value of y from it agrees w/ secret



SYN Cookies: Discussion

- Illustrates general strategy: rather than holding state, encode it so that it is returned when needed
- For SYN cookies, attacker must complete 3-way handshake in order to burden server
  - Can't use spoofed source addresses
- Note #1: strategy requires that you have enough bits to encode all the state
  - (This is just barely the case for SYN cookies)
- Note #2: if it's expensive to generate or check the cookie, then it's not a win

Application-Layer DoS

- Rather than exhausting network or memory resources, attacker can overwhelm a service's processing capacity
- There are many ways to do so, often at little expense to attacker compared to target (asymmetry)

The link sends a request to the web server that requires heavy processing by its “backend database”.

Algorithmic complexity attacks

- Attacker can try to trigger worst-case complexity of algorithms / data structures
- Example: You have a hash table.
  Expected time: $O(1)$ Worst-case: $O(n)$
- Attacker picks inputs that cause hash collisions.
  Time per lookup: $O(n)$
  Total time to do n operations: $O(n^2)$
- Solution? Use algorithms with good worst-case running time.

Application-Layer DoS

- Rather than exhausting network or memory resources, attacker can overwhelm a service's processing capacity
- There are many ways to do so, often at little expense to attacker compared to target (asymmetry)
- Defenses against such attacks?
- Approach #1: Only let legit users issue expensive requests
  - Relies on being able to identify/authenticate them
  - Note: that this itself might be expensive!
- Approach #2: Force legit users to "burn" cash
- Approach #3: Over-provisioning ($$$)

DoS Defense in General Terms

- Defending against program flaws requires:
  - Careful design and coding/testing/review
  - Consideration of behavior of defense mechanisms
    - E.g. buffer overflow detector that when triggered halts execution to prevent code injection ⇒ denial-of-service
- Defending resources from exhaustion is hard. Requires:
  - Isolation and scheduling mechanisms
    - Keep adversary's consumption from affecting others
  - Reliable identification of different users

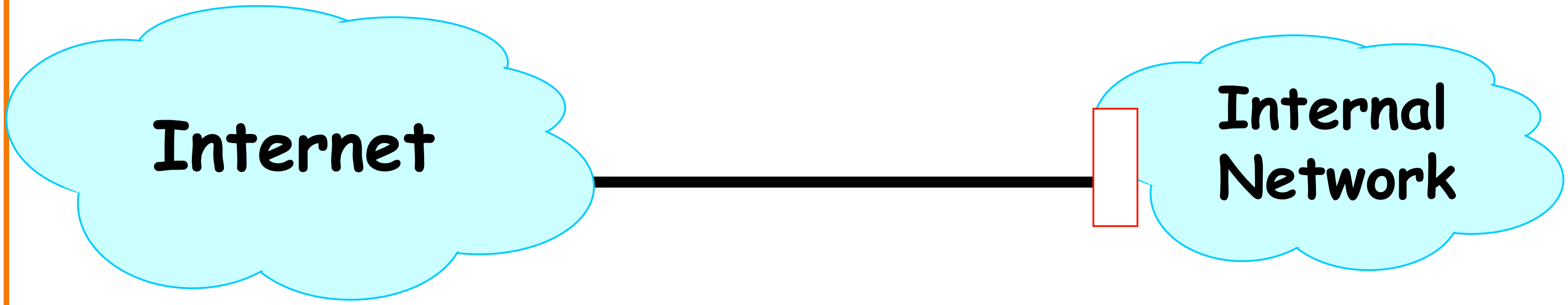
Firewalls

Firewalls

- Harden set of systems against external attack
- More network services → greater risk
  - Larger attack surface
- Can turn off unnecessary services
  - Requires knowledge of all services running
  - Sometimes trusted users require access
- Scaling issues
  - Hundreds/thousands of systems
  - Many different operating systems, hardware, users

Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking in the network outsiders from having unwanted access your network services
  - Interpose a firewall the traffic to/from the outside must traverse
  - Chokepoint can cover thousands of hosts
    - Where in everyday experience do we see such chokepoints?



Selecting a Security Policy

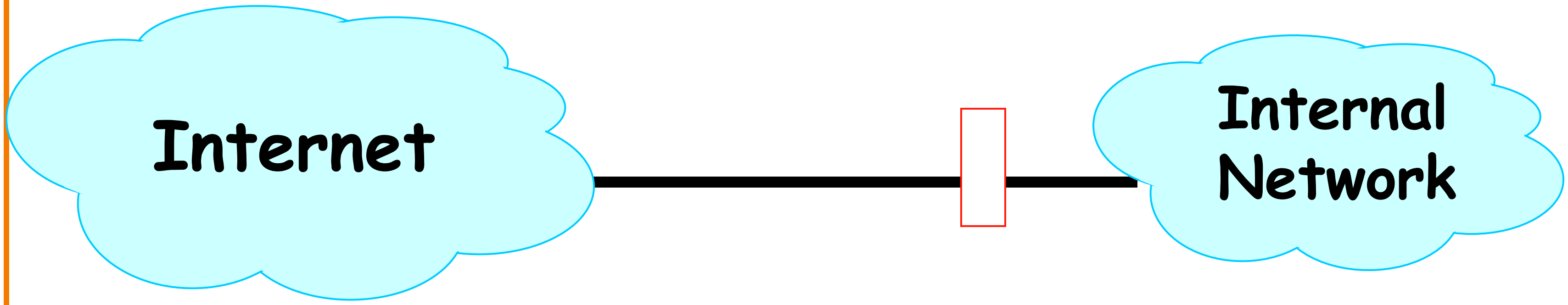
- Firewall enforces an (access control) policy:
  - Who is allowed to talk to whom, accessing what service?
- Distinguish between inbound & outbound connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common threat model. There are thousands of internal users (and we've vetted them). There are billions of outsiders.
- Conceptually simple access control policy:
  - Permit inside users to connect to any service
  - External users restricted:
    - Permit connections to services meant to be externally visible
    - Deny connections to services not meant for external access

Default policies

- Default allow
  - Begin by permitting external access to services
  - Turn off as problems recognized
- Default deny
  - Begin by denying external access to services
  - Turn on access on case-by-case basis
- Generally we use default deny
  - Flexibility vs conservative design
  - Flaws in default deny are noticed more quickly (less painfully)

Stateful Packet Filter

- Stateful packet filter is a router that checks each packet against security rules and decides to forward or drop it
  - Firewall keeps track of all connections (inbound/outbound)
  - Each rule specifies which connections are allowed/denied (*access control policy*)
  - A packet is forwarded if it is part of an allowed connection



Example rule

- Permits TCP connection that is
  - Initiated by host 4.5.5.4
  - Connecting to port 80 of host 3.1.1.2
- Permits any packet (*) associated with connection
- Firewall keeps table of allowed active connections
  - Checks traffic against table

```
allow tcp connection 4.5.5.4:* -> 3.1.1.2:80
```

Example rule

- Permits TCP connection that is
  - Initiated by any internal host (`*:*`)
  - Connecting to port 80 of 3.1.1.2 on external network
- Permits any packet (`*`) associated with connection
- `/in` indicates network interface

```
allow tcp connection *:*/in -> 3.1.1.2:80/out
```

Example ruleset

- Permits all outbound TCP connections
  - Those initiated by internal hosts
- Permits inbound TCP connection to web server (port 80) at IP address 1.2.2.3

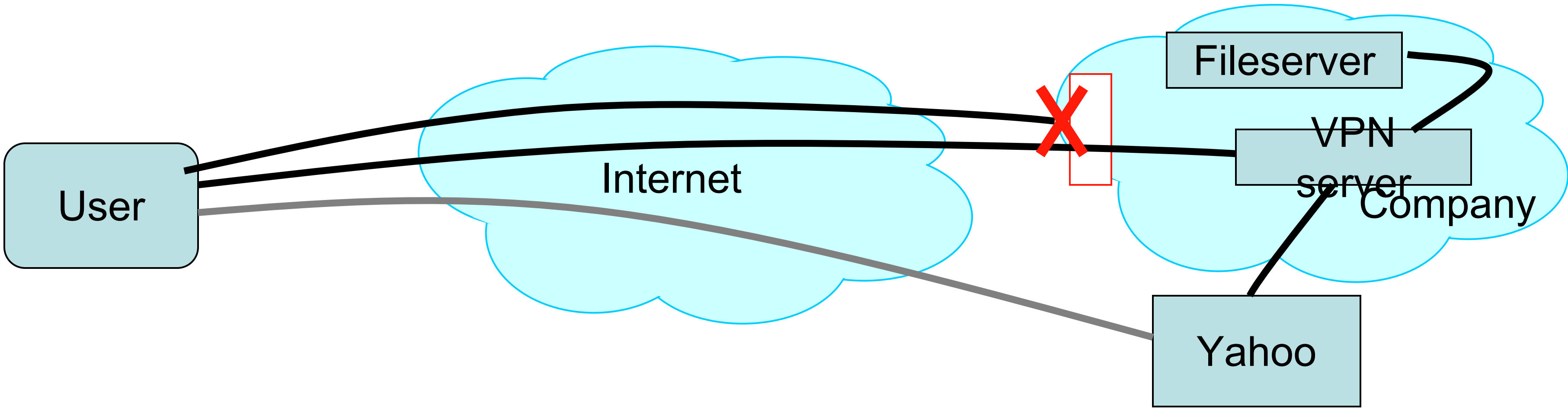
```
allow tcp connection *:*/in -> *:*/out
allow tcp connection *:*/out -> 1.2.2.3:80/in
```

Example ruleset

```
allow tcp connection *:*/in -> *:*/out
allow tcp connection *:*/out -> 1.2.2.3:80/in
```

- Firewall should permit outbound TCP connections (i.e., those that are initiated by internal hosts)
- Firewall should permit inbound TCP connection to our public webserver at IP address 1.2.2.3

Secure External Access to Inside Machines



- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create secure channel (Virtual Private Network, or VPN) to tunnel traffic from outside host/network to inside network
  - Provides Authentication, Confidentiality, Integrity
  - However, also raises perimeter issues

    (Try it yourself at http://www.net.berkeley.edu/vpn/)

Firewall Advantages

- Central control – easy administration and update
  - Single point of control: update one config to change security policies
  - Potentially allows rapid response
- Easy to deploy – transparent to end users
  - Easy incremental/total deployment to protect 1000's
- Addresses an important problem
  - Security vulnerabilities in network services are rampant
  - Easier to use firewall than to directly secure code …

Firewall Disadvantages

- Functionality loss – less connectivity, less risk
  - May reduce network's usefulness
  - Some applications don't work with firewalls
    - Two peer-to-peer users behind different firewalls
- The malicious insider problem
  - Assume insiders are trusted
    - Malicious insider (or anyone gaining control of internal machine) can wreak havoc
- Firewalls establish a security perimeter
  - Like Eskimo Pies: "hard crunchy exterior, soft creamy center"
  - Threat from travelers with laptops, cell phones, …