CSE 127 Computer Security

Stefan Savage, Spring 2020, Lecture 14

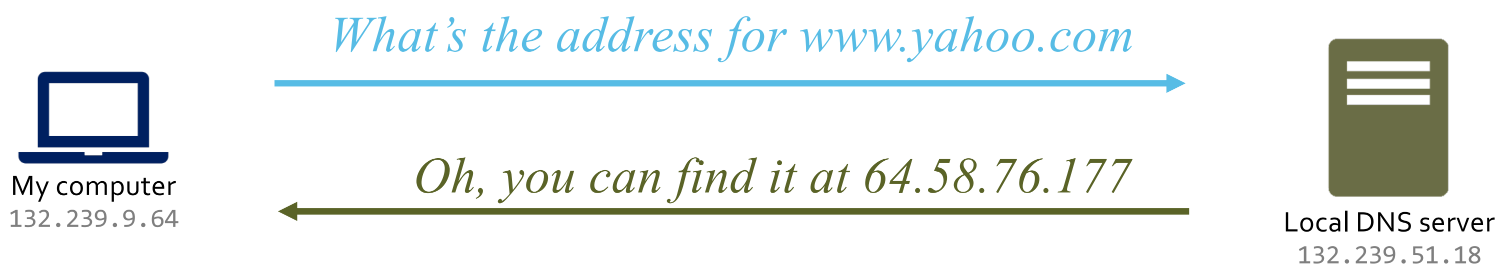
Network Security II: DNS, Denial-of-service and perimeter defense

Today

- The Domain Name System
  - Another place where names at different layers can bound
  - The target of quite a few attacks
- Denial of service
  - Network attacks on availability
- Network perimeter defenses
  - Firewalls, NATs, NIDS/NIPS

Remember this? How did this work?

- Where is www.yahoo.com?



Ignore for now:
How did you know the address of the Local DNS server?
How did you send a message to it?
How did the Local DNS server know the answer?

Domain Name System

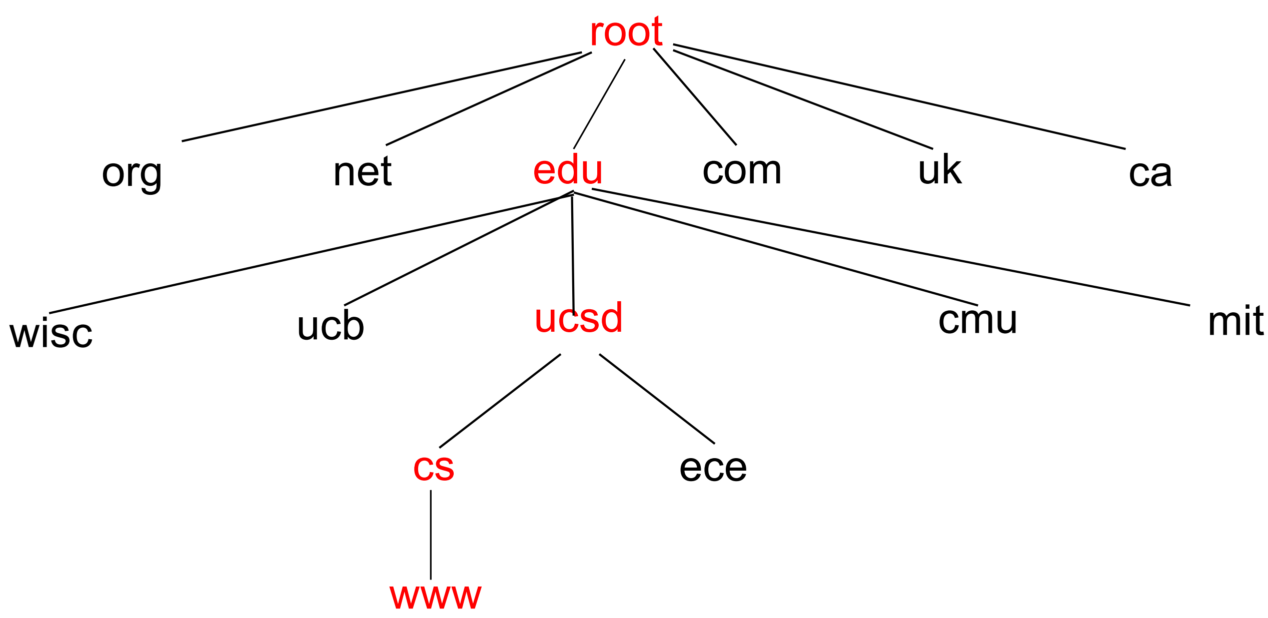
- We humans do not tend to remember 32bit IP numbers…

- Solution: domain names
  - Human readable identifiers (e.g., www.cs.ucsd.edu, google.com, etc)

- Problem: how to map DNS names to IP addresses?
  - In the old days we had a big file – *literally* (download from sri-nic.arpa)
  - Today we use a distributed name servers called the Domain Name System (DNS)

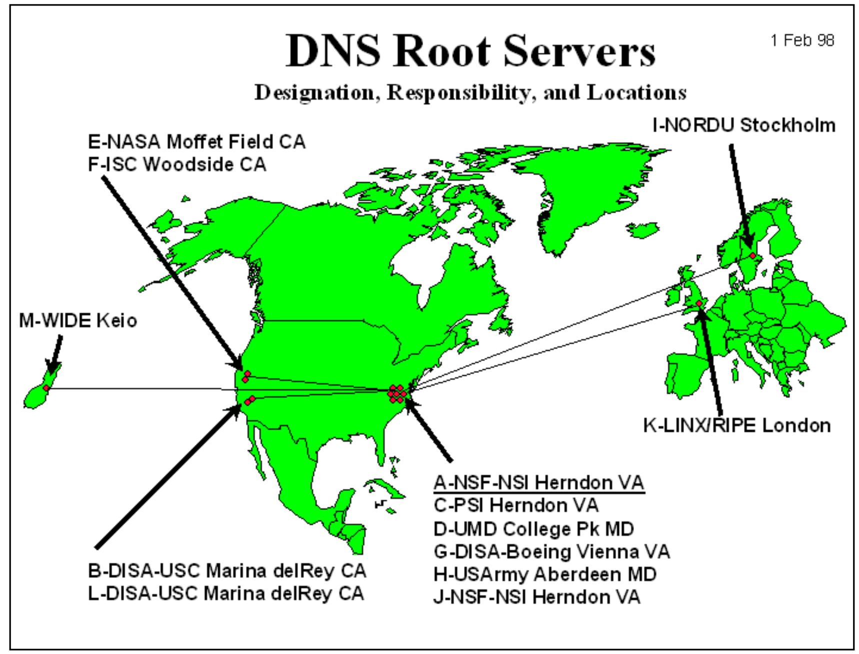
Domain Name System (DNS)

- Hierarchical Name Space

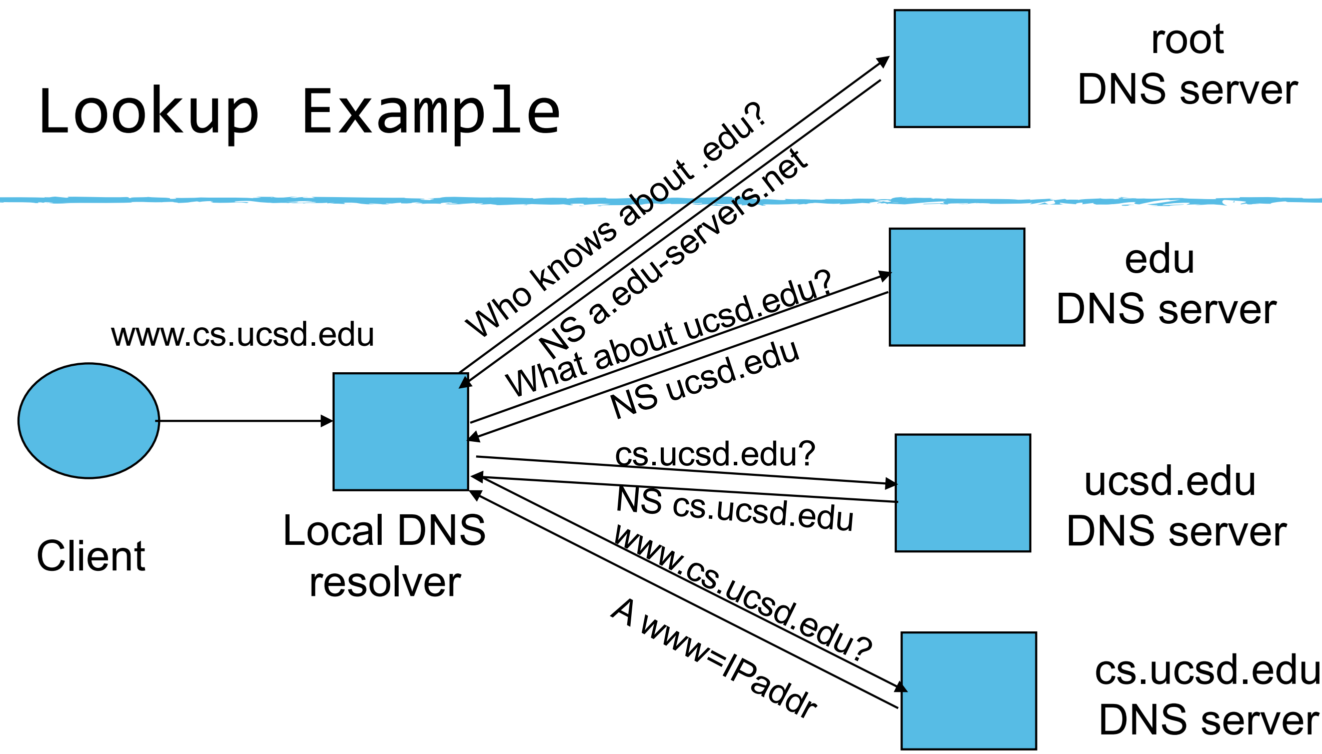


DNS Root Name Servers

- Hierarchical service
  - 13 root name servers for top-level domains
    - **Hardcoded into all systems**
    - Choose one at random
  - Authoritative name servers for subdomains
  - Local name resolvers (also called recursive resolvers) contact authoritative servers when they do not know a name



DNS Lookup Example



DNS record types (partial list):

- NS: name server (points to other server)
- A: address record (contains IP address)
- MX: address in charge of handling email
- TXT: generic text (e.g. used to distribute site public keys (DKIM))

Caching

- DNS **responses are cached**
  - Quick response for repeated translations
  - Useful for finding servers as well as addresses
    - NS records for domains
- DNS **negative queries** are cached
  - Save time for nonexistent sites, e.g. misspelling
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record, delete cached entry after TTL expires

DNS cache poisoning

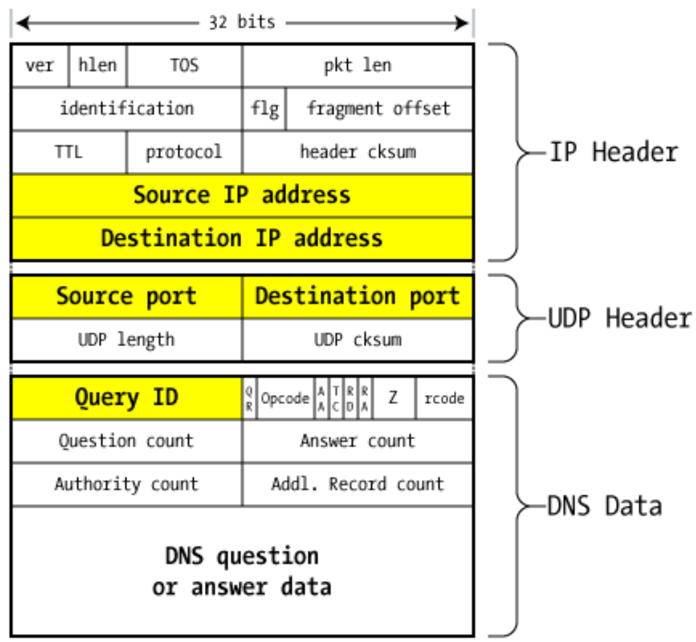
- Basic idea:
  - If I can convince a DNS resolver to cache a bad mapping (e.g., www.cs.ucsd.edu points to 127.0.0.1) then everyone who uses it for www.cs.ucsd.edu will get that incorrect resolution
  - Can then be used for fraud, man-in-the-middle attacks, etc
- Used in lots of attacks
  - January 2005, the domain name for a large New York ISP, Panix, was hijacked to a site in Australia.
  - In November 2004, Google and Amazon users were sent to Med Network Inc., an online pharmacy
  - In March 2003, a group dubbed the "Freedom Cyber Force Militia" hijacked visitors to the Al-Jazeera Web site and presented them with the message "God Bless Our Troops"
  - 2000 campaign: Hilary2000.org -> hilaryno.com

But how to do it?

- Man-in-the-middle attacker: easy
  - Observe DNS requests from server
  - Send false response to server and block true response
- Passive attacked: easy also
  - Observe DNS requests from server
  - Send false response to server before true response
- What about off-path attacker
  - If you know when user is likely to make request, you can flood responses to their server blindly
  - But there’s a problem – matching request and response

DNS Packet

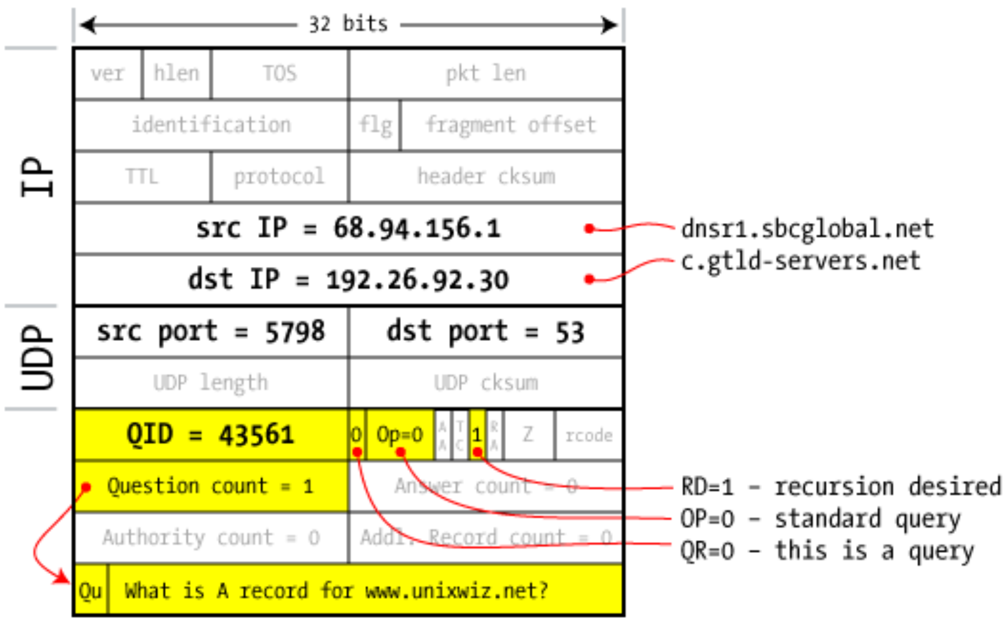
- Query ID:
  - 16 bit random value
  - Links response to query



(from Steve Friedl)

Resolver to NS request

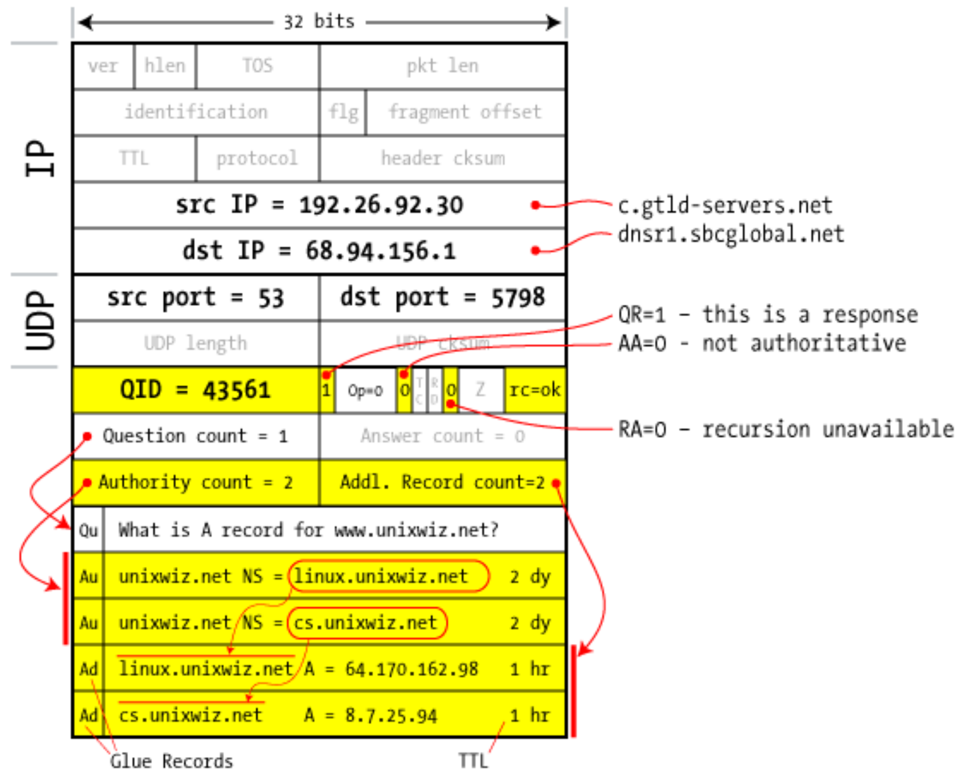
- Context
  - SBC Global customer looks up unixwiz.net
  - Goes to their DNS server dnsr1.sbcglobal.net
  - This is the request sent from that server to c.gtld-servers.net which can give answers for domains ending in .net



Response to resolver

Response contains IP addr of next NS server (called "glue")

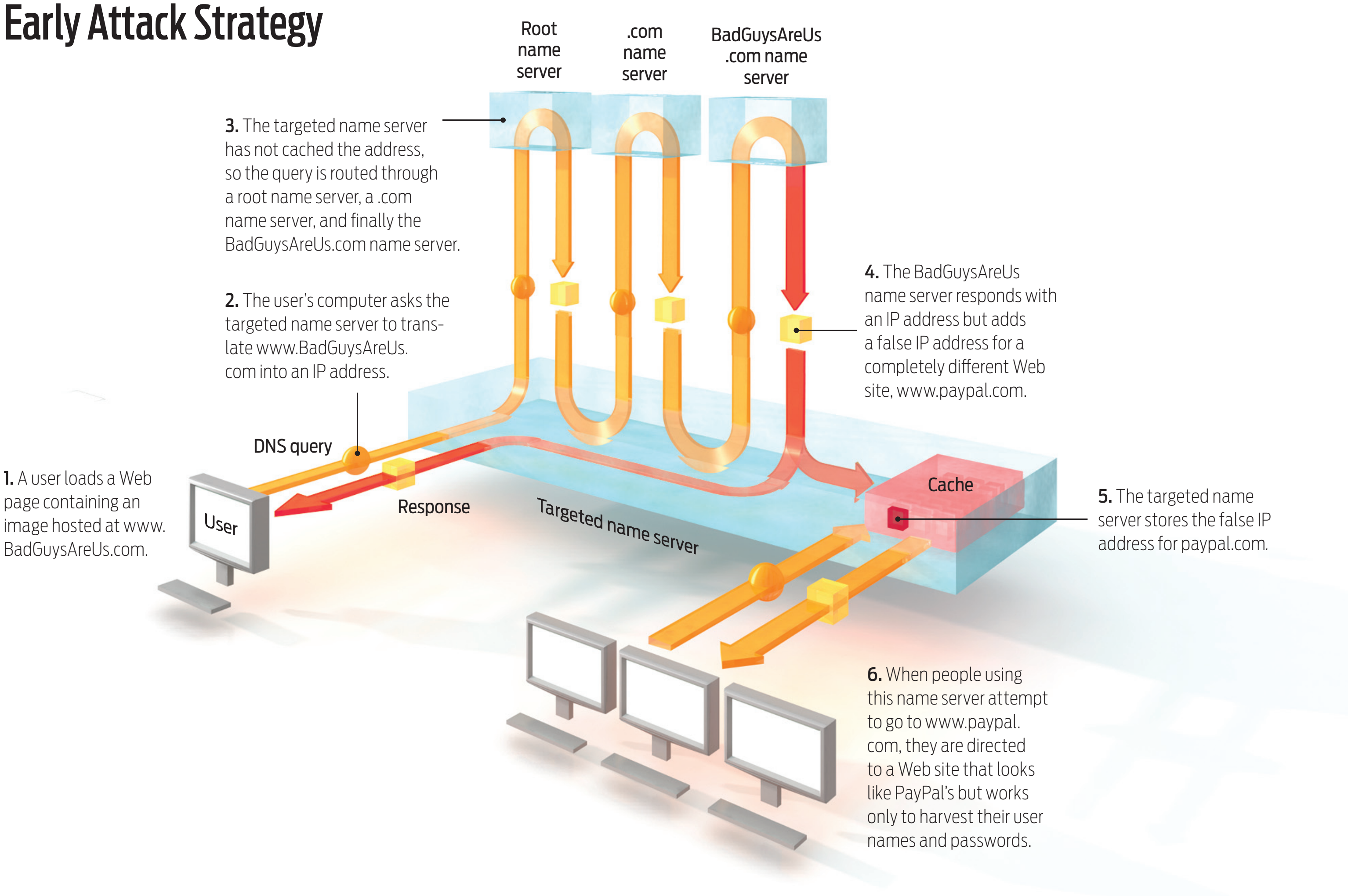
Response ignored if unrecognized QueryID



DNS additional section

- Answers to questions you didn't ask…
  - There is a good reason for it… if I tell you that the name server for foo.com is ns1.foo.com… how do you find its IP address?
- But this is a problem… what if I run a DNS server for foo.com and when I get asked for the IP address for bar.foo.com, I put some additional stuff in the "additional section" like:
  - You can find the IP address for paypal.com at 41.2.6.2 (address I control)
  - And you can also find the IP address for amazon.com and chase.com there too…

Early Attack Strategy

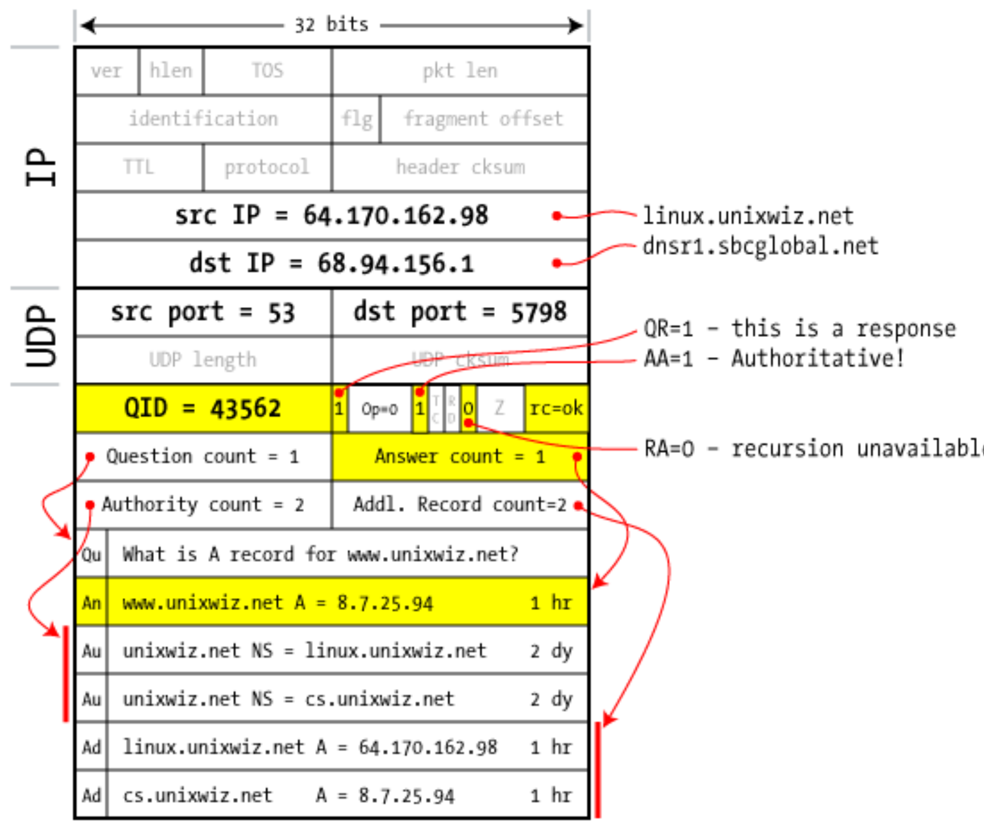


(so

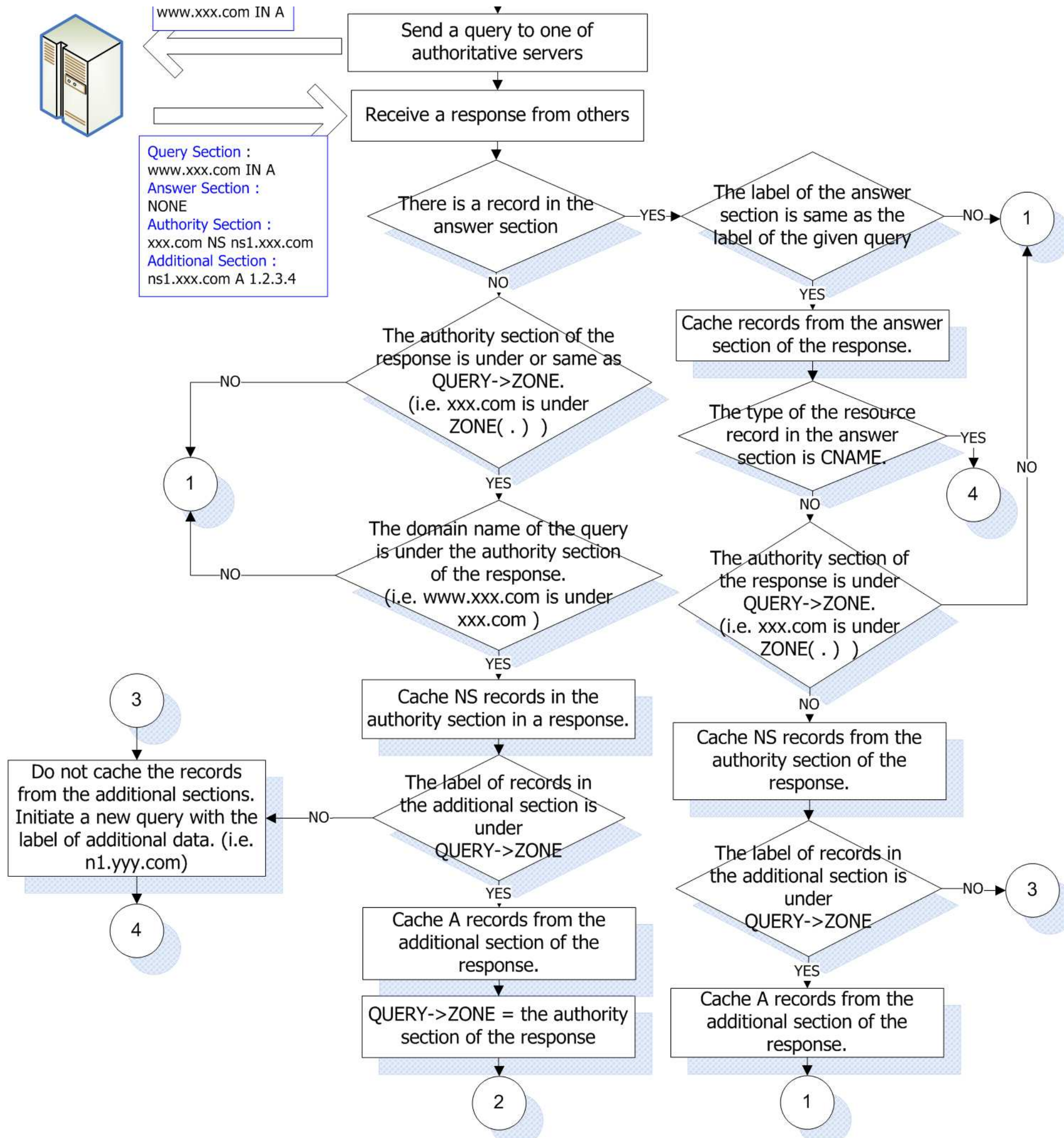
Authoritative response to resolver

<u>bailiwick checking:</u>
response is cached **only** if it is within the same domain of query
(i.e. a.com cannot set NS for b.com)

final answer



Bailiwick Checking Rule from BIND



source: Son and Shmatikov, "The H
DNS Cache Poisoning" SECURECO

But we forgot something

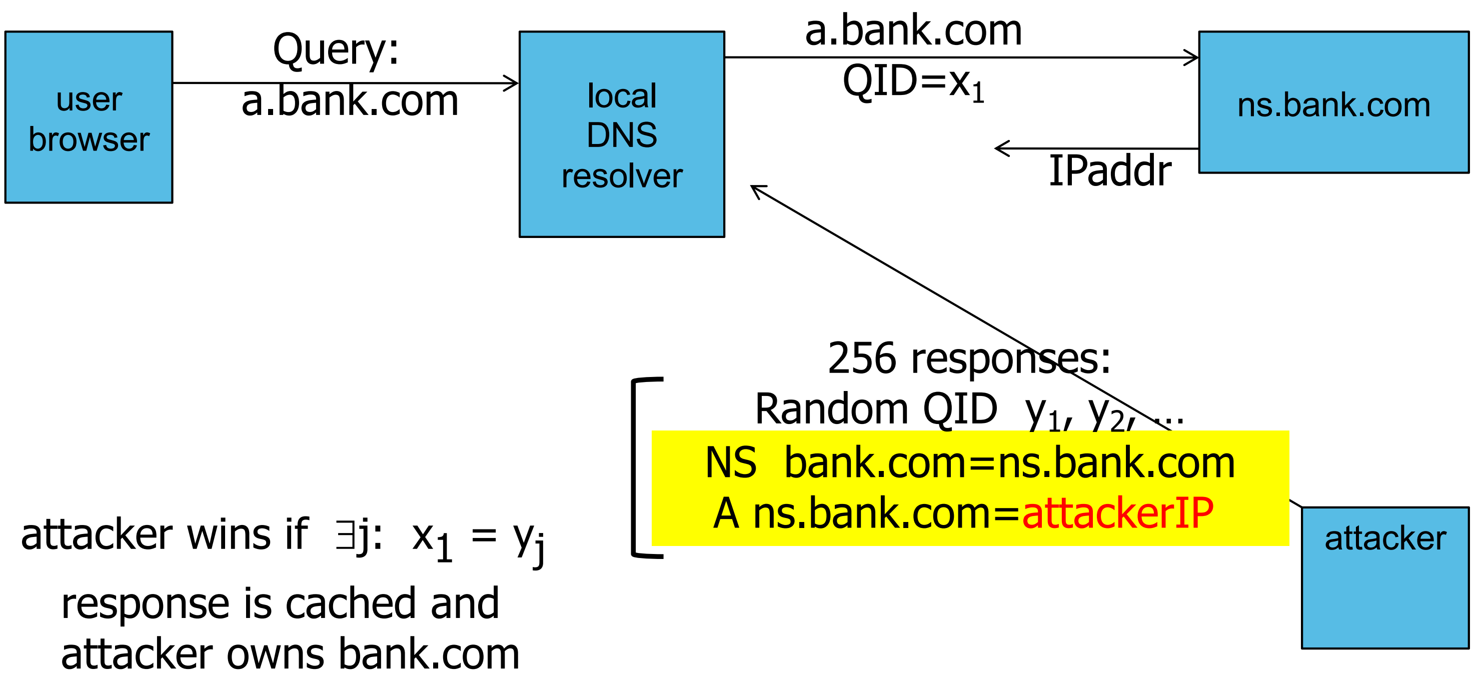
- A decade goes by and Dan Kaminsky realizes that the bailiwick checking rule doesn’t really protect us

- Unnoticed hole that allows arbitrary DNS poisoning at a distance

- To fix bug, ***unprecedented*** coordinated global operation to do secret mass migration of DNS infrastructure (2008)

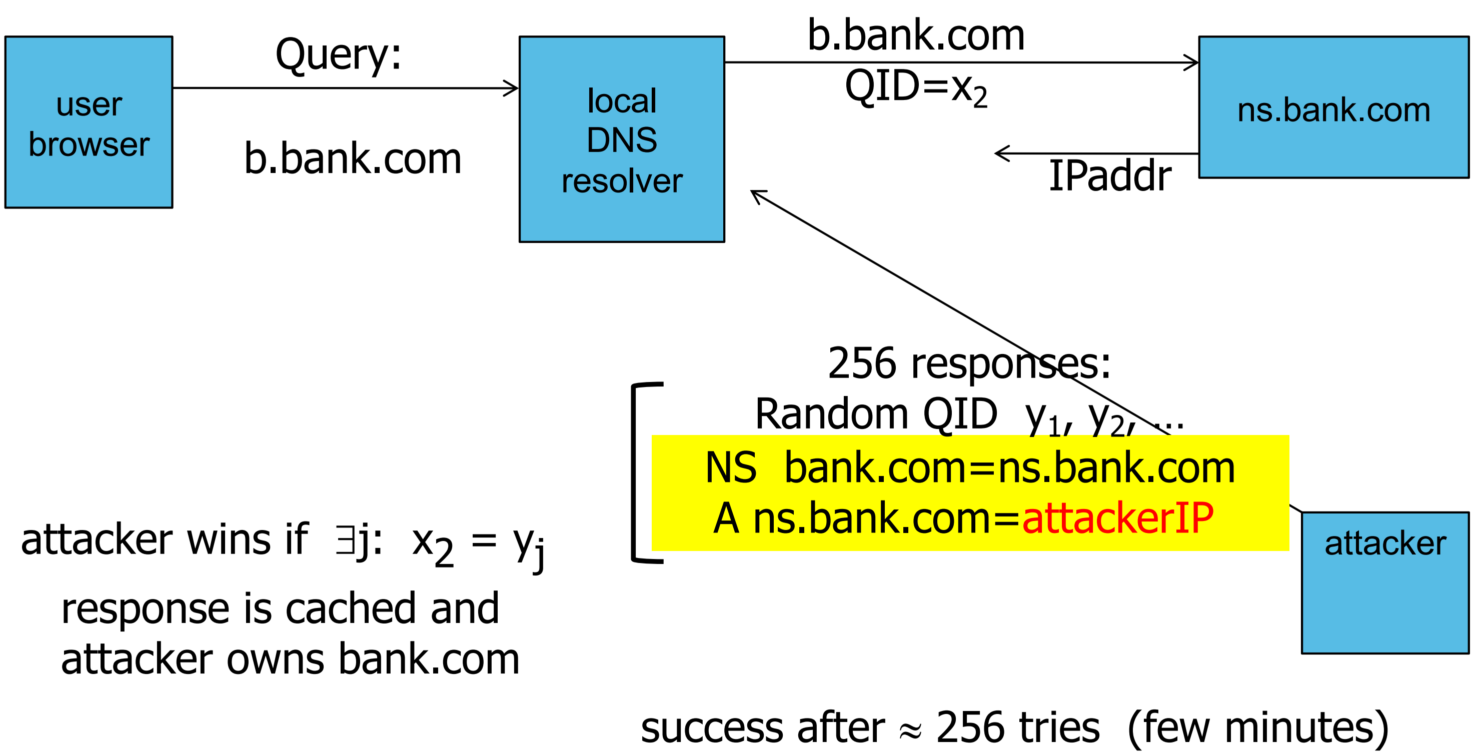
DNS cache poisoning (a la Kaminsky'08)

- Victim machine visits attacker's web site, downloads Javascript

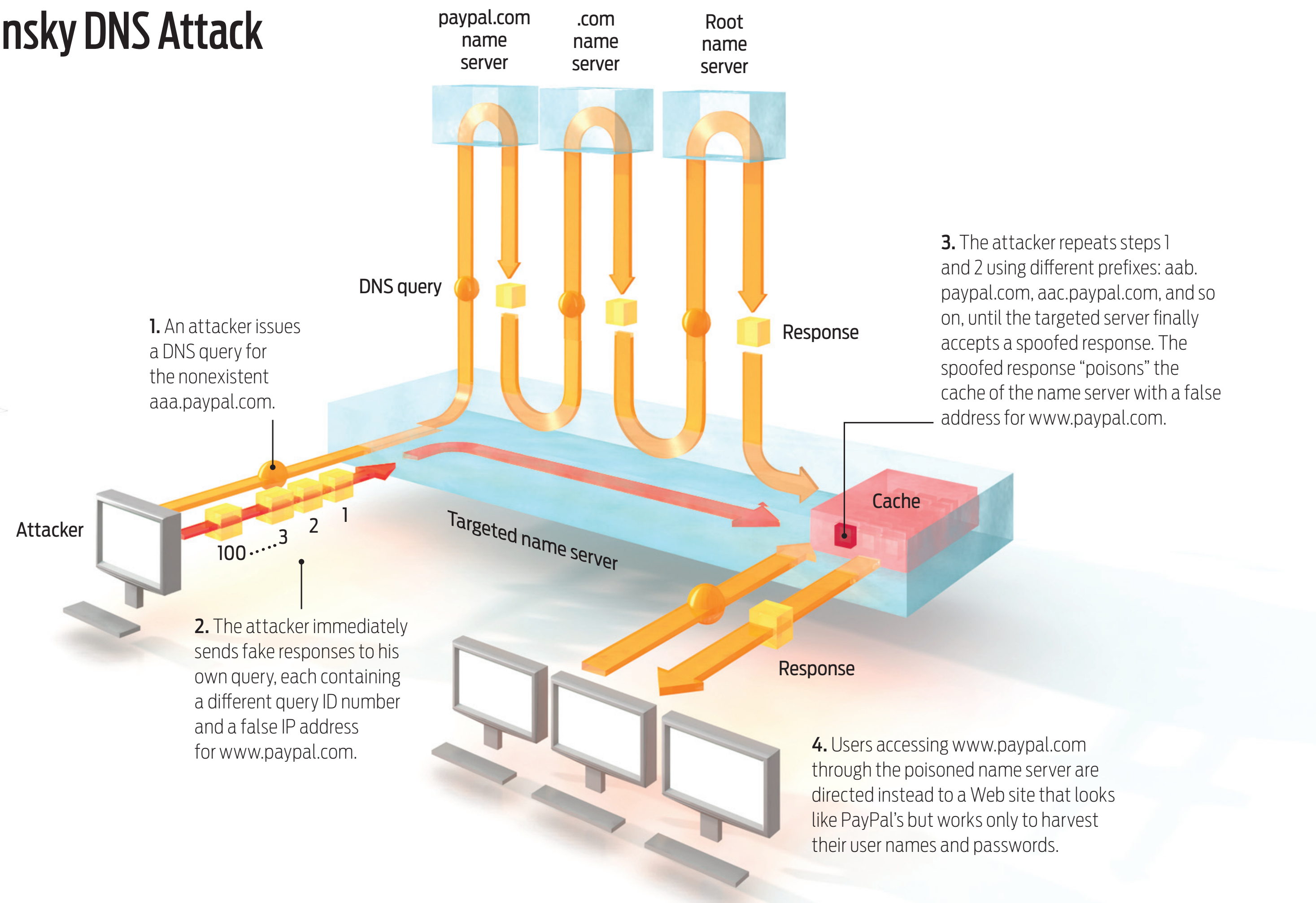


If at first you don't succeed …

- Victim machine visits attacker's web site, downloads Javascript



Kaminsky DNS Attack



BRYAN CHRISTIE DESIGN

Defenses

- **Increase Query ID size**. How? Some approaches in use
  - Randomize src port and match it, additional 11 bits
    - Now attack takes many hours
  - 0x20 encoding – randomly vary capitalization (DNS is case insensitive) check that you get same capitalization back

- Try to detect poisoning
  - Ignore responses not directly necessary to query

- Authenticated requests/responses
  - Provided by DNSsec (digital signatures on DNS records) … but few domains use DNSsec

DNS Summary

- Current DNS system does not provide strong evidence binding request to response
- Response can provide more data than was asked for
- Together allows attacker to “poison” DNS and divert traffic to their sites

- This is also why its so important for HTTPS to check certificate signatures (i.e., because just because you ask for www.amazon.com doesn’t mean you’ll get it)

Denial-of-service

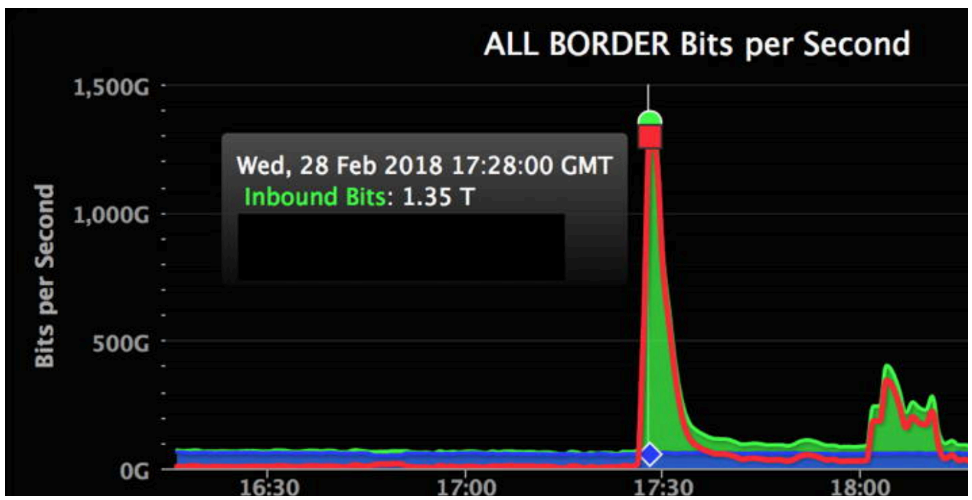
- Attack against *availability*, not confidentiality, integrity, authenticity, etc
- Two kinds of attacks:
  - **Logic vulnerabilities:** exploit bugs to cause crash
    - e.g. Ping-of-Death, Land
    - Fix via filtering and patching
  - **Resource consumption**: overwhelm with spurious requests
    - e.g. SYN flood, bandwidth overflow
    - Much tougher to fix…
    - Same idea, more quantity: Distributed denial-of-service attacks (DDOS)
      - **Lots of hosts** attack a victim at once

Resource consumption of Service

- Server CPU/Memory resources
  - Consumes connection state (e.g. SYN flood)
  - Time to evaluate messages (interrupt livelock)
    - Some messages take "slow path" (e.g. invalid ACK)
  - Can cause new connections to be dropped and existing connections to time-out
  - Make DB process lots of queries
    - Attack cache – lots of random queries
- Network resources
  - Some routers are packet-per-second limited, FIFO queuing (send small pkts)
  - If attack is greater than forwarding capacity, good data will be dropped (send big pkts)

This is a widespread problem

- Most attacks small and focused on individuals or small sites
  - Particularly gamers; so-called booter services
- But some attacks are huge (e.g., 2018 Attack on GitHub)



What to do?

- Defenses against address spoofing
  - For attacks from randomly spoofed addresses
- Filtering based on attack features or IP address
- Buy more resources

Address spoofing

- Filter packets with incorrect source addresses
  - **Network egress**: filter outbound packets on a link whose source addresses are not reached using the link as the next hop (i.e., this couldn't be your source address)
    - But requires network routers to do "good deeds" for others…
  - **Network ingress**: filter inbound packets whose source address are not in the routing table at all
- SYN Cookies
  - Issue: allocating per TCP session state is **expensive** (that's why the SYN flood attack works)
  - Delay allocation of state until remote host commits to three-way handshake
  - Send back SYN/ACK packet **without allocating state** on server; server's initial sequence number (ISN) encodes a secret "cookie" that is function of some combination of (src,dst,srcport,dstport and time).
  - Allocate state when client sends ACK to server's SYN/ACK (using cookie to validate)

Address spoofing(2)

- TTL-based IP filtering
  - From a given host the TTL is decremented by a certain number of hops (based on network topology)
  - Std IP implementations set the packet TTL value to a small set of values (32, 64, 128, 255) [can normalize because Internet diameter is mostly < 32)
  - Thus, keep track of TTLs for each source network and if attack starts, filter packets whose TTLs are inconsistent
    - Example: suppose packets arriving from 128.2.x.x typically take 4 hops to get to you
So you'd expect TTLs of 28, 60, 124, or 251… maybe +/- 1
Attacker doesn't know what TTL to put in DoS packets sent to you
Discard packets from 128.2.x.x that have any other ttl values

Packet filtering

- Idea, if there is a common feature to the packet (i.e. “Die, you loser” in the payload, static source port #, odd TCP flags, etc.) then look for those packets and drop them
- If no feature exists then try to find way to add a “good” feature to legitimate flows that complete a 3-way handshake
- Instead of dropping packets, can simply rate-limit packets that are suspicious
- Third-party services will offer these capabilities on your behalf in the network
  - Like dialysis for Internet traffic – route network traffic to devices designed to filter out bad packets using one of the above techniques

Buy more resources

- Large content distribution networks (e.g. Akamai, CloudFlare) can handle very large attacks
- Each attacker gets diverted (i.e., via DNS) to local Akamai server instead of target
  - Total bandwidth Akamai can handle is the product of the bandwidth to all Akamai servers
  - Akamai has weathered attacks well in excess of 1TB/s
- Issue: who pays for that? $$$

Special case: Reflection attacks

- Spoof source address to be that of victim
- Common example
  - Send name server request to 1000s of DNS servers *on behalf* of victim
  - All name servers send responses to victim
- Advantages (for attacker)
  - Amplification: some protocols response size >> request size (NTP, DNS, SSDP)
  - Anonymity: attack doesn't come from attacker's machines

- Solution: try not to have "open" Internet services that allow this

DoS Summary

- In general, some of the toughest problems to solve
  - Network service model allows unsolicited requests
  - Bad guys can leverage large # of resources
  - Hard to attribute network actions
  - Few systems can account for effort spent per request or isolate impact of some requests from others
- DDoS-based extortion and retribution
  (e.g., against security companies) is not uncommon
  Also widely used for political disruption

Network Perimeter Defense

- Idea: network defenses on “outside” of organization (e.g., between org and Internet)
  - Assumptions?
- Typical elements
  - Firewalls
  - Network Address Translation
  - Network Intrusion Detection/Prevention Systems(NIDS/NIPS) (and other kinds of network content analysis)

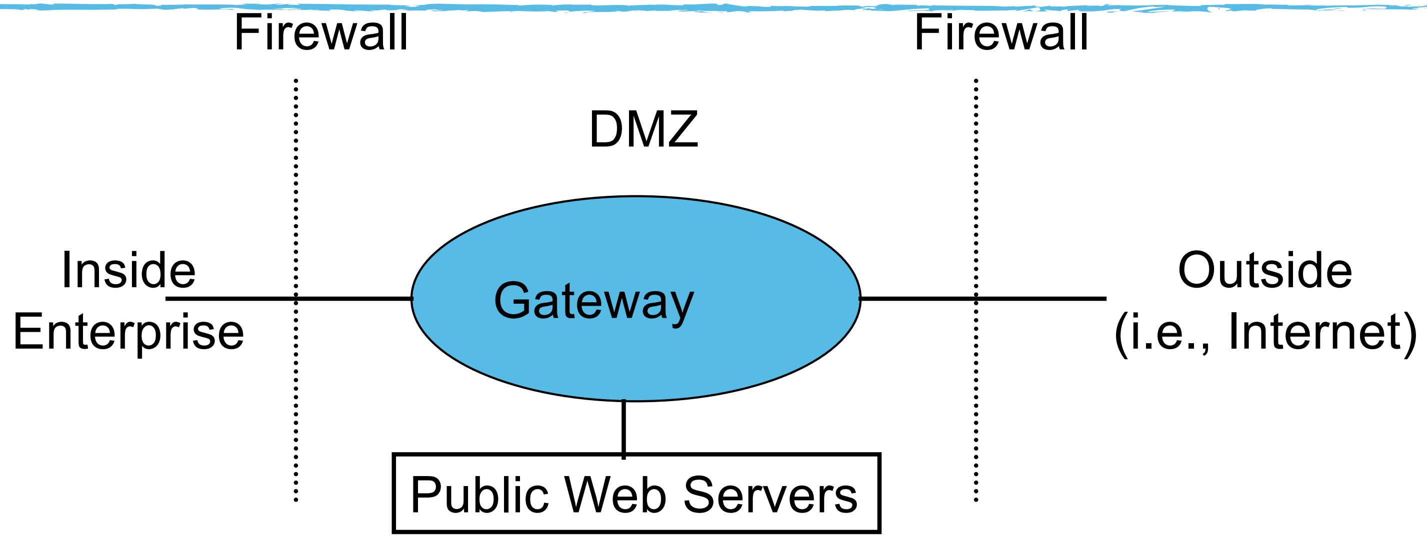
Firewalls

- Access control policy
  - You'd like to limit who can talk to what service
  - Deny access to bad people; allow it to good
- Firewalls: One place to try to enforce this policy is **in the network**
  - Inbound: requests from outside the network to connect to services inside
  - Outbound: requests from inside the network to connect to services outside
- Conceptually simple idea motivates much of this
  - The outside is bad and scary, while people inside are good and friendly
  - Allow internal users to communicate with any outside service
  - Limit outside connections to a small set of services designed to be externally visible
- Questions:
  - What information do you use to filter?
  - Where do you do the filtering?

Kinds of Firewalls

- Personal firewalls
  - Run at the end hosts
  - e.g. Norton, Windows, etc.
  - Benefit: has more application/user specific information
- Network firewalls
  - Intercept and evaluate communications from many hosts
  - Deployed "in-line" in network infrastructure (i.e., mediates all communications)
  - Different levels of abstraction: packet filtering vs application proxies

Network Firewall
common deployment strategy



- Filters protect against “bad” communications.
- Protect services offered internally from outside access.
- Provide outside services to hosts located inside.

Packet Filtering Firewalls

- Packet filtering firewalls can take advantage of the following information from network and transport layer headers:
  - Source IP
  - Destination IP
  - Source Port
  - Destination Port
  - Flags (e.g. ACK)

- Some firewalls keep state about open TCP connections
  - Allows conditional filtering rules of the form "if internal machine has established the TCP connection, permit inbound reply packets"

Ports

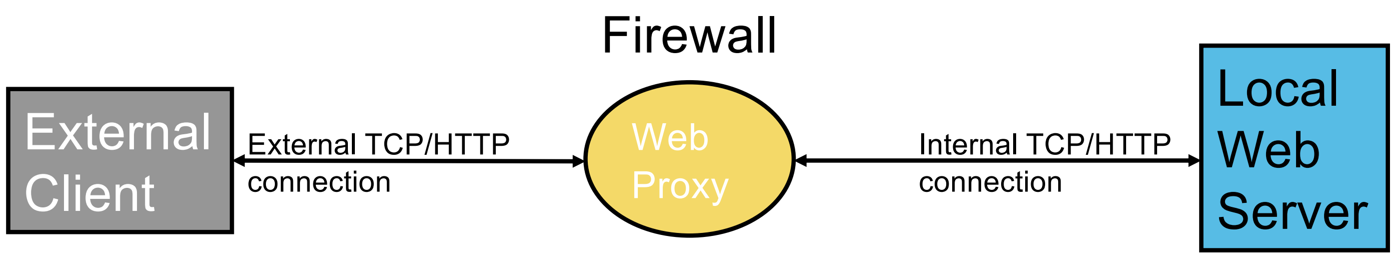
- Ports are used to distinguish applications and services on a machine.
- Low numbered ports are often reserved for server listening.
- High numbered ports are often assigned for client requests.
- Imperfect language…
- Result: everything gets "tunneled" through ports not blocked by the firewall (e.g., 80, 443)

- Port 20 (TCP): FTP data
- Port 21 (TCP): FTP control
- Port 22 (TCP): ssh
- Port 25 (TCP): SMTP (Mail)
- Port 80 (TCP): HTTP
- Port 123 (UDP): NTP
- Port 143 (TCP): IMAP
- Port 443 (TCP): HTTPS
- Port 2049 (UDP): NFS
- Ports 6000 to 6xxx (TCP): X11

Principles for Firewall Configuration

- Least Privilege:
  - Turn off everything that is unnecessary
    (e.g. Web Servers should disable port 25 [SMTP])
- Failsafe Defaults:
  - By default should reject
  - (Note that this can cause usability problems…)
- Egress Filtering:
  - Filter outgoing packets too!
  - You know the valid IP addresses for machines internal to the network, so drop those that aren't valid.
  - As per earlier, this can help prevent DoS attacks in the Internet.

Proxy-based Firewalls



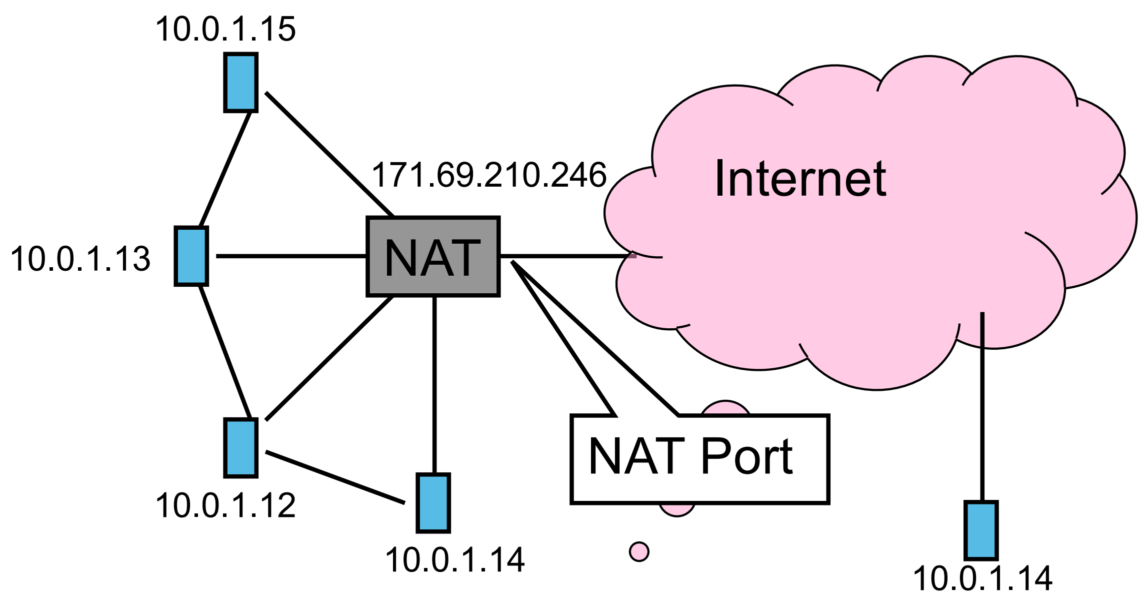
- Proxy acts like *both* a client and a server.
  - Note, must terminate connection. Complications for HTTPS/TLS… must take out certificates
- More semantics available: able to filter using application-level info
  - For example, block based on URL, or on particular javascript strings
- Proxies can provide other services too
  - Caching, load balancing, etc.
  - Key escrow (e.g., reverse proxies for ssh, SSL)

Firewalls Pro/Con

- Benefits
  - Reduced "attack surface" against external attackers
  - Filter out lots of "noise" in network traffic (helps focus attention)
  - Reduced liability (common practice)
- Costs
  - Actual cost: both hardware and administration
  - Bottleneck and single point of failure on network
  - False sense of security
    - Limited language (addresses, ports); doesn't help with worms/viruses, ssh exploits, cross-site scripting, etc
    - Inside vs outside model is fragile (once an internal host is compromised firewall does no good); What about wireless laptops?
    - Modern companies increasingly offer no additional trust to machines inside the firewall (so-called "zero trust" architectures)

Network Address Translation

- Idea: Break the invariant that IP addresses are globally unique
  - Special addresses that are only **local**: 10.x.x.x, 192.168.x.x and 172.16.0.0-172.31.255.255



Typical NAT Behavior

- NAT maintains a table of the form:
  <client IP> <client port> <NAT ID>

- Outgoing packets (on internal port):
  - Look for client IP address, client port in the mapping table
  - If found, replace client port with previously allocated NAT ID (same size as port #)
  - If not found, allocate a new unique NAT ID and replace source port with NAT ID
  - Replace source address with NAT address (i.e., public IP address)

- Incoming Packets (on NAT port)
  - Look up destination port number as NAT ID in port mapping table
  - If found, replace destination address and port with client entries from the mapping table
  - If not found, the packet is not for us and should be rejected

- Table entries expire after 2-3 minutes of no activity to allow them to be garbage collected

NAT Pro/Con

- Benefits
  - Only allows connections to the outside that are established from *inside.*
    - Hosts from outside can only contact internal hosts that appear in the mapping table, and they're only added when they establish the connection
  - Don't need as large an external address space
    - (e.g., 10 machines can share 1 IP address)
- Costs
  - Rewriting IP addresses isn't always easy (what if they appear in the **content** of the packet too? e.g., FTP. Then what happens to sequence numbers?)
  - Breaks some protocols (e.g., some streaming protocols have client invoke server and then server opens new connection to client)
  - But we've paid these costs, NAT is now ubiquitous…

Network content analysis

- Lots of devices want to look at network traffic **content** for security
  - Network Intrusion detection/prevention Systems (NIDS/NIPS)
    - Try to find signatures of attacks or malware
  - Spam filters
    - Try to detect unwanted e-mail
  - Data leakage
    - Try to prevent sensitive information from leaving company
  - Traffic differentiation
    - Filter or slow down BitTorrent traffic, Netflix traffic, etc
- Doing this as in the network is attractive because its cheaper and easier to manage than putting endpoint monitoring on each host

Challenges

- Expensive to look into each packet
  - 10Gbps -> ~1M packets per second… ns' per byte
  - Also must reassemble pkts into in-order streams (e.g., what if signature you are looking for spans two packets)
- Network vantage point is imperfect
  - What does a packet *mean*?
  - What if a session is encrypted?
  - Network evasion?

Network evasion

- Typically network intrusion detection systems are deployed like firewalls (between internal network and Internet)
- Key assumption is that NIDS sees the same traffic as destination host
- Not quite true…
  - Lots of ways to evade a NIDS by exploiting ambiguity

TTL evasion

- Suppose destination host is **2 hops** inside network ***after*** NIDS box
- This is what NIDS sees (each letter is one pkt)

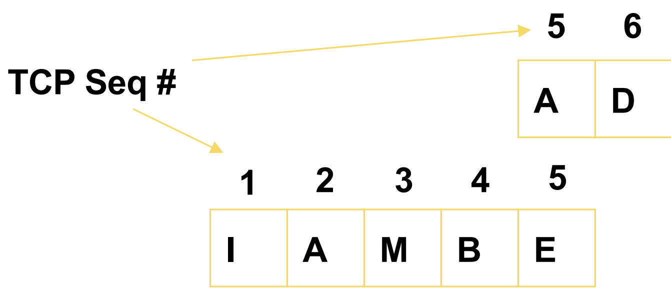
ttl=2	ttl=2	ttl=2	ttl=2	ttl=1	ttl=2	ttl=2
I	A	M	B	E	A	D

- This is what the destination host sees

ttl=1	ttl=1	ttl=1	ttl=1	ttl=1	ttl=1
I	A	M	B	A	D

Sequence # evasion

- Suppose attacker sends two packets



- What does destination see?
  - IAMBED? IAMBAD? IAMBD? Depends on host

- **Lots** of other evasion techniques…

Solution

- Protocol normalization
  - NIDS rewrites packets to remove all ambiguity
  - E.g.
    - all packets have ttl rewritten to reach any internal destination
    - IDS tracks each flow and does not allow overlapping packets
- Can be very tricky to get right and expensive
  - Potentially must buffer large amounts of data
  - What if you get seq #2 through #100, but not seq #1?
  - Tradeoff: when to drop data vs when to buffer

Bottom line

- The network vantage point is a very appealing place to implement policy because its central – **everyone has to go through it**!
  - But very challenging to infer the semantics of unknown communication
    - Port numbers communication very limited info
    - Parsing content is tough and many ways to evade
    - End-to-end encryption makes it hard to get content
- In spite of this, everyone uses these approaches because they don't have anything better

Next time

- User authentication