Protocol Security and DoS Attacks

CS155 Computer and Network Security

Acknowledgments: Lecture slides are from the Computer Security course taught by Dan Boneh and Zakir Durumeric at Stanford University. When slides are obtained from other sources, a a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.

Stanford University

Ethernet

Provides connectivity between hosts on a *Local Area Network*

Frames are addressed to a device's physical (MAC) address

Switches forward frames based on *learning* where different MACs are located. *No guarantees not sent to other hosts!*

No security (confidentiality, authentication, or integrity)

ARP (Address Resolution Protocol)

ARP allows hosts to find each others' MAC addresses on the local network

Client: To Broadcast (all MACs): *Which MAC address has IP 192.168.1.1?*

No built-in security. Attacker can impersonate a host by faking its identity and responding to ARP requests or sending gratuitous ARP announcement

IP (Internet Protocol)

Provides routing between hosts on the Internet. Unreliable. Best Effort.

Routers simply route IP packets based on their destination address.

No inherent security. Packets have a checksum, but it's non-cryptographic.
Attackers can change any packet.

Source address is set by sender—can be faked by an attacker

BGP (Border Gateway Protocol)

Internet Service Providers announce their presence on the Internet via BGP

No authentication—possible to announce someone else's network

Commonly occurs (often due to operator error)

TCP (Transmission Control Protocol)

TCP provides reliable stream of data on top of unreliable IP and Ethernet

Data is split into segments and sender/receiver acknowledge received byte of data, sender retransmit dropped packets

Every TCP connection starts with a three-way handshake

DNS Security

Users/hosts trust the host-address mapping provided by DNS

- Used as basis for many security policies:
  - Browser same origin policy, URL address bar

Interception of requests or compromise of DNS servers can result in incorrect or malicious responses

DNS Spoofing

Scenario: DNS client issues query to server

Attacker would like to inject a fake reply
  Attacker does not see query or real response

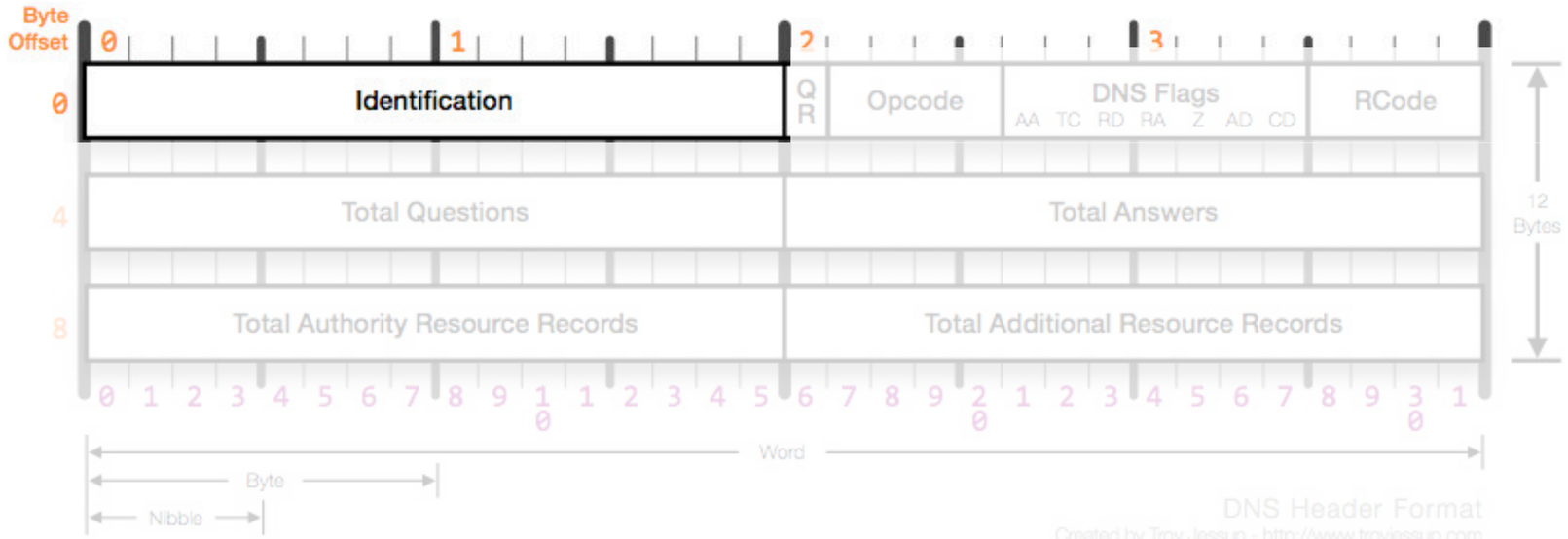
How does client authenticate response?

DNS Spoofing

How does client authenticate response?

UDP port numbers must match
Destination port usually port 53 by convention

16-bit query ID must match



DNS Caching

Recursive resolvers cache records to avoid repeating recursive resolution process for each query

Lifetime of record determined by record TTL
Could also be evicted from cache due to limited memory

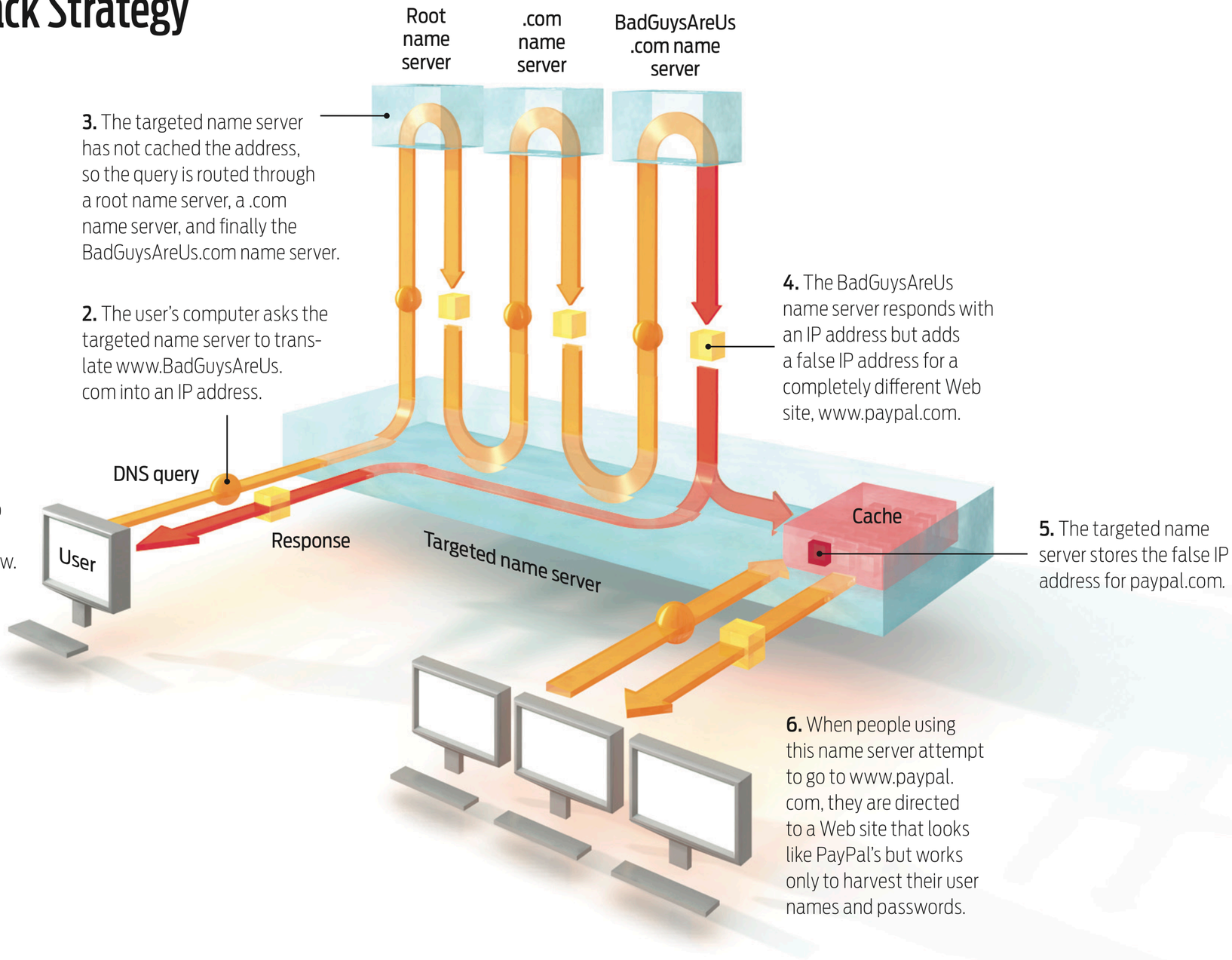
Injecting spoofed records into a resolver's cache is called *DNS cache poisoning*

DNS Cache Poisoning

DNS query results include Additional Records section
– Provide records for anticipated next resolution step

Early servers accepted and cached all additional records provided in query response

Early Attack Strategy



Glue Records

Can we just stop using additional section?

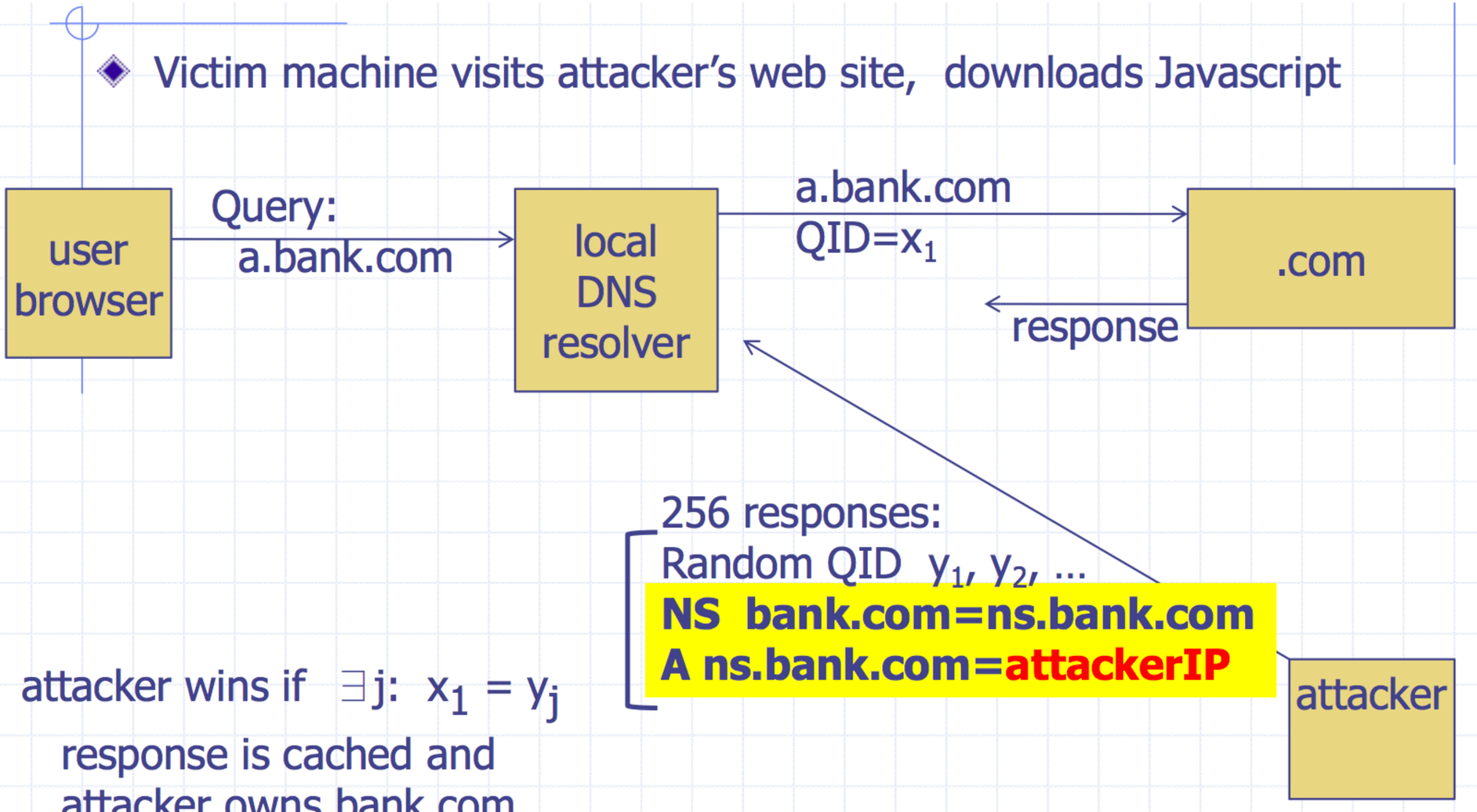
– Only accept answers from authoritative servers?

Glue records: non-authoritative are records necessary to contact next hop in resolution chain

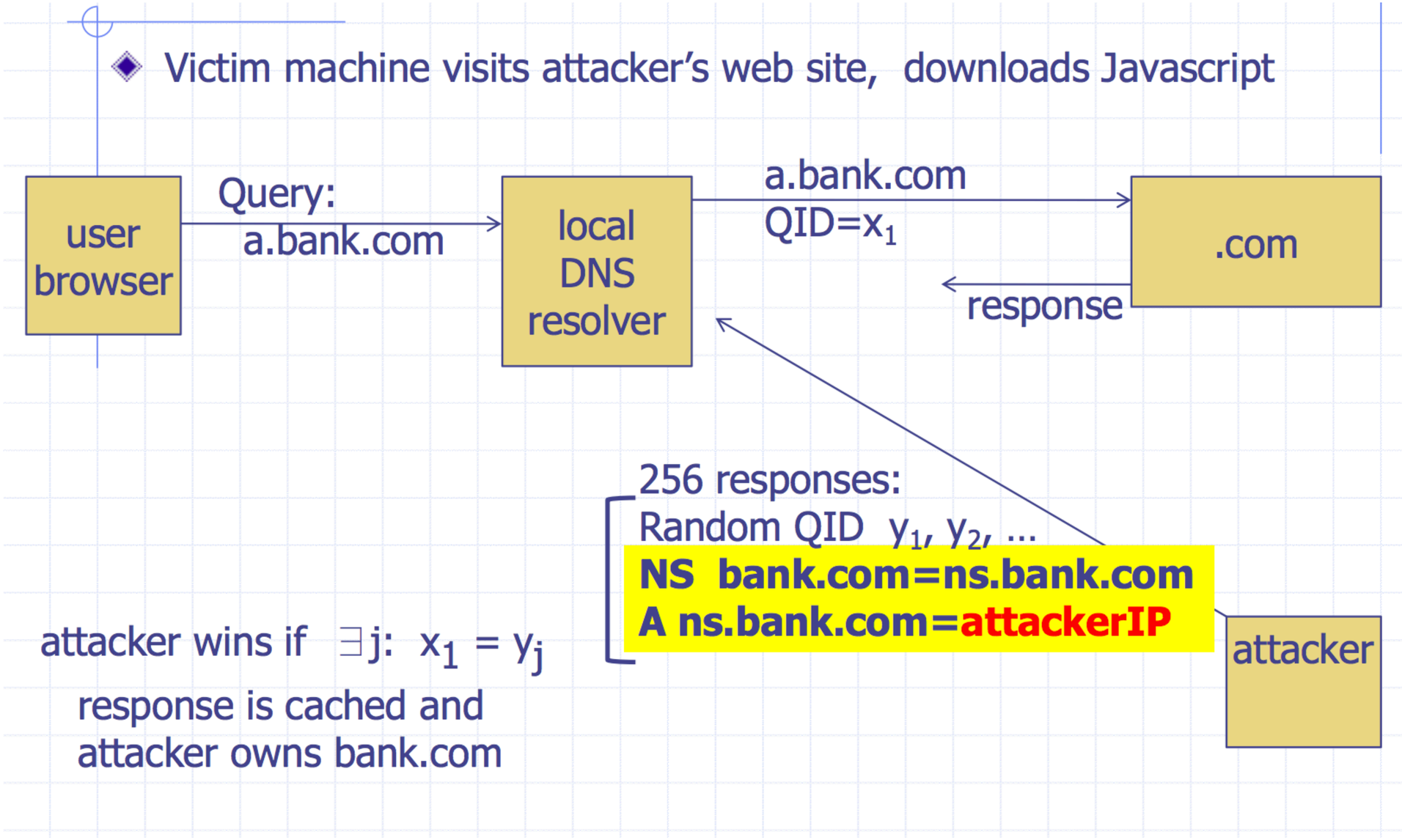
– Necessary given current design of DNS

Bailiwick Checking: Only accept additional records that are for a domain in the original question.

Kaminsky Attack



Try Again!


Victim machine visits attacker's web site, downloads Javascript
user browser
Query:
a.bank.com
local DNS resolver
a.bank.com
QID=x1
.com
response
256 responses:
Random QID y1, y2, ...
NS bank.com=ns.bank.com
A ns.bank.com=attackerIP
attacker
attacker wins if ∃j: x1 = yj
response is cached and
attacker owns bank.com

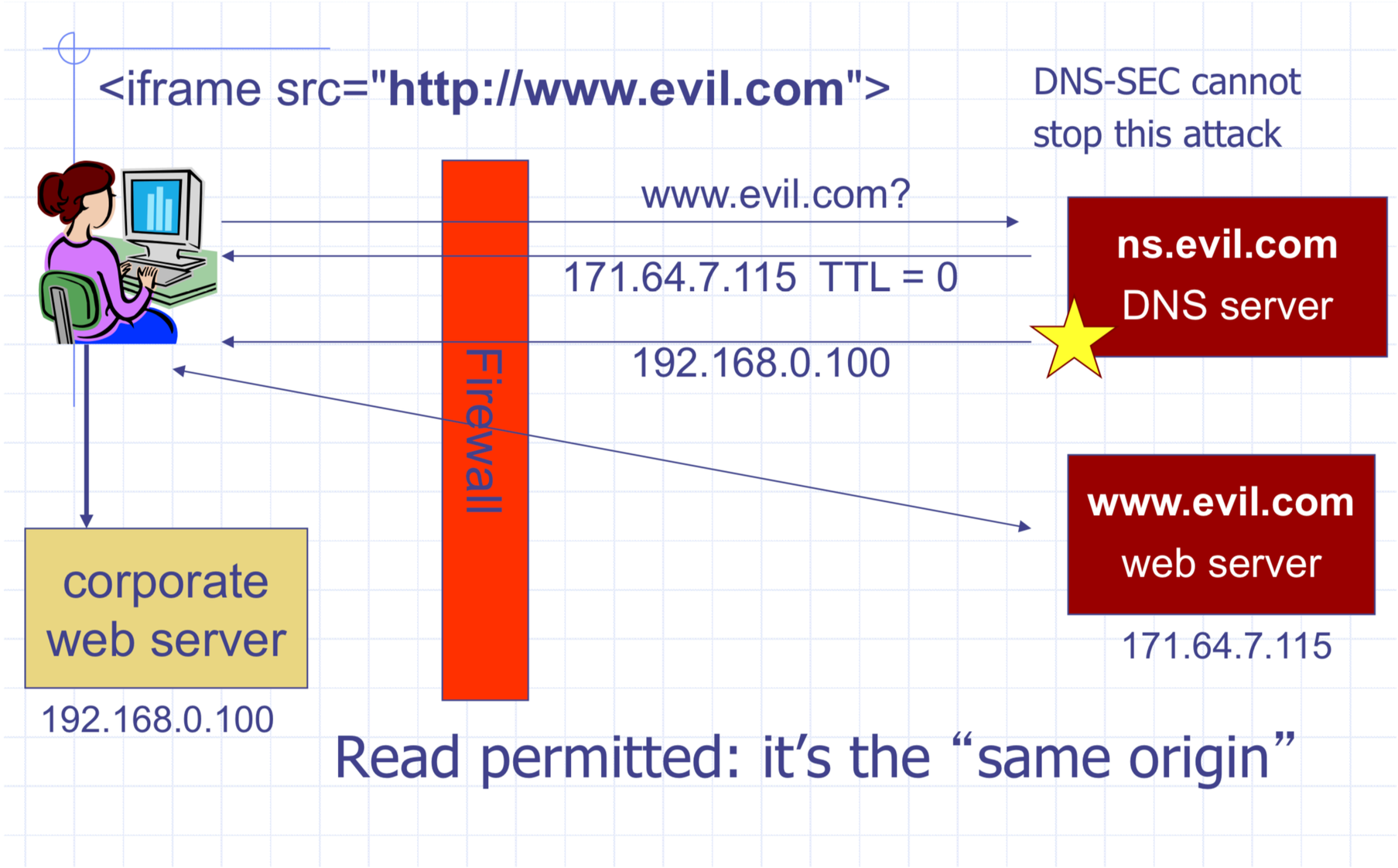
Defenses

Increase QueryID space. But how? Don’t want to change packet.

Randomize src port, additional 11 bits of entropy

- Attack now takes several hours

DNS Rebinding



Rebinding Defenses

Browser Mitigations:

- Refuse to switch IPs mid session
- Interacts poorly with proxies, VPNs, CDNs, etc
- Not consistently implemented in any browser

Server Defenses

- Check Host header for unrecognized domains
- Authenticate users with something else beyond IP address

DNSSEC

Adds authentication and integrity to DNS responses

Authoritative DNS servers sign DNS responses using cryptographic key

Clients can verify that a response is legitimate by checking signature through PKI similar to HTTPS

Most people don’t use DNSSEC and never will. Use TLS instead.

Network Security Takeaway

Assume the network is out to get you.

If you want any guarantee of any security, use TLS.

Denial of Service Attacks

Goal: take large site offline by overwhelming it with network traffic such that they can't process real requests

How: find mechanism where attacker doesn't have to spend a lot of effort, but requests are difficult/expensive for victim to process

Types of Attacks

DoS Bug: design flaw that allows one machine to disrupt a service. Generally a protocol asymmetry, e.g., easy to send request, difficult to create response. Or requires server state.

DoS Flood: control a large number of requests from a botnet of machines you control

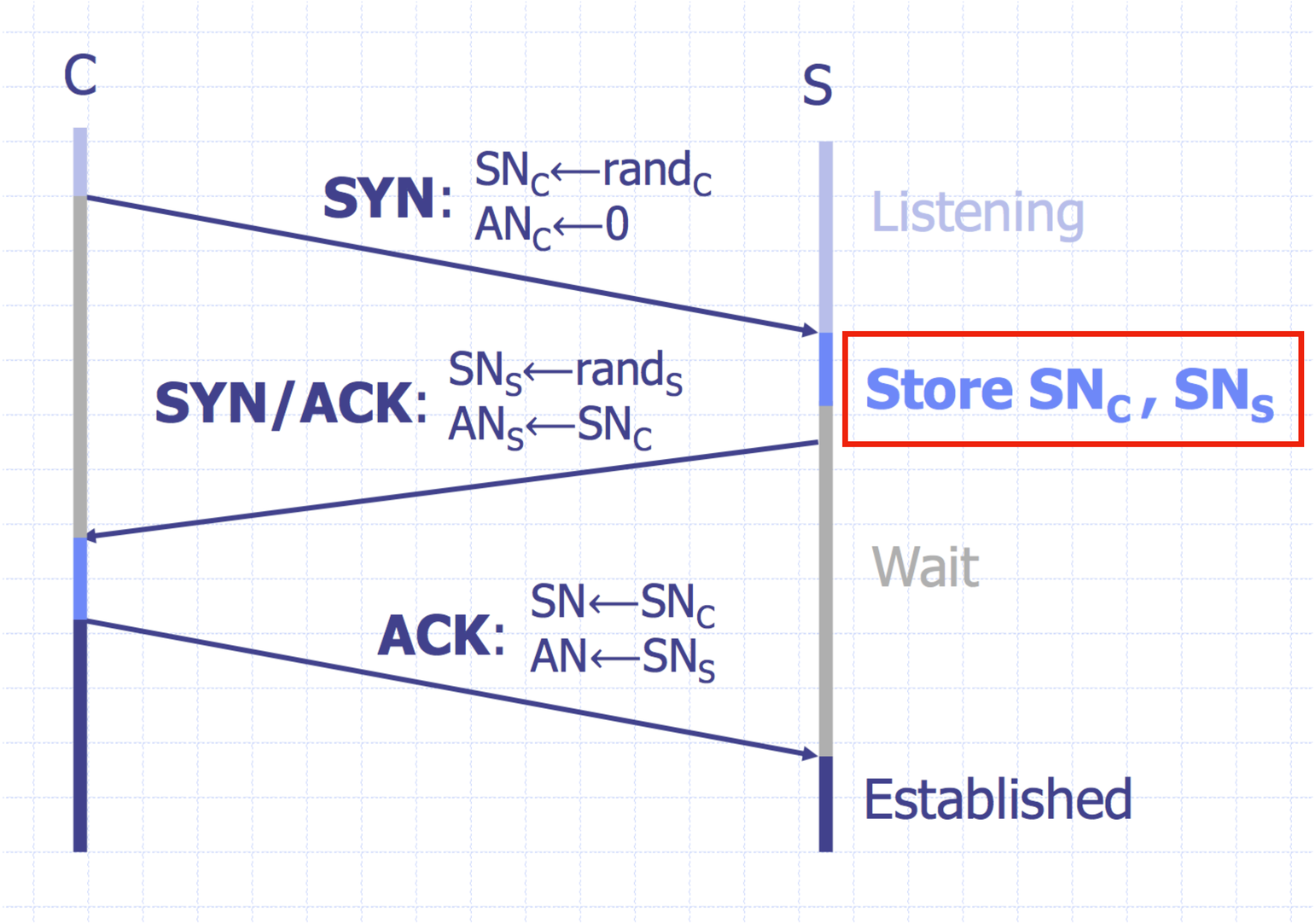
Possible at Every Layer

Link Layer: send too much traffic for switches/routers to handle

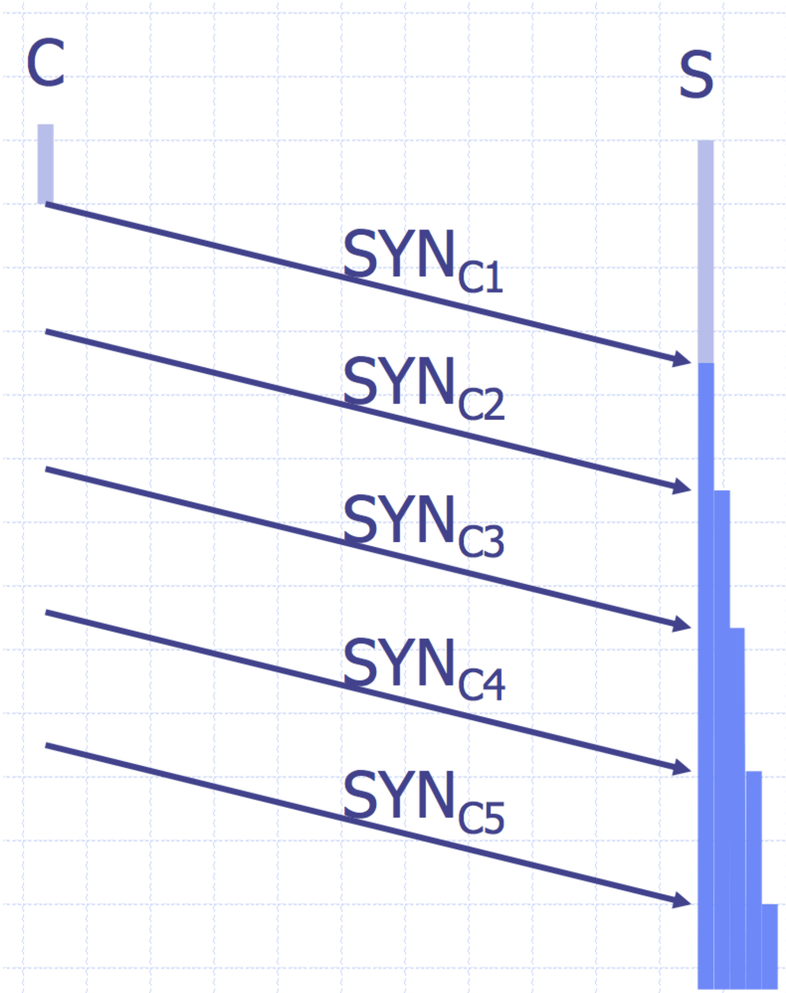
TCP/UDP: require servers to maintain large number of concurrent connections or state

Application Layer: require servers to perform expensive queries or cryptographic operations

TCP Handshake



SYN Floods



Single machine:

- SYN Packets with **random source IP addresses**
- Fills up backlog queue on server
- No further connections possible

Core Problem

Problem: server commits resources (memory) before confirming identify of client (when client responds)

Bad Solution:

- Increase backlog queue size
- Decrease timeout

Real Solution: Avoid state until 3-way handshake completes

SYN Cookies

Idea: Instead of storing SNc and SNs…
send a cookie back to the client.

$L = MAC_{key}$ (SAddr, SPort, DAddr, DPort, SN_C, T)

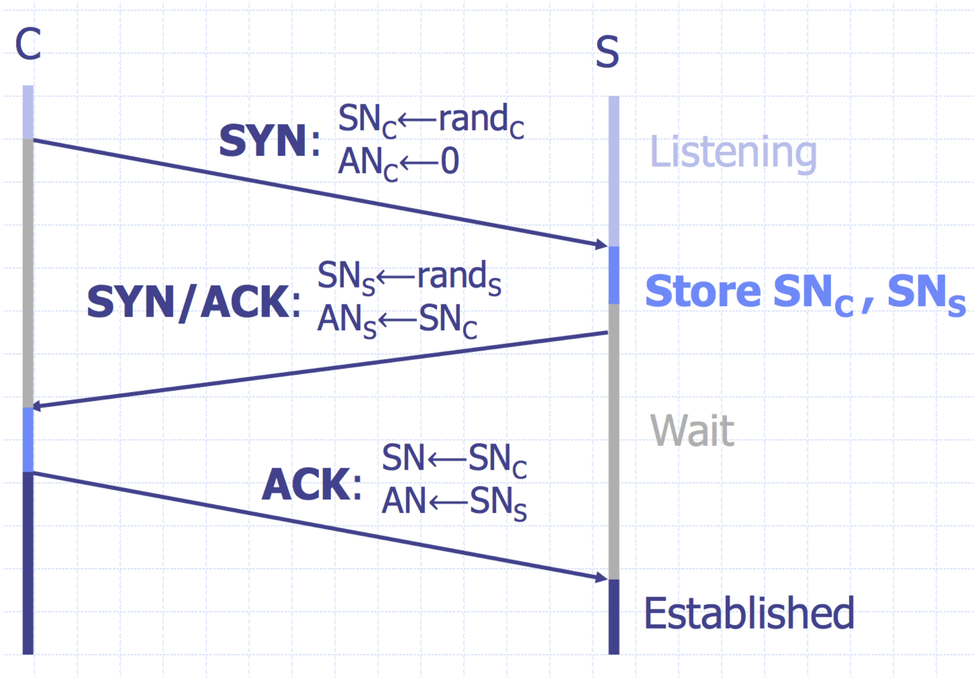
key: picked at random during boot

T = 5-bit counter incremented every 64 secs.

$SN_s = (T \| mss \| L)$

Honest client sends ACK ($AN=SN_s$, $SN=SN_C+1$)

Server allocates space for socket only if valid SNs



Server does not save state
(loses TCP options)