Denial-of-Service and Resource Exhaustion

Nick Feamster
CS 7260
April 2, 2007

Today's Lecture

- What is Denial of Service?
- Attacks and Defenses
  - Packet-flooding attacks
    - **Attack:** SYN Floods
    - **Defenses:** Ingress Filtering, SYN Cookies, Client puzzles
  - Low-rate attacks
    - **Detection:** Single-packet IP Traceback
- **Network-level defenses:** sinkholes and blackholes

- Inferring Denial of Service Activity
- Distributed Denial of Service
- Worms
- Other resource exhaustion: spam

Denial of Service: What is it?

- Attempt to *exhaust resources*
  - **Network:** Bandwidth
  - **Transport:** TCP connections
  - **Application:** Server resources
- Typically high-rate attacks, but not always

Pre-2000 Denial of Service

DoS Tools

- Single-source, single target tools
- IP source address spoofing
- Packet amplification (e.g., smurf)

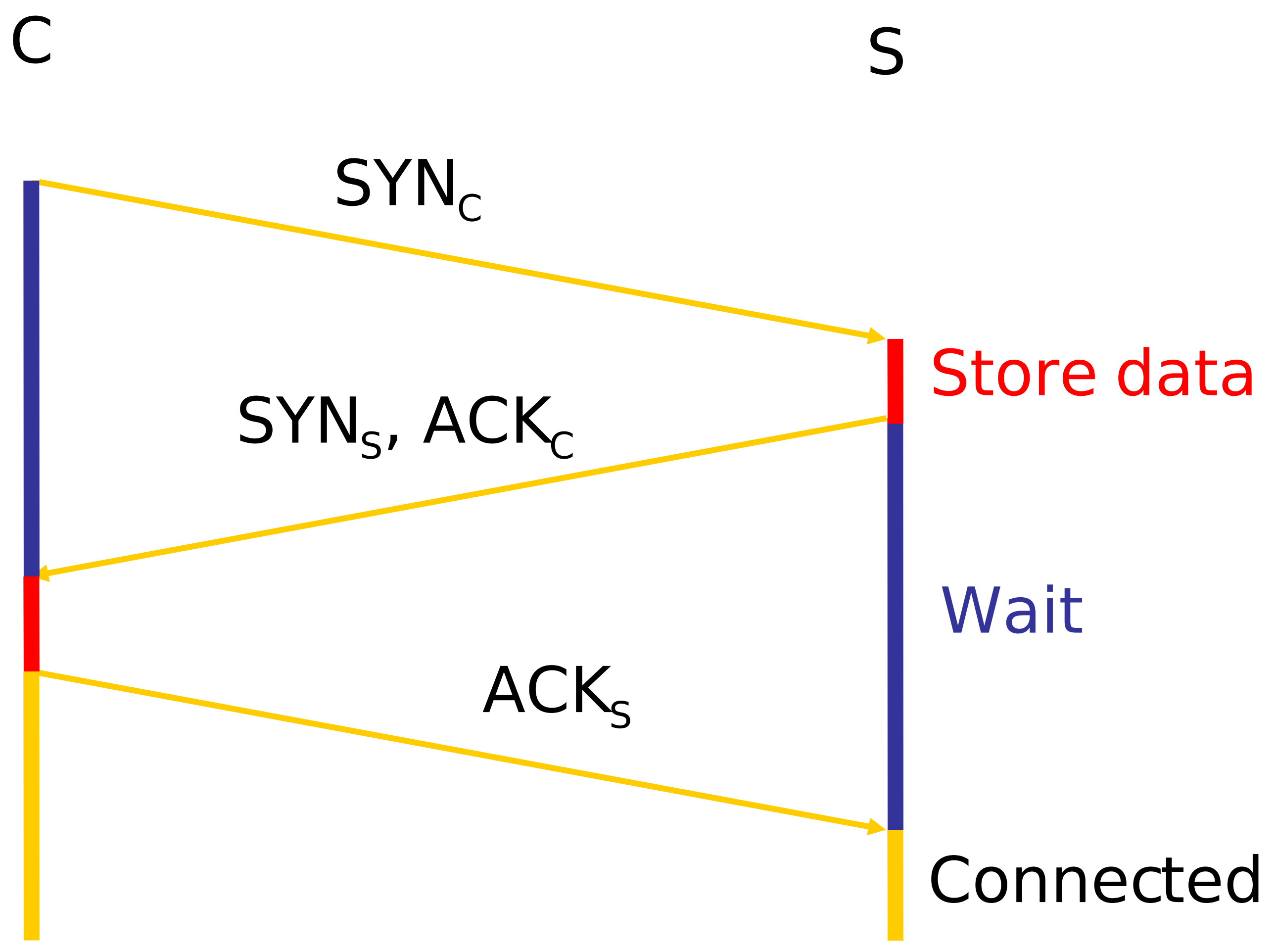
Deployment

- Widespread scanning and exploitation via scripted tools
- Hand-installed tools and toolkits on compromised hosts (unix)

Use

- Hand executed on source host

TCP: 3-Way Handshake



TCP handshake

- Each arriving SYN stores state at the server
  - TCP Control Block (TCB)
  - ~ 280 bytes
    - FlowID, timer info, Sequence number, flow control status, out-of-band data, MSS, other options agreed to
  - Half-open TCB entries exist until timeout
  - Fixed bound on half-open connections

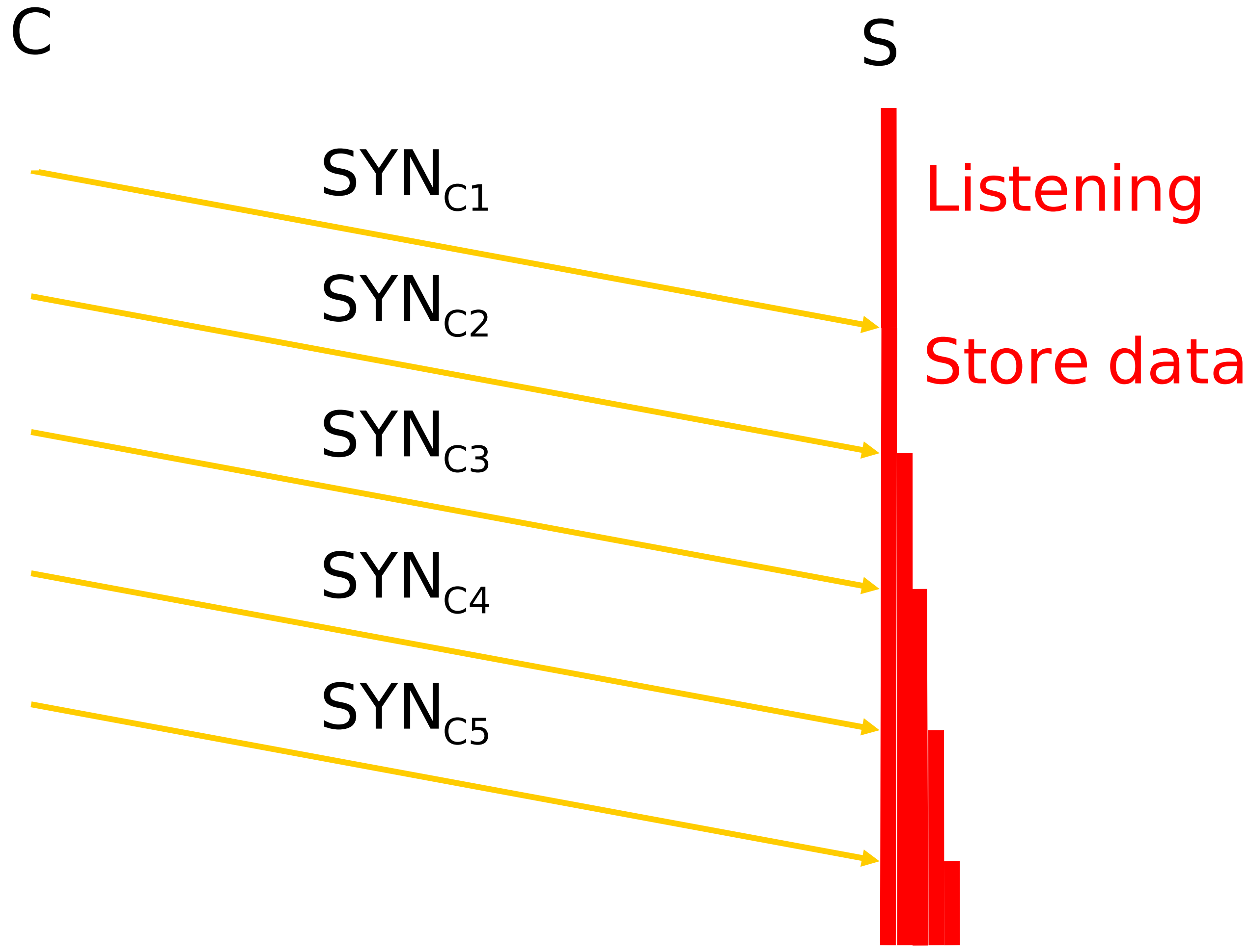
- Resources exhausted $\Rightarrow$ requests rejected

TCP SYN flooding

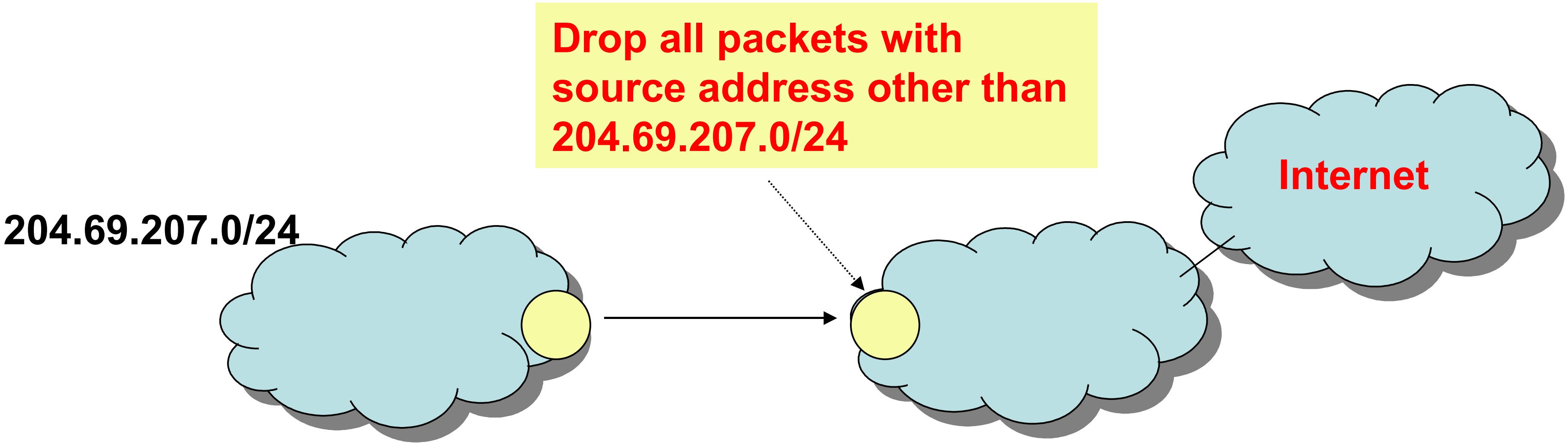
- **Problem:** No client authentication of packets before resources allocated

- Attacker sends many connection requests
  - Spoofed source addresses
  - RSTs quickly generated if source address exists
  - No reply for non-existent sources
    - Attacker exhausts TCP buffer to w/ half-open connections

SYN Flooding



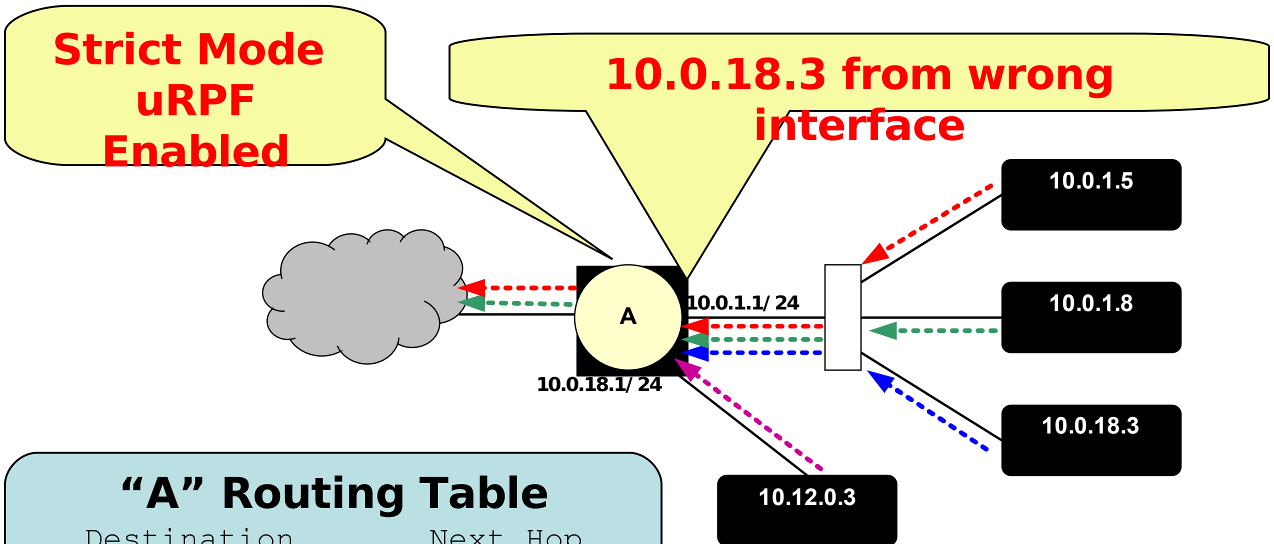
Idea #1: Ingress Filtering



- **RFC 2827:** Routers install filters to drop packets from networks that are not downstream
- Feasible at edges
- Difficult to configure closer to network “core”

Idea #2: uRPF Checks

Accept packet from interface only if forwarding table entry for source IP address matches ingress interface

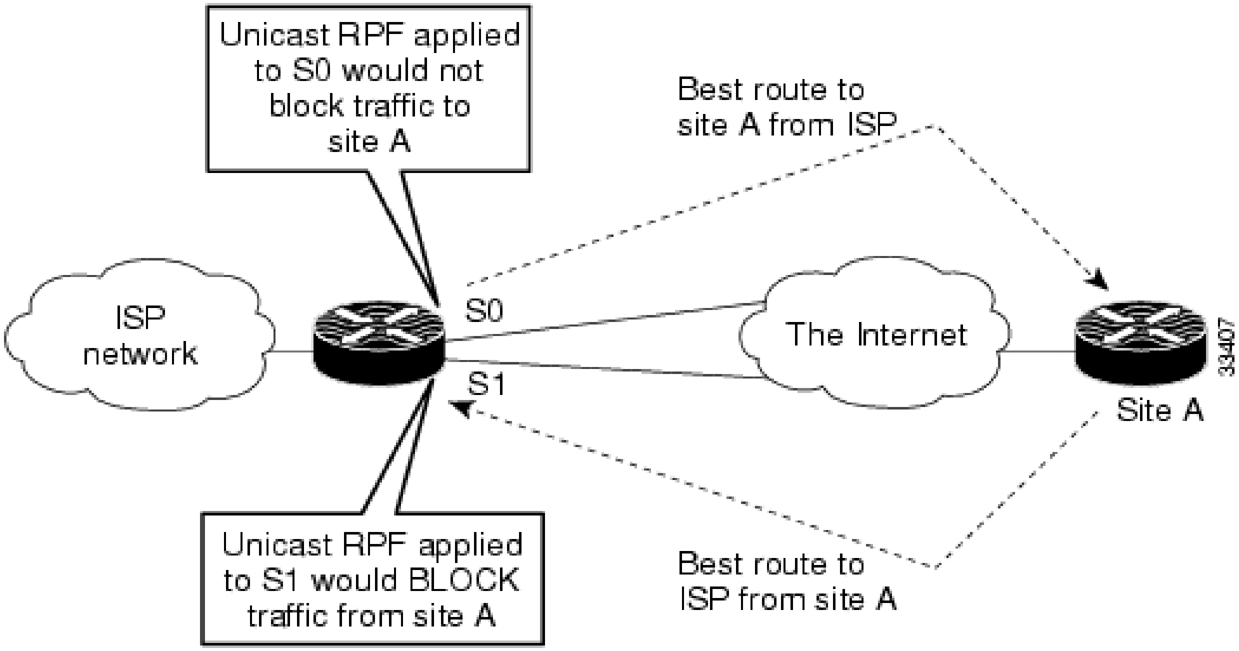


"A" Routing Table

Destination	Next Hop
10.0.1.0/24	Int. 1
10.0.18.0/24	Int. 2

- Unicast Reverse Path Forwarding
  - Cisco: "**ip verify unicast reverse-path**"
- Requires symmetric routing

Problems with uRPF



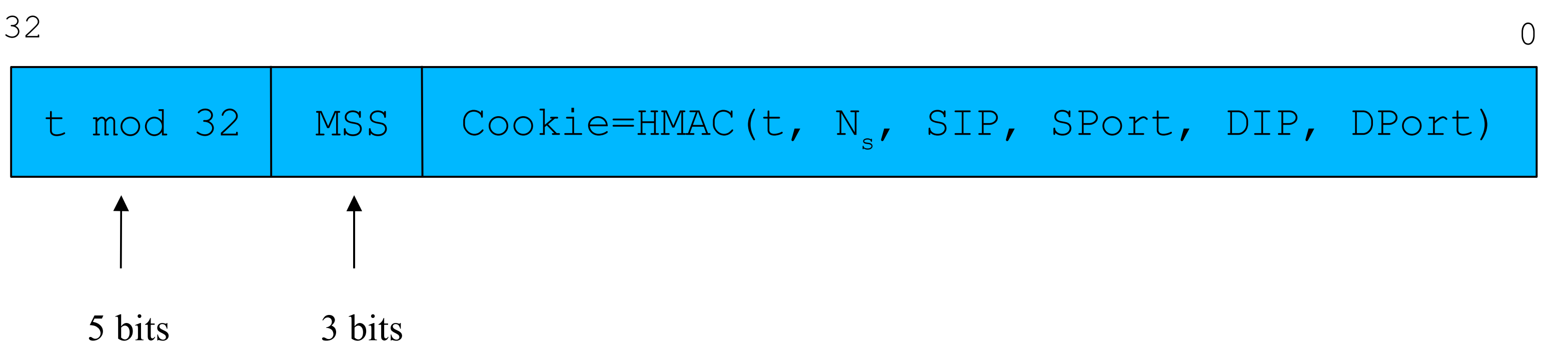
- Asymmetric routing

Idea #3: TCP SYN cookies

- General idea
  - Client sends SYN w/ ACK number
  - Server responds to Client with SYN-ACK cookie
    - sqn = f(src addr, src port, dest addr, dest port, rand)
    - Server does not save state
  - Honest client responds with ACK(sqn)
  - Server checks response
  - If matches SYN-ACK, establishes connection

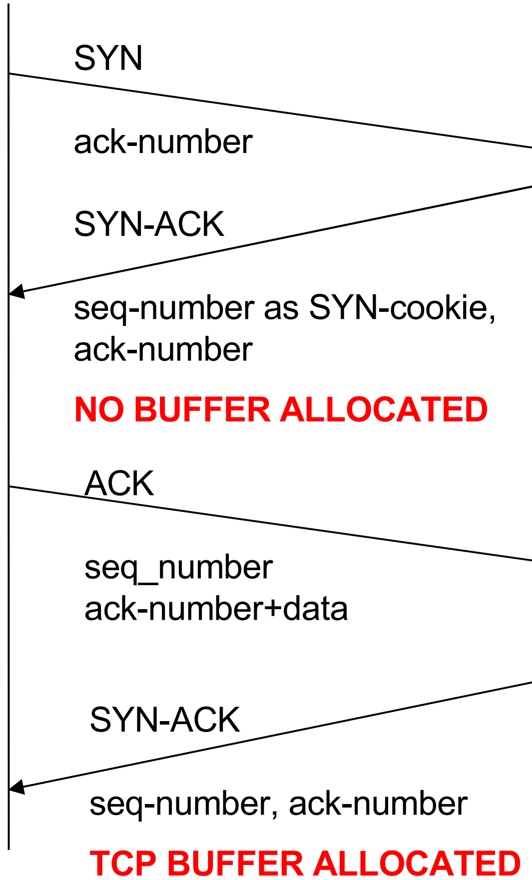
TCP SYN cookie

- TCP SYN/ACK seqno encodes a cookie
  - 32-bit sequence number
    - **t mod 32:** counter to ensure sequence numbers increase every 64 seconds
    - **MSS:** encoding of server MSS (can only have 8 settings)
    - **Cookie:** easy to create and validate, hard to forge
      - Includes timestamp, nonce, 4-tuple

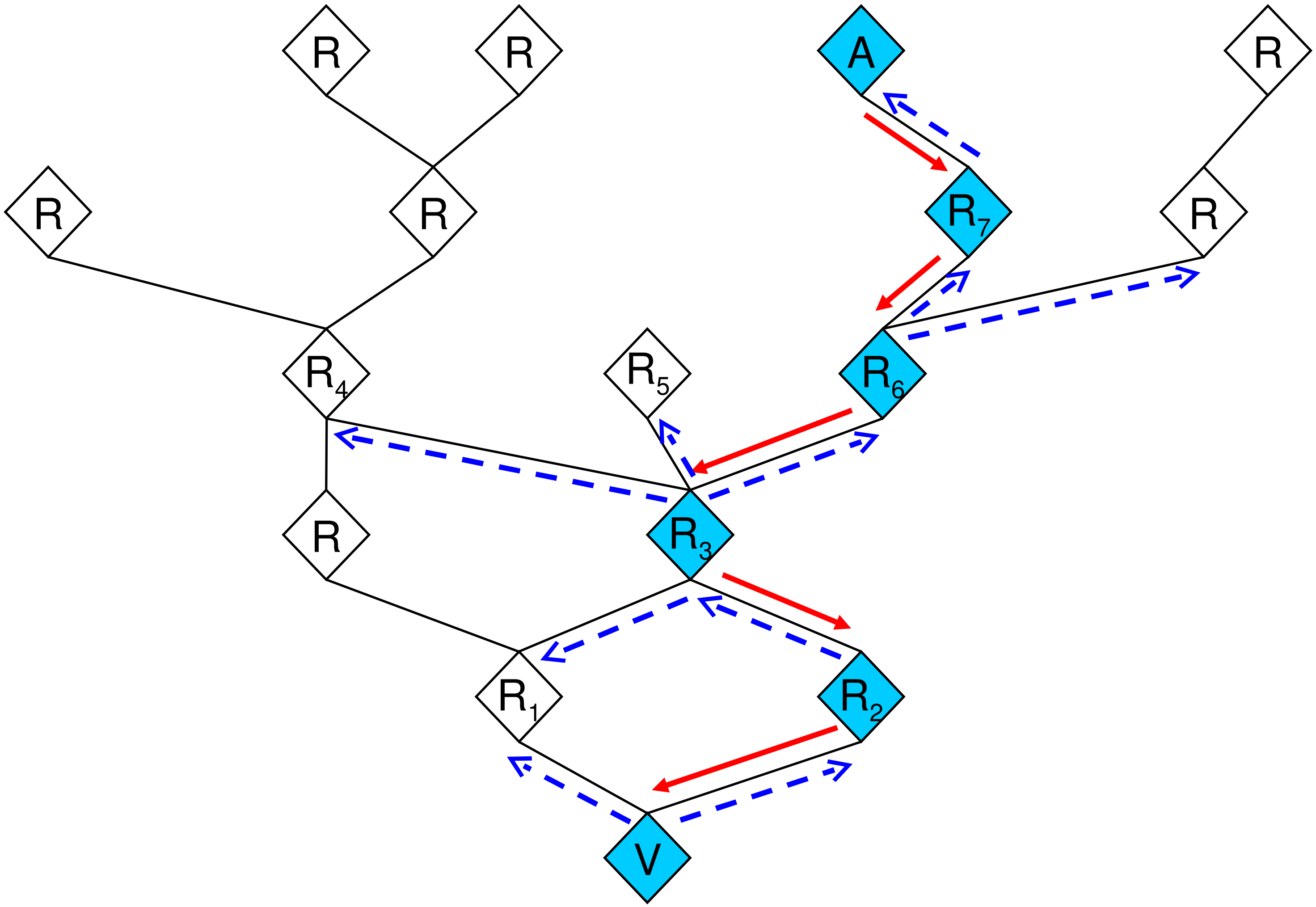


SYN Cookies

- client
  - sends SYN packet and ACK number to server
  - waits for SYN-ACK from server w/ matching ACK number
- server
  - responds w/ SYN-ACK packet w/ initial SYN-cookie sequence number
  - Sequence number is cryptographically generated value based on client address, port, and time.
- client
  - sends ACK to server w/ matching sequence number
- server
  - If ACK is to an unopened socket, server validates returned sequence number as SYN-cookie
  - If value is reasonable, a buffer is allocated and socket is opened



IP Traceback



Logging Challenges

- Attack path reconstruction is difficult
  - Packet may be transformed as it moves through the network
- Full packet storage is problematic
  - Memory requirements are prohibitive at high line speeds (OC-192 is ~10Mpkt/sec)
- Extensive packet logs are a privacy risk
  - Traffic repositories may aid eavesdroppers

Single-Packet Traceback: Goals

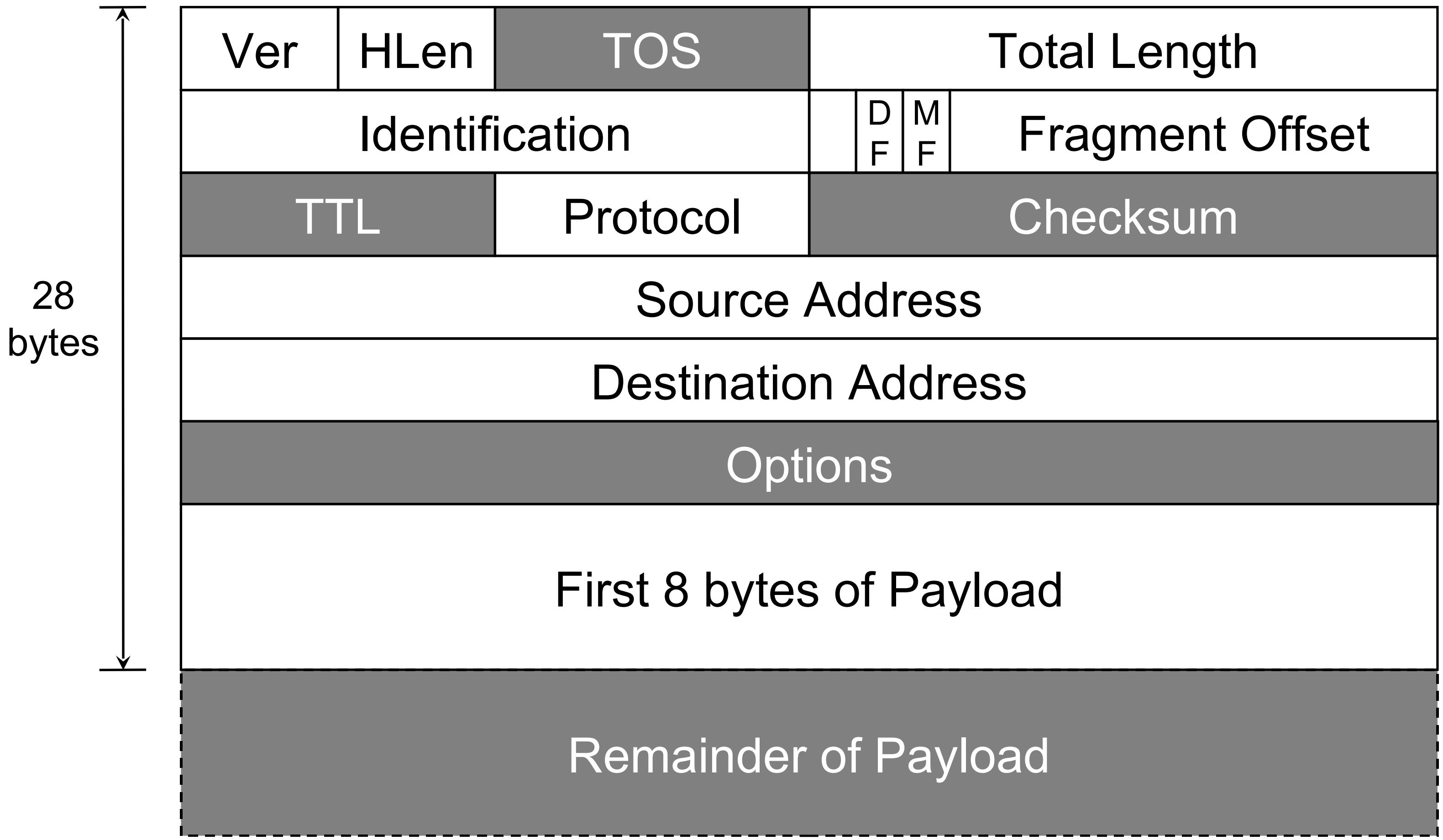
- Trace a *single* IP packet back to source
  - Asymmetric attacks (*e.g.,* Fraggle, Teardrop, ping-of-death)
- Minimal cost (resource usage)

One solution: Source Path Isolation Engine (SPIE)

Packet Digests

- Compute hash(p)
  - Invariant fields of p only
  - 28 bytes hash input, 0.00092% WAN collision rate
  - Fixed sized hash output, *n*-bits
- Compute *k* independent digests
  - Increased robustness
  - Reduced collisions, reduced false positive rate

Hash input: Invariant Content



Hashing Properties

- Each hash function
  - Uniform distribution of input -> output

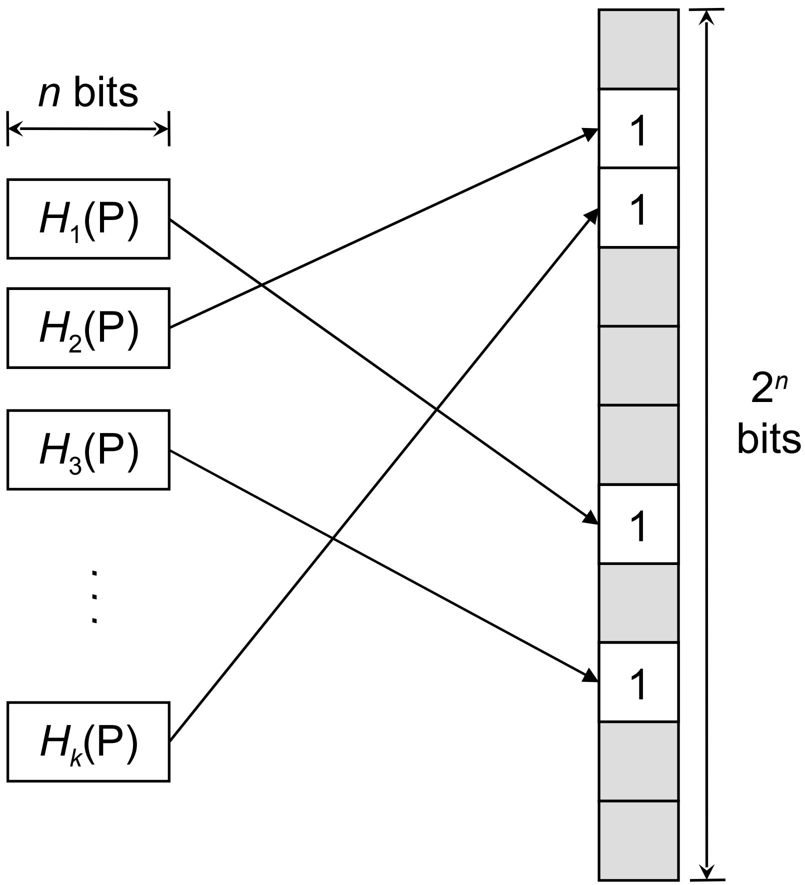
    H1(x) = H1(y) for some x,y -> unlikely

- Use k independent hash functions
  - Collisions among k functions independent
  - H1(x) = H2(y) for some x,y -> unlikely

- Cycle k functions every time interval, t

Digest Storage: Bloom Filters

- **Fixed structure size**
  - Uses 2^n bit array
  - Initialized to zeros
- **Insertion**
  - Use *n*-bit digest as indices into bit array
  - Set to ‘1’
- **Membership**
  - Compute *k* digests, d_1, d_2, etc…
  - If (filter[d_i]=1) for all i, router forwarded packet

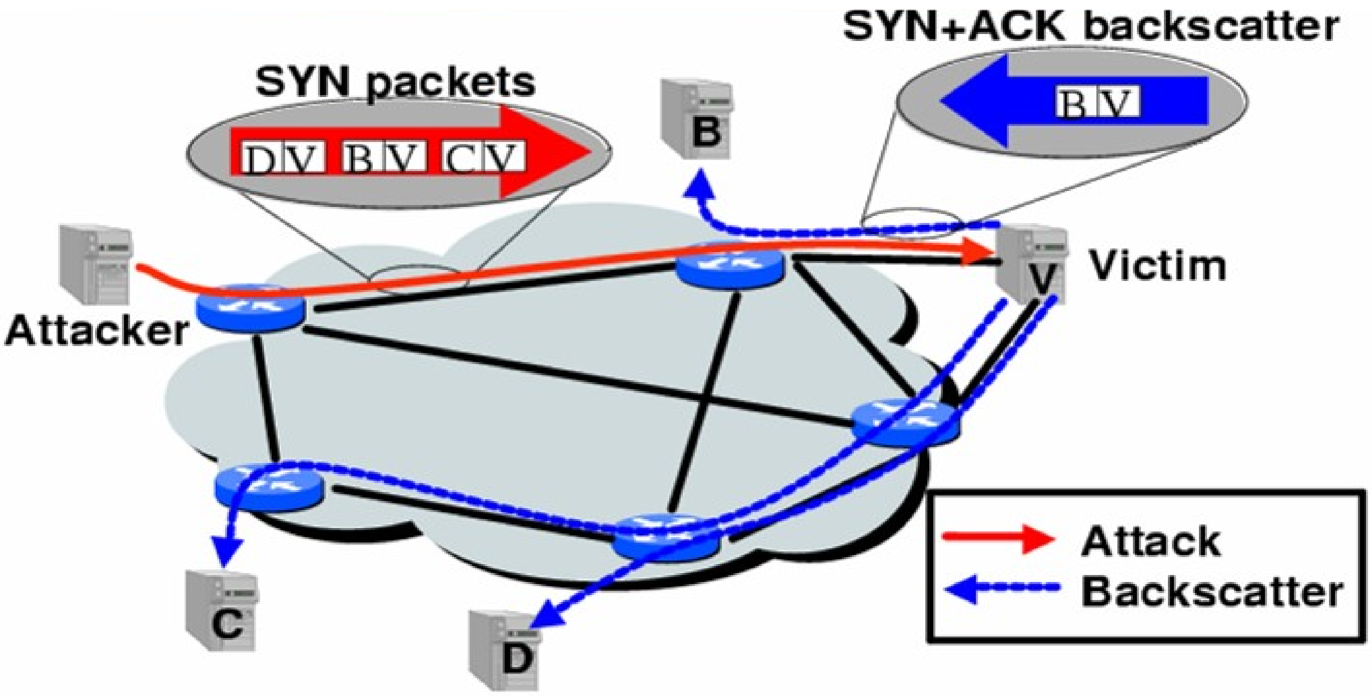


Other In-Network Defenses

- Automatic injection of blackhole routes
- Rerouting through traffic "scrubbers"

Inferring DoS Activity

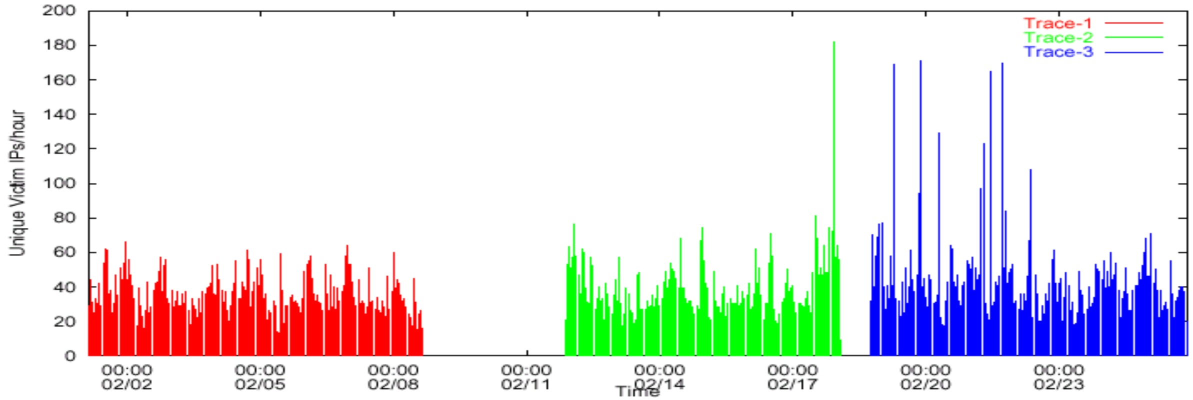
IP address spoofing creates random ***backscatter.***

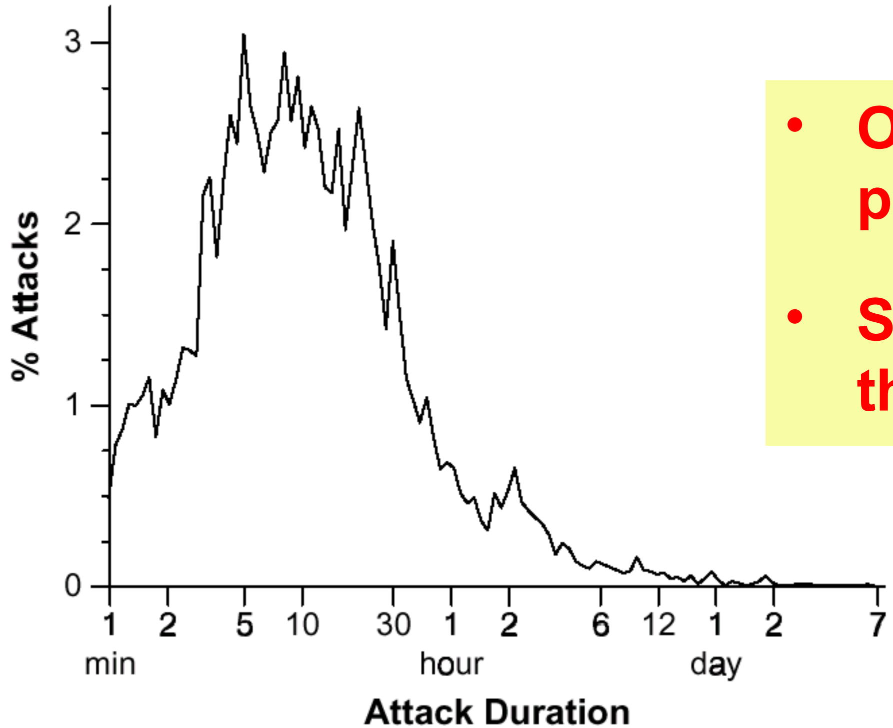


Backscatter Analysis

- Monitor block of n IP addresses
- Expected # of backscatter packets given an attack of m packets:
  - $E(X) = nm / 2^{32}$
  - Hence, $m = x * (2^{32} / n)$
- Attack Rate $R >= m/T = x/T * (2^{32} / n)$

Inferred DoS Activity





- **Over 4000 DoS/DDoS attacks per week**
- **Short duration: 80% last less than 30 minutes**

Moore *et al.* Inferring Internet Denial of Service Activity

DDoS: Setting up the Infrastructure

- Zombies
  - Slow-spreading installations can be difficult to detect
  - Can be spread quickly with **worms**

- Indirection makes attacker harder to locate
  - No need to spoof IP addresses

What is a Worm?

- Code that replicates and propagates across the network
  - Often carries a “payload”
- Usually spread via exploiting flaws in open services
  - “Viruses” require user action to spread
- **First worm:** Robert Morris, November 1988
  - 6-10% of all Internet hosts infected (!)
- Many more since, but none on that scale until July 2001

Example Worm: Code Red

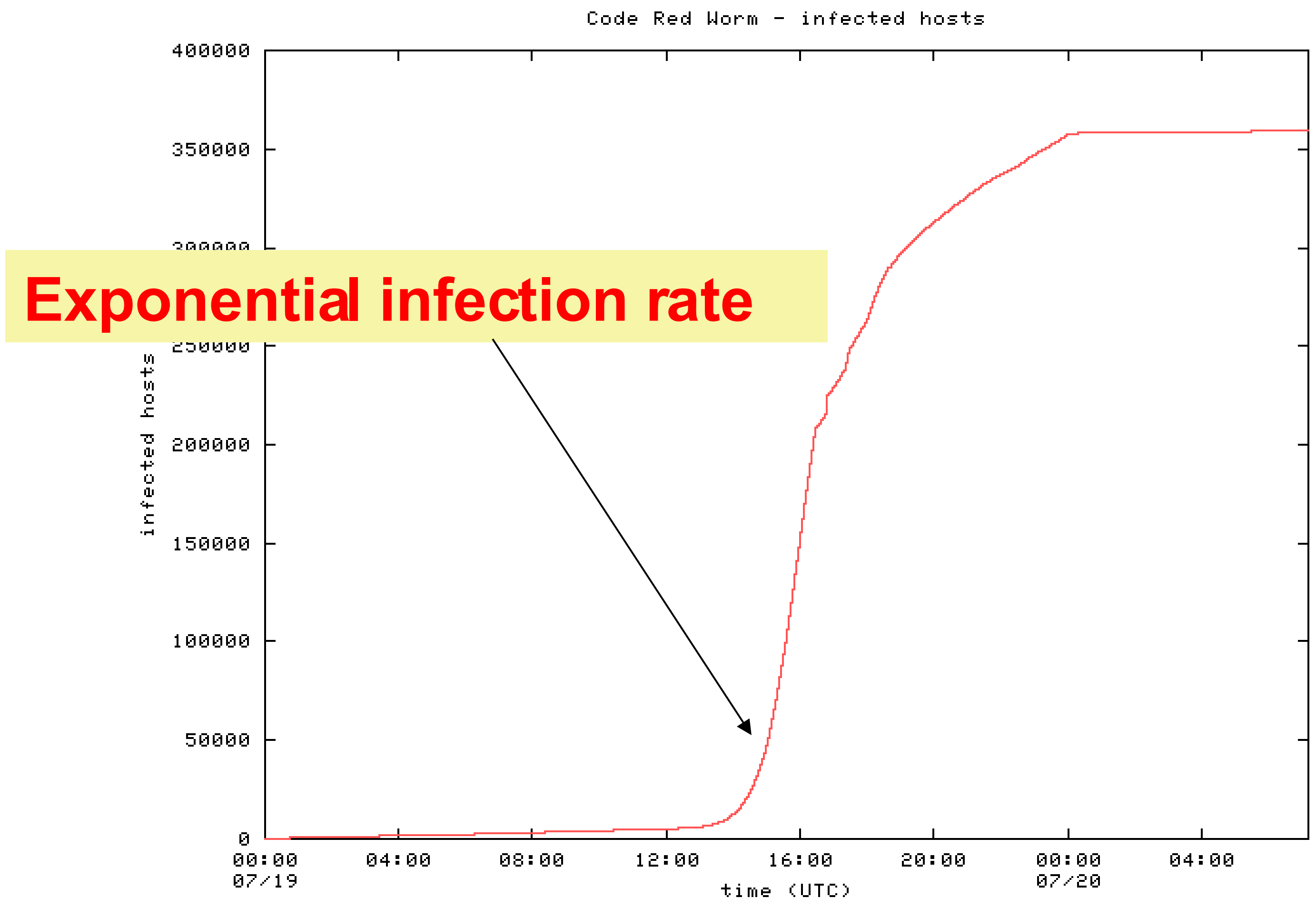
- Initial version: July 13, 2001
- Exploited known ISAPI vulnerability in Microsoft IIS Web servers
- 1st through 20th of each month: spread
  20th through end of each month: attack
- **Payload:** Web site defacement
- **Scanning:** Random IP addresses
- **Bug:** failure to seed random number generator

Code Red: Revisions

- Released July 19, 2001
- **Payload:** flooding attack on www.whitehouse.gov
  - Attack was mounted at the *IP address of the Web site*
- **Bug:** died after 20th of each month
- Random number generator for IP scanning fixed

Code Red: Host Infection Rate

Measured using backscatter technique



Modeling the Spread of Code Red

- Random Constant Spread model
  - *K: initial* compromise rate
  - *N:* number of vulnerable hosts
  - *a*: fraction of vulnerable machines already compromised

$$Nda = (Na)K(1 - a)dt$$

Newly infected machines in *dt*

Machines already infected

Rate at which uninfected machines are compromised

Bristling Pace of Innovation

Various improvements to increase the infection rate

- **Code Red 2:** August 2001
  - **Localized scanning**
  - Same exploit, different codebase
  - Payload: root backdoor

- **Nimda:** September 2001
  - Spread via **multiple exploits** (IIS vulnerability, email, CR2 root backdoor, copying itself over network shares, etc.)
  - Firewalls were not sufficient protection

Designing Fast-Spreading Worms

- **Hit-list scanning**
  - Time to infect first 10k hosts dominates infection time
  - **Solution:** Reconnaissance (stealthy scans, etc.)

- **Permutation scanning**
  - **Observation:** Most scanning is redundant
  - **Idea:** Shared permutation of address space. Start scanning from own IP address. Re-randomize when another infected machine is found.

- **Internet-scale hit lists**
  - *Flash worm:* complete infection within 30 seconds

Recent Advances: Slammer

- February 2003
- Exploited vulnerability in MS SQL server
- Exploit fit into a single UDP packet
  - ***Send and forget!***
- Lots of damage
  - BofA, Wash. Mutual ATMs unavailable
  - Continental Airlines ticketing offline
  - Seattle E911 offline

Scary recent advances: Witty

- March 19, 2004
- Single UDP packet exploits flaw in the *passive analysis* of Internet Security Systems products.
- “Bandwidth-limited” UDP worm ala’ Slammer.
- Initial spread seeded via a *hit-list*.
- All 12,000 vulnerable hosts infected within 45 mins
- **Payload:** slowly corrupt random disk blocks

Why does DDoS work?

- Simplicity
- “On by default” design
- Readily available zombie machines
- Attacks look like normal traffic
- Internet’s federated operation obstructs cooperation for diagnosis/mitigation

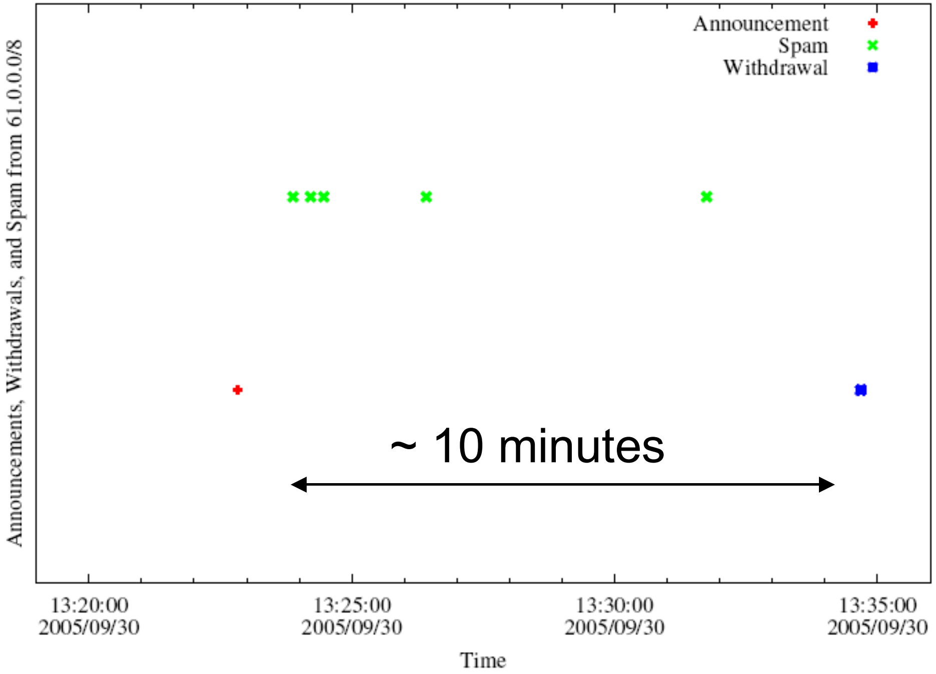
Resource Exhaustion: Spam

- Unsolicited commercial email
- As of about February 2005, estimates indicate that about 90% of all email is spam
- Common spam filtering techniques
  - Content-based filters
  - DNS Blacklist (DNSBL) lookups: Significant fraction of today's DNS traffic!

Can IP addresses from which spam is received be spoofed?

BGP Spectrum Agility

- Log IP addresses of SMTP relays
- Join with BGP route advertisements seen at network where spam trap is co-located.



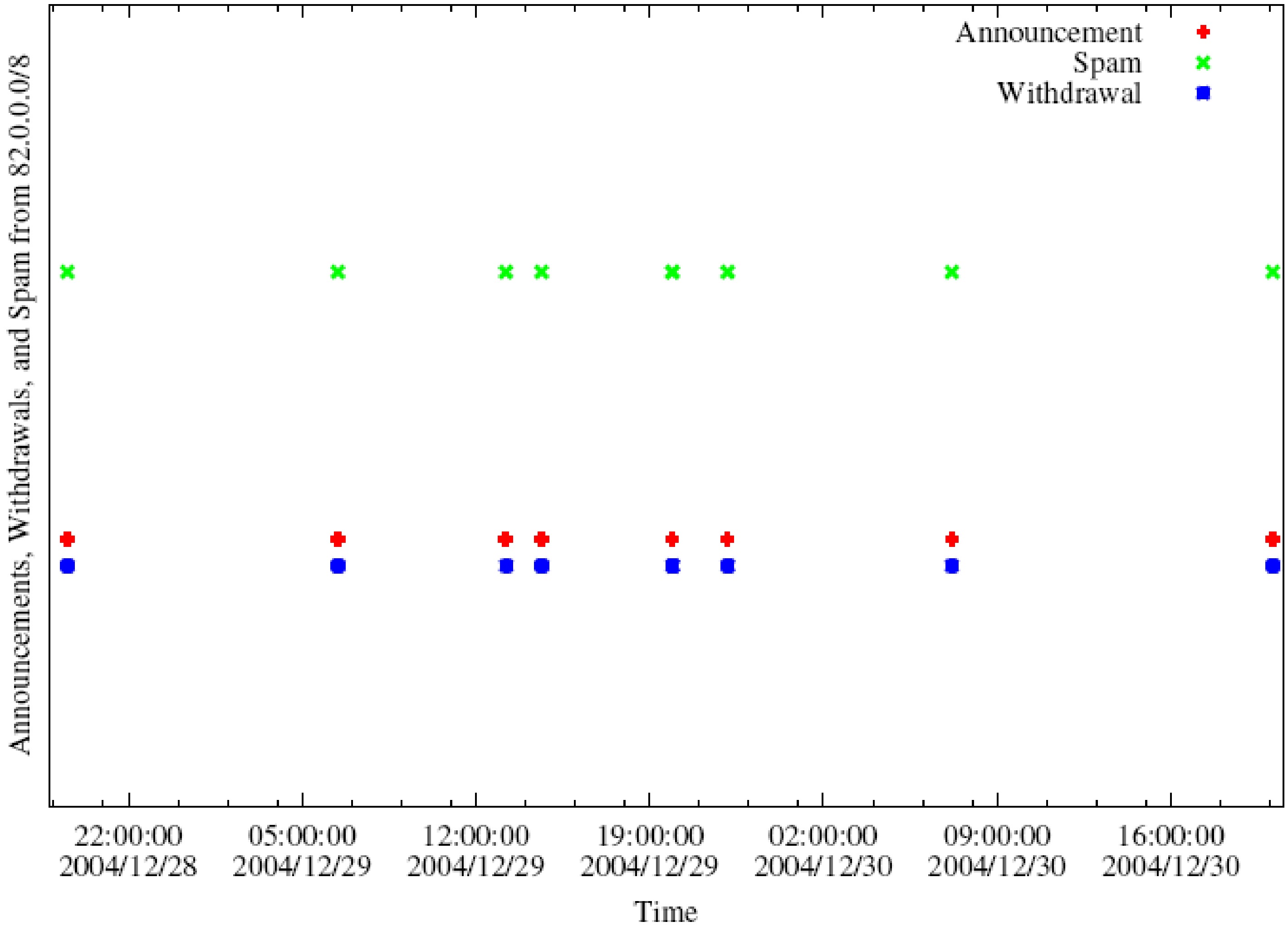
A small club of persistent players appears to be using this technique.

Common short-lived prefixes and ASes

61.0.0.0/8 4678
66.0.0.0/8 21562
82.0.0.0/8 8717

Somewhere between 1-10% of all spam (some clearly intentional, others might be flapping)

A Slightly Different Pattern



Why Such Big Prefixes?

- **Flexibility:** Client IPs can be scattered throughout dark space within a large /8
  - Same sender usually returns with different IP addresses

- **Visibility:** Route typically won't be filtered (nice and short)

Characteristics of IP-Agile Senders

- IP addresses are widely distributed across the /8 space
- IP addresses typically appear only once at our sinkhole
- Depending on which /8, 60-80% of these IP addresses were not reachable by traceroute when we spot-checked
- Some IP addresses were in *allocated*, albeing unannounced space
- Some AS paths associated with the routes contained reserved AS numbers

Some evidence that it’s working

Spam from IP-agile senders tend to be listed in fewer blacklists