Network Security: Internet Worms and Firewalls

Brad Karp

UCL Computer Science

CS 3035/GZ01

11th December 2014

Outline

- Internet worms
  - Self-propagating, possibly malicious code spread over Internet
- Firewalls:
  - Simple, perimeter-based security

What’s a Worm?

- Vast numbers of Internet-attached hosts run vulnerable server software
- Worm: self-replicating code, containing
  - Exploit for widely used, vulnerable server software
  - Payload: code that executes after exploit succeeds
- Payload connects to other Internet hosts, sends copy of {exploit, payload} to each…
- Unlike virus, spread not human-mediated

What’s in the Payload?

- Could be anything…arbitrary code execution allowed by many exploits
- Install login facility for attacker, to allow use at will in botnet
  - Botnets used widely today to launch DDoS attacks, send spam
  - Market in botnets exists today (3-10 US cents/host/week for spam proxy in 2005 [Paxson])
- Send sensitive files to attacker
- Destroy or corrupt data
- Enormous possibility for harm, in financial, privacy, and inconvenience terms

Code-RedI Worm

- June 18th, 2001: eEye releases description of buffer overflow vulnerability in Microsoft IIS (web server)
- June 26th, 2001: Microsoft releases patch
- July 12th, 2001: Code-RedI worm released (i.e., first sent to vulnerable host)
- Estimated number hosts infected: 360,000
- Estimated damages: $2.6 billion from loss of service availability, downtime, cleanup…

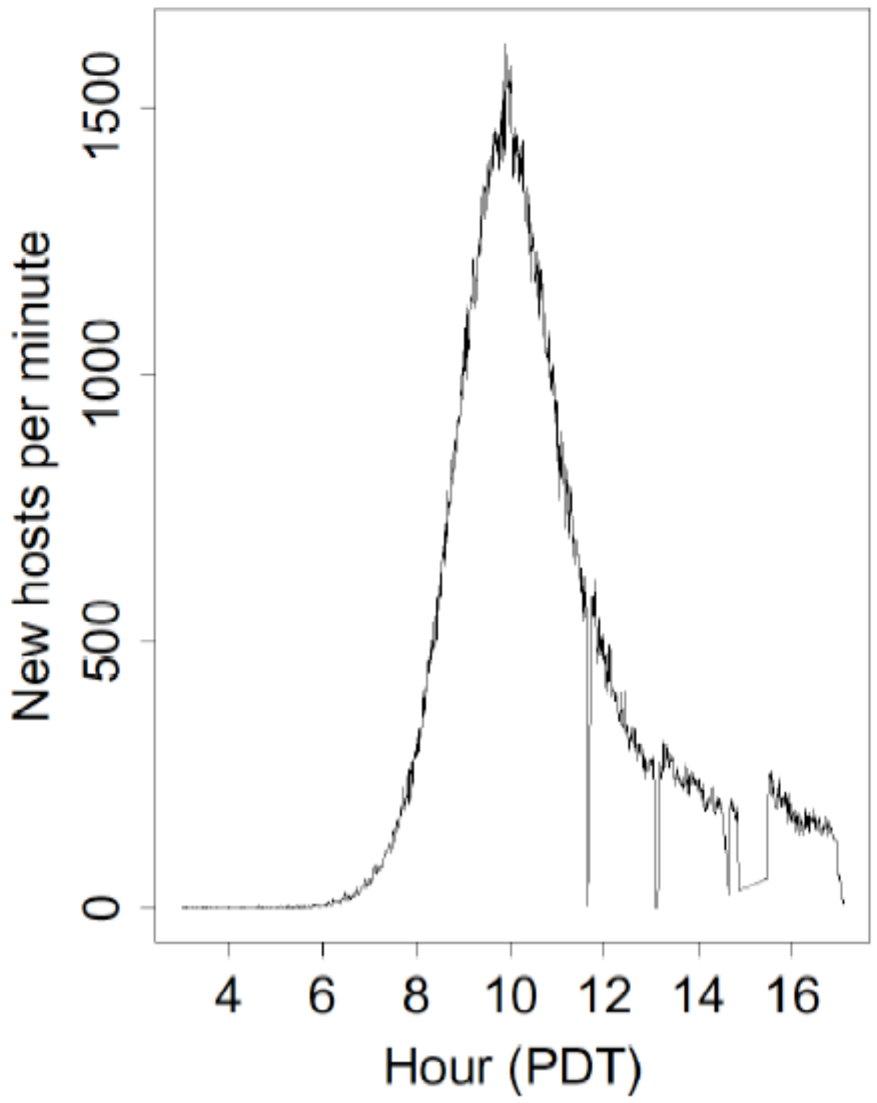
Code-RedI Behavior

- Payload: defaces web site
  - If language == English
    - HELLO! Welcome to http://www.worm.com! Hacked By Chinese!
- 1st – 19th of every month: spread
  - Connect to random 32-bit IP address, send copy of self (exploit+payload)
- 20th through end of every month:
  - Flood traffic to 198.137.240.91 (www.whitehouse.gov)
- Bug: fixed seed for random number generator
  - All hosts generate same sequence of IPs!
  - Result: only linear growth in infected population
- Only memory-resident; vanishes on reboot

Code-RedI v2: “Bugfix” Release

- July 19th, 2001: new variant (“v2”) released
  - Uses random seed
  - Now all infected hosts try different targets
- White House changes IP address of its server to avoid DDoS attack
  - Result: July 20th, Code-RedI v2 dies out
- 360K hosts infected in 14 hours

Growth of Code-RedI v2



- Source:
  Vern Paxson,
  ICSI/UC Berkeley

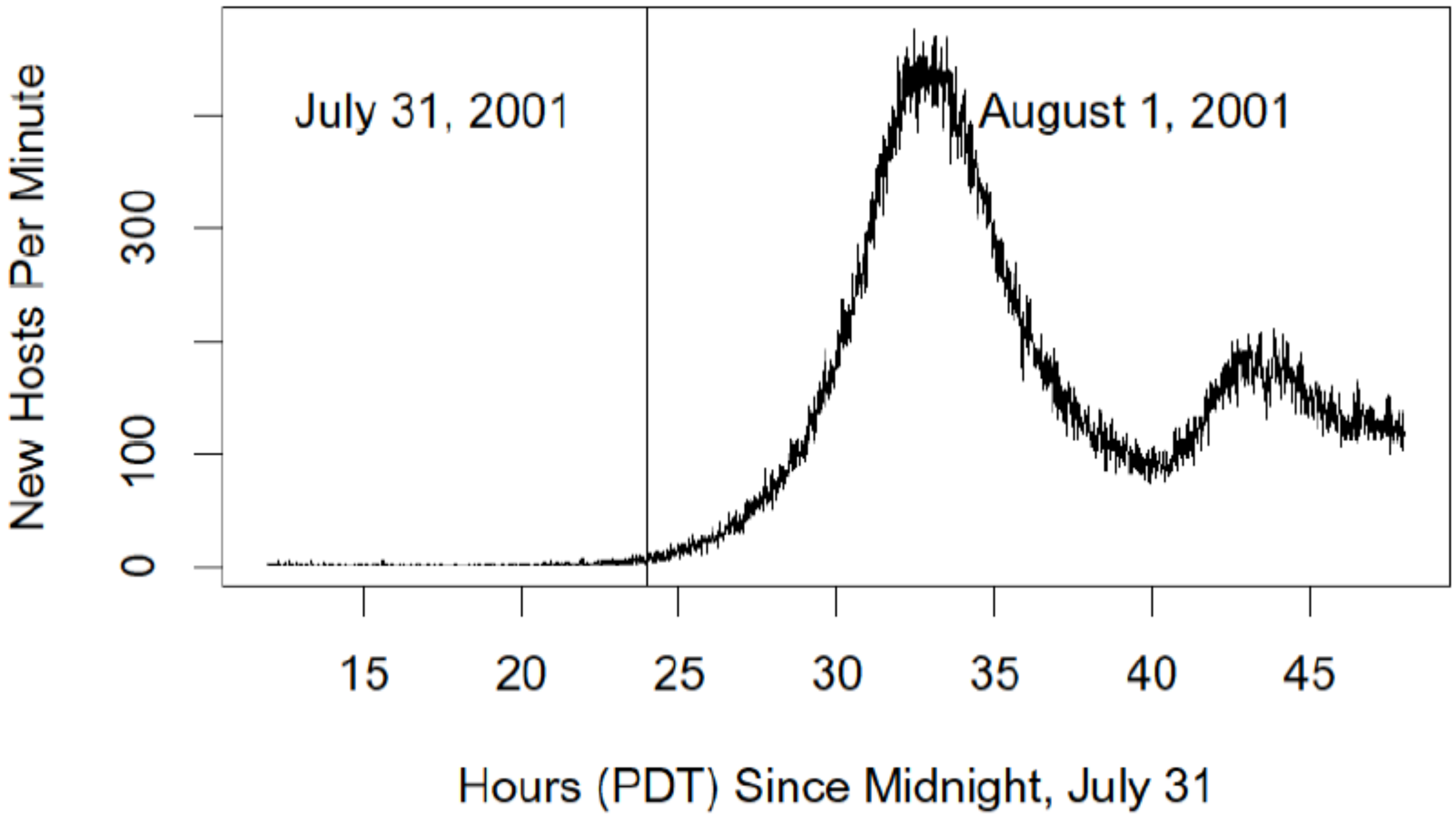
Network Telescopes

- Monitor traffic arriving at sizeable regions of Internet address space. Reveals, e.g.,:
  - "Backscatter" (responses to randomly source-spoofed DDoS attacks)
  - Worms' random scanning of IP addresses
  - Attackers' random scanning for servers running particular service
- LBNL: 2 /16 networks, or 1/32768th of Internet address space
- UCSD/Univ. Wisconsin: 1 /8 network, or 1/256th of Internet address space

Spread of Code-RedI v2

- Network telescope estimate of infected host count:
  - Count unique source IPs that attempt to connect to port 80 on non-used addresses
- Infected population over time fits logistic function
  - S-shaped curve: exponential growth at start, then slowing growth after most vulnerable nodes infected
- Worm dies just as 20th starts
  - But even one host with wrong clock can keep trying to infect others
  - On August 1st, worm begins to spread again!

Return of Code Red Worm



- Source: Vern Paxson, ICSI/UC Berkeley

A Competitor: Code-Red II

- Targets same IIS vulnerability; unrelated code
- Released August 4th, 2001
- Installs superuser backdoor; persists after reboot
- Spreads preferentially to local addresses:
  - ½ probability generates address on same /8
  - 3/8 probability generates address on same /16
  - 1/8 probabliity generates random non-class-D, non-loopback address
- Result: squeezes out Code-Red I v2!

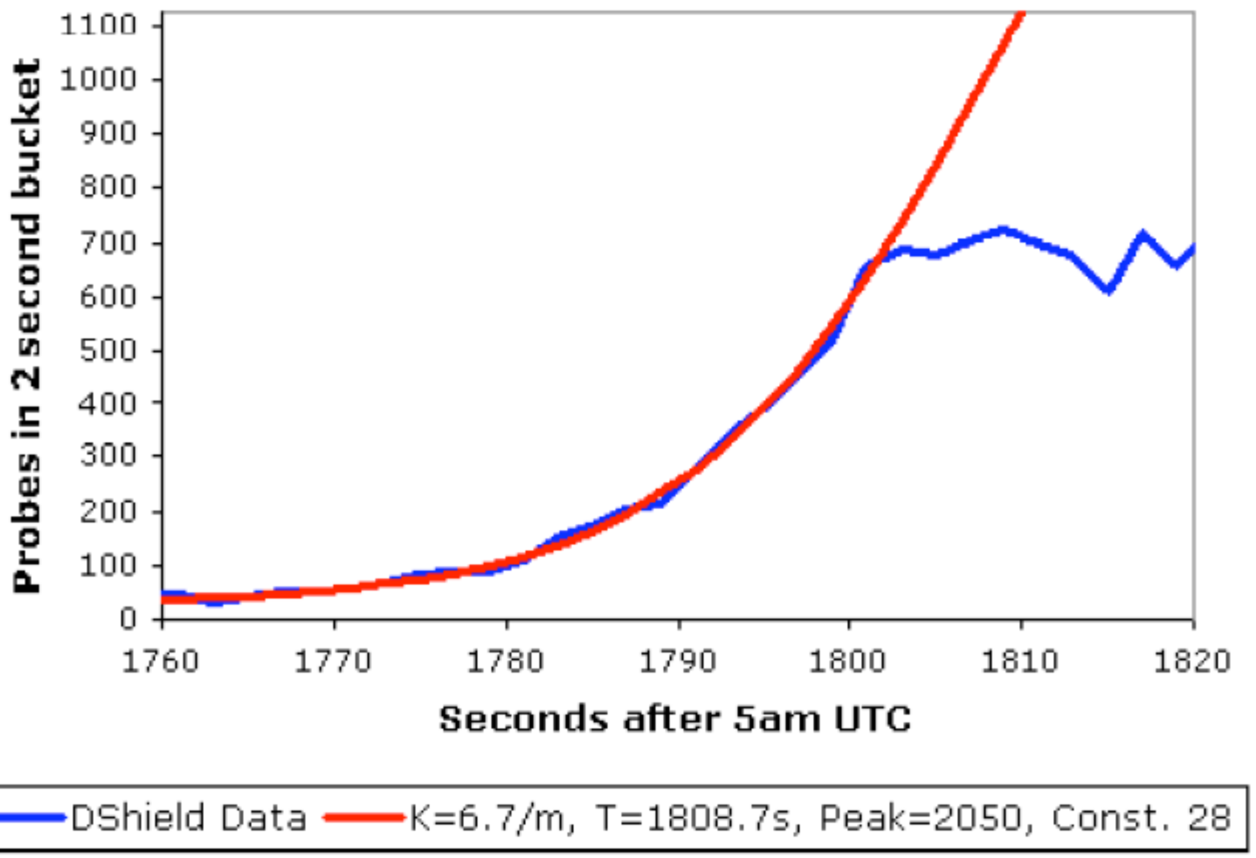
Slammer: A Fast UDP Worm

- Exploit: buffer overflow vulnerability in Microsoft SQL Server 2000
  - Vulnerability reported in June 2002
  - Patch released July 2002
- SQL service uses connectionless UDP (rather than connection-oriented TCP)
- Entire worm fit in one packet!
  - No need to wait for RTT; send single packet, try next target address
- Slammer infected over 75K hosts in 10 minutes
- Growth rate limited by Internet's capacity

Slammer's Behavior

- Peak address scanning rate: 55 million scans / second
  - Reached in 3 minutes
  - Beyond that point, congestion-limited
- Payload non-malicious, apart from aggressive scanning
- Outages in 911 (emergency telephone) service, Bank of America ATM network
  - Purely from traffic load; crashed some network equipment, saturated some bottleneck links

Slammer's Growth Limited by Internet Bandwidth (!)

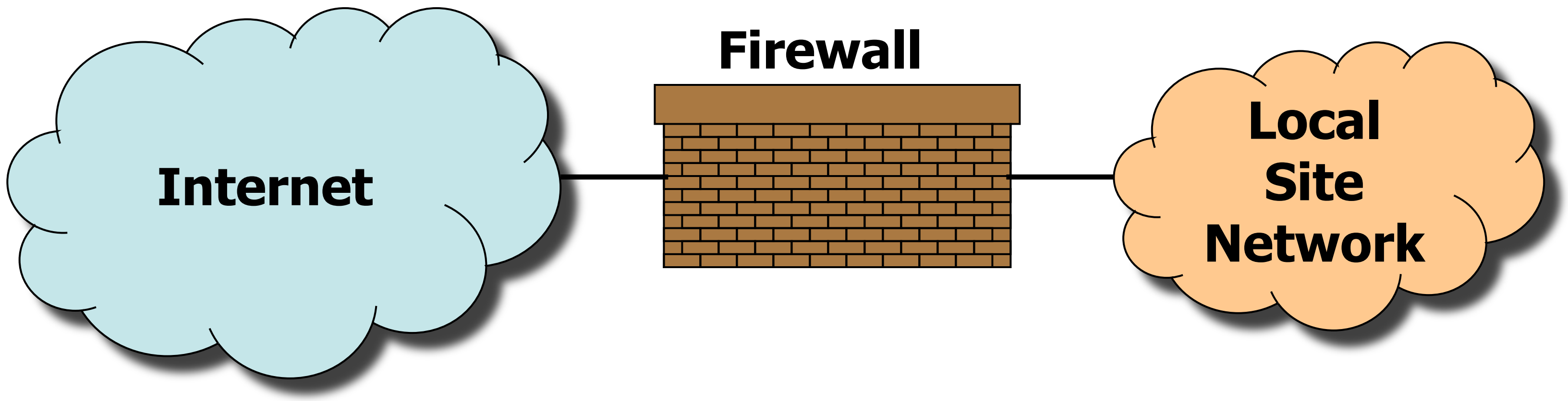


- Source: Vern Paxson, ICSI/UC Berkeley

Worm Propagation Methods

- Random scanning (e.g., Code-Red, Slammer)
- Meta-server worm: query a service for hosts to infect (e.g., ask Google, "powered by phpbb")
- Topological worm: find candidates from files on infected host's disk (e.g., web server logs, bookmark files, email address books, ssh known hosts files, …)
  - Very fast; stealthy—no random scanning behavior to attract attention
- Contagion worm: piggyback worm on application's usual connections
  - Connection patterns appear normal!

Firewalls: Perimeter-Based Defense



- Define trusted perimeter (typically boundary of own infrastructure)
- All packets between Internet and trusted perimeter flow through firewall
- Firewall inspects, filters traffic to limit access to non-secure services by remote, untrusted hosts

Firewall: Physical Topology vs. Filtering Policies

- Topological placement of firewall depends on perimeter at which defense desired, e.g.,
  - Firewall between company's net and Internet
  - Firewall between secret future product group's LAN and rest of company's net
  - Firewall A between Internet and public servers, firewall B between servers and rest of company's net
  - Software personal firewall on desktop machine
- Filtering policy depends on which attacks want to defend against, e.g.,
  - Packet filtering router
  - Application-level gateway (proxy for ftp, HTTP, &c.)
  - Personal firewall disallows Internet Explorer from making outbound SMTP connections

Background: Internet Services and Port Numbers

- Recall that UDP and TCP protocols identify service by destination 16-bit port number
- Well-known services: typically listen on ports <= 600
  - UNIX: must be root to listen on or send from port < 1024
- Outgoing connections typically use high source port numbers
  - App can ask OS to pick unused port number
- See **`/etc/services`** on UNIX host for list of well-known ports

Non-Secure Services

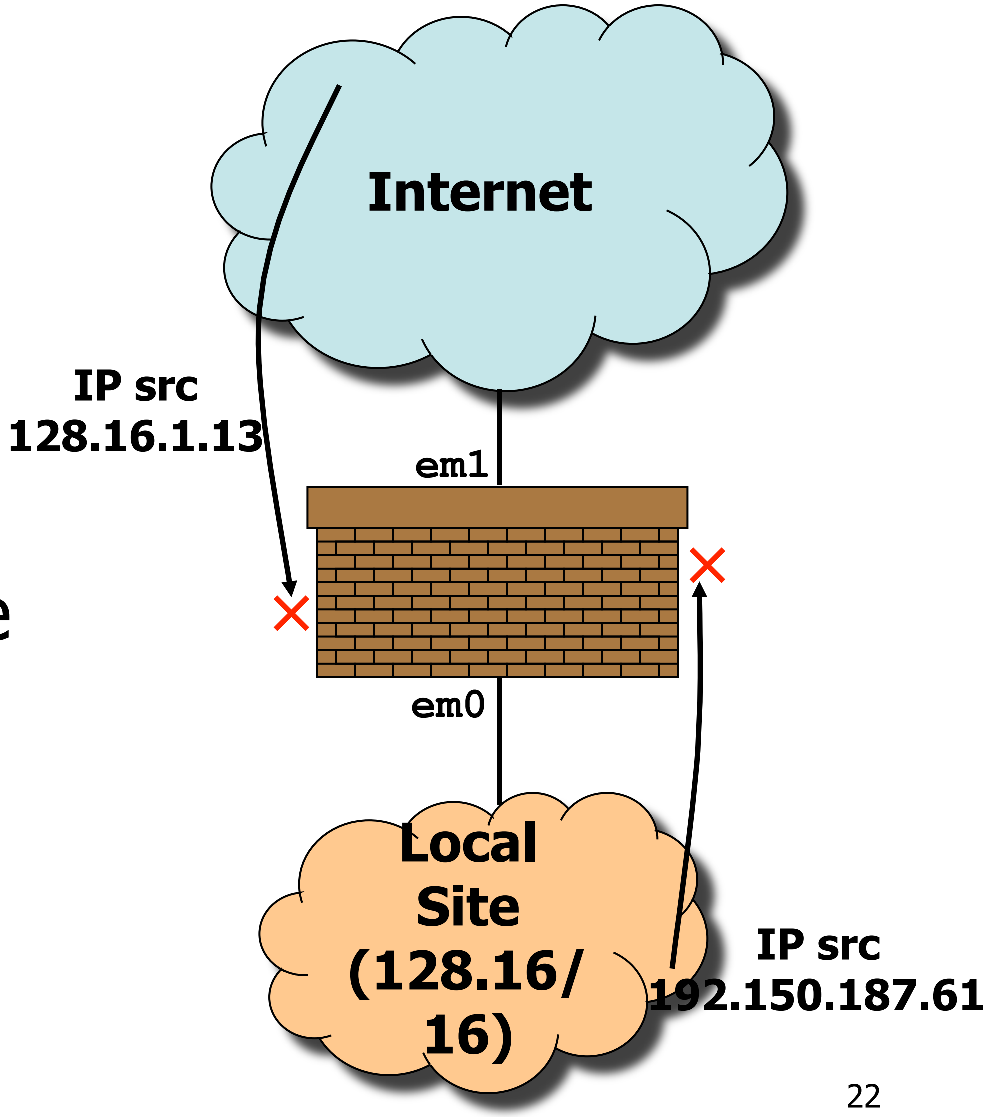
- NFS server (port 2049)
  - Recall: can read/write entire file system given file handle for any directory
  - File handles guessable on many platforms
- Portmap (port 111)
  - Relays RPC requests, so they appear to come from localhost
- FTP (port 21)
  - Client instructs server to connect to self; can instead direct server to connect to 3rd party (“bounce” attack)
- Yellow pages/NIS
  - Allows remote retrieval of password database
- Any server with a vulnerability
  - MS SQL (UDP 1434), DNS (53), rlogin (513), lpd (515), …

Firewalls: Packet Filtering

- Examine protocol fields of individual packets; filter according to rules
  - IP source, destination addresses
  - IP protocol ID
  - TCP/UDP source, destination ports
  - TCP packet flags (e.g., SYN, FIN, …)
  - ICMP message type
- Example: to prevent remote lpd exploit, block all inbound TCP packets to destination port 515
  - Remote users shouldn't be printing at your site anyway

Firewall Example: Blocking Source Spoofing

- Block traffic from outside your site with a source address in your site's address block
- Egress filtering: block traffic from within your site with a source address not in your site's address block
  - e.g., rule: `"deny ip not from 128.16/16 recv em0 xmit em1"`



Firewall Example: Blocking Outbound Mail

- Worms often use infected hosts to send spam or confidential documents
- Defense: authorize only a few servers at site to send outbound mail; filter all outbound mail connections from others
- e.g., rules:

```
allow tcp from 128.16.1.20 not to
128.16/16 dst-port 25
deny tcp from 128.16/16 not to
128.16/16 dst-port 25
```

Firewall Example: Block All Inbound Traffic by Default

- Little control over what software users run on desktops (including servers) at most sites
- May wish to avoid remote exploits of any software run on users' desktops
- Policy:
  - disallow all inbound TCP connections but those to known legitimate servers (e.g., one public web server, one mail server)
  - allow all outbound TCP connections
- Implementation:
  - Stateless way: drop all inbound TCP packets with SYN flag set, but not ACK flag

Stateful Firewalling

- Stateful way to implement "outbound TCP only":
  - Firewall stores state for every active TCP connection (src IP, src port, dst IP, dst port)
  - Only forwards "legal" packets for current state
    - e.g., if connection unknown, only allow outbound packets with SYN flag set, but not ACK flag
    - e.g., if connection known, only allow inbound packets with data after SYN/ACK seen
  - Time out connection state for long-idle connections
- Also used to block inbound UDP only
  - No standard SYN, ACK fields in UDP to support stateless filtering
- Risk: state memory exhaustion on firewall

Firewalling Complex Protocols

- Consider FTP
- Client connects to server, instructs server to open TCP connection back to client on specified client-side port
- Client's firewall won't allow inbound connection!
- One solution: application-level proxy
  - Client's firewall starts FTP application-level proxy upon detecting FTP session
  - Proxy on firewall acts as client for TCP connections with remote server, server for TCP connections with local client
  - Can enforce policy for many protocols (SMTP, HTTP, &c.)
  - But not used for encrypted protocols (SSL, SSH, &c.)