

# CS 6v81 - Network Security

*DoS Defense*

Source: modified from slides by Dan Boneh

## What is network DoS?

- ▶ Goal: Take out a large site with little computing work
- ▶ How: **Amplification**
  - ▶ Small number of packets ⇒ big effect
- ▶ Two types of amplification attacks:
  - ▶ DoS bug:
    - ▶ Design flaw allowing one machine to disrupt a service
  - ▶ DoS flood:
    - ▶ Command bot-net to generate flood of requests

2


## DoS can happen at any layer

- ▶ This lecture:
  - ▶ Sample DoS at different layers (by order):
    - ▶ Link
    - ▶ TCP/UDP
    - ▶ Application
    - ▶ Payment
  - ▶ Some generic DoS solutions
  - ▶ Some network DoS solutions
- ▶ Sad truth:
  - ▶ Current Internet not designed to handle DDoS attacks

3

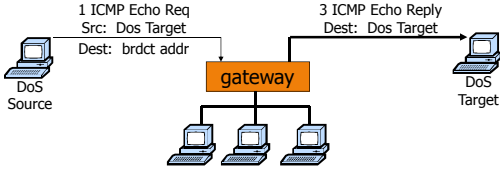
## Warm up: 802.11b DoS bugs

- ▶ Radio jamming attacks: trivial, not our focus.
- ▶ Protocol DoS bugs: [Bellardo, Savage, '03]
  - ▶ NAV (Network Allocation Vector):
    - ▶ 15-bit field. Max value: 32767
    - ▶ Any node can reserve channel for NAV seconds
    - ▶ No one else should transmit during NAV period
    - ▶ ... but not followed by most 802.11b cards
  - ▶ De-association bug:
    - ▶ Any node can send disassociate packet to AP
    - ▶ Disassociate packet unauthenticated
    - ▶ ... attacker can repeatedly disassociate anyone



4

## Smurf amplification DoS attack



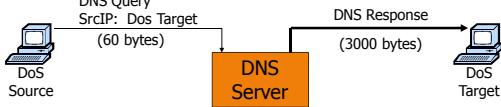
- ▶ Send ping request to broadcast addr (ICMP Echo Req)
- ▶ Lots of responses:
  - ▶ Every host on target network generates a ping reply (ICMP Echo Reply) to victim

Prevention: reject external packets to broadcast address

5

## Modern day example (May '06)

### DNS Amplification attack: ( ×50 amplification )



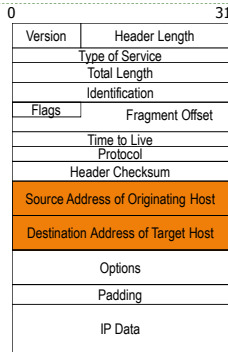
580,000 open resolvers on Internet (Kaminsky-Shiffman'06)

Prevention: reject DNS queries from external addresses

6

## Review: IP Header format

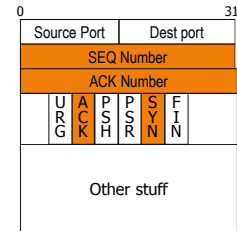
- ▶ Connectionless
  - ▶ Unreliable
  - ▶ Best effort



7

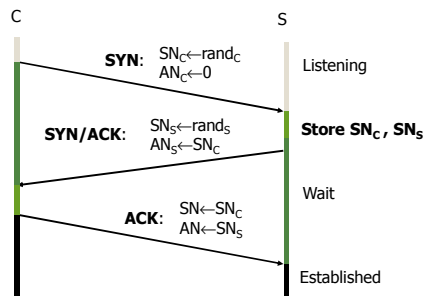
## Review: TCP Header format

- ▶ TCP:
  - ▶ Session based
  - ▶ Congestion control
  - ▶ In order delivery

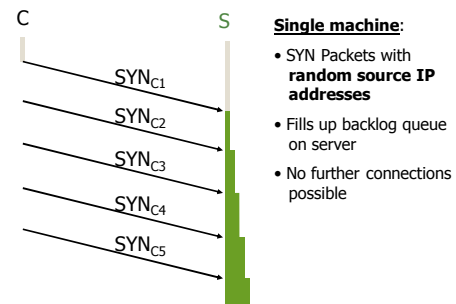


8

## Review: TCP Handshake



## TCP SYN Flood I: low rate (DoS bug)



## SYN Floods

OS	Backlog queue size
<b>Linux 1.2.x</b>	10
<b>FreeBSD 2.1.5</b>	128
<b>WinNT 4.0</b>	6

Backlog timeout: 3 minutes

- ⇒ Ideally: attacker need only send 128 SYN packets every 3 minutes.
- ⇒ Low rate SYN flood

## A classic SYN flood example

- ▶ **MS Blaster worm** (2003)
  - ▶ Infected machines at noon on Aug 16<sup>th</sup>:
    - ▶ SYN flood on port 80 to [windowsupdate.com](http://windowsupdate.com)
    - ▶ 50 SYN packets every second.
      - each packet is 40 bytes.
    - ▶ Spoofed source IP: a.b.X.Y where X,Y random.
  - ▶ **MS solution:**
    - ▶ New name: [windowsupdate.microsoft.com](http://windowsupdate.microsoft.com)
    - ▶ Win update file delivered by Akamai

12

## Low rate SYN flood defenses

- ▶ Non-solution:
  - ▶ Increase backlog queue size or decrease timeout
- ▶ Correct solution:
  - ▶ **Syncookies**: remove state from server
  - ▶ Small performance overhead

13

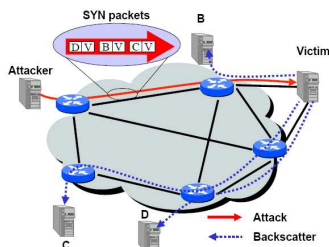
## Syncookies [Bernstein, Schenk]

- ▶ Idea: Remove SYN state from server
- ▶ Server responds to Client with SYN-ACK cookie:
  - ▶ T = 5-bit counter incremented every 64 secs.
  - ▶ M = 3-bit encoding of MSS
  - ▶  $L = F_{key}(SAddr, SPort, DAddr, DPort, T) + SN_C$ 
    - ▶ In practice,  $F_{key}(X) = MD5(\text{key} || X || \text{key})$
    - ▶ Key: picked at random during boot
  - ▶  $SN_S = (T || M || L)$
  - ▶ Server does not save state
- ▶ Honest client responds with ACK(AN=SN<sub>S</sub>)
  - ▶ Server allocates space for socket only if valid SN<sub>S</sub>.

14

## SYN floods: backscatter [MVS'01]

- ▶ SYN with forged source IP ⇒ SYN/ACK to random host



15

## Backscatter measurement [MVS'01]

- ▶ Listen to unused IP address space
  - ▶  $0$  to  $2^{32}$  /8 network monitor
- ▶ Lonely SYN/ACK packet likely to be result of SYN attack
- ▶ In 2001: found about 400 SYN attacks/week
  - ▶ Larger experiments:
    - ▶ Internet motion sensor (U.Mich)
    - ▶ Network telescope (UCSD)

16

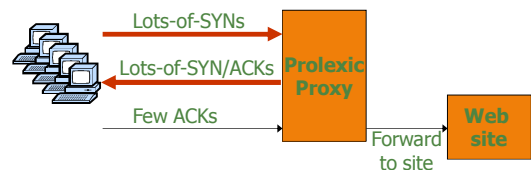
## SYN Floods II: Massive flood (e.g BetCris.com '03)

- ▶ Command bot army to flood specific target: (DDoS)
  - ▶ **20,000** bots can generate **2Gb/sec** of SYNs (2003)
- ▶ At web site:
  - ▶ Saturates network uplink or network router
  - ▶ Random source IP ⇒ attack SYNs look the same as real SYNs
- ▶ What to do ???

17

## Prolexic

- ▶ Idea: only forward established TCP connections to site



- ▶ Prolexic capacity: 20Gb/sec link  
can handle  $40 \cdot 10^6$  SYN/sec

18

## Other junk packets

Attack Packet	Victim Response
TCP SYN to open port	TCP SYN/ACK
TCP SYN to closed port	TCP RST
TCP ACK or TCP DATA	TCP RST
TCP RST	No response
TCP NULL	TCP RST
ICMP ECHO Request	ICMP ECHO Response
UDP to closed port	ICMP Port unreachable

- ▶ Proxy must keep floods of these away from web site

19

## Obvious next step: TCP con flood

- ▶ Command bot army to:
  - ▶ Complete TCP connection to web site
  - ▶ Send short HTTP HEAD request
  - ▶ Repeat
- ▶ Will bypass SYN flood protection proxy
- ▶ ... but:
  - ▶ Attacker can no longer use random source IPs.
    - ▶ Reveals location of bot zombies
  - ▶ Proxy can now block or rate-limit bots.

20

## DNS DoS Attacks (e.g. bluesecurity '06)

- ▶ DNS runs on UDP port 53
  - ▶ DNS entry for victim.com hosted at victim\_isp.com
- ▶ DDoS attack:
  - ▶ flood victim\_isp.com with requests for victim.com
  - ▶ **Random source IP address** in UDP packets
- ▶ Takes out entire DNS server: (collateral damage)
  - ▶ bluesecurity DNS hosted at Tucows DNS server
  - ▶ DNS DDoS took out Tucows hosting many many sites
- ▶ What to do ???

21

## Root level DNS attacks

- ▶ **Feb. 6, 2007:**
  - ▶ Botnet attack on the 13 Internet DNS root servers
  - ▶ Lasted 2.5 hours
  - ▶ None crashed, but two performed badly:
    - ▶ g-root (DoD), l-root (ICANN)
    - ▶ Most other root servers use anycast

22

## DNS DoS solutions

- ▶ Generic DDoS solutions:
  - ▶ Later on. require major changes to DNS.
- ▶ DoS resistant DNS design:
  - ▶ **CoDoNS:** [Srirer'04]
    - ▶ Cooperative Domain Name System
  - ▶ P2P design for DNS system:
    - ▶ DNS nodes share the load
    - ▶ Simple update of DNS entries
    - ▶ Backwards compatible with existing DNS

23

## DoS at higher layers

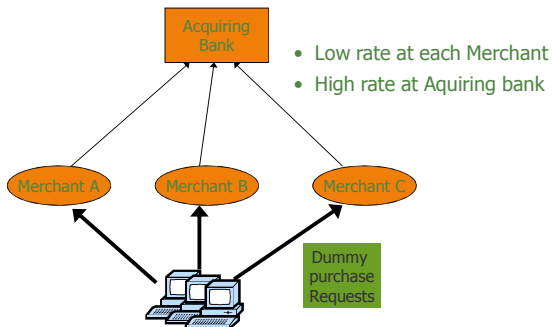
- ▶ SSL/TLS handshake [SD'03]



- ▶ RSA-encrypt speed  $\approx 10\times$  RSA-decrypt speed
  - ⇒ Single machine can bring down ten web servers
- ▶ Similar problem with application DoS:
  - ▶ Send HTTP request for some large PDF file
  - ⇒ Easy work for client, hard work for server.

24

## Payment DDoS



25

## DoS Mitigation

## 1. Client puzzles

- ▶ Idea: slow down attacker
- ▶ Moderately hard problem:
  - ▶ Given challenge C find X such that
$$\text{LSB}_n(\text{SHA-1}(C \parallel X)) = 0^n$$
  - ▶ Assumption: takes expected  $2^n$  time to solve
  - ▶ For  $n=16$  takes about .3sec on 1GHz machine
  - ▶ Main point: checking puzzle solution is easy.
- ▶ During DoS attack:
  - ▶ Everyone must submit puzzle solution with requests
  - ▶ When no attack: do not require puzzle solution

27

## Examples

- ▶ TCP connection floods (RSA '99)
  - ▶ Example challenge:  $C = \text{TCP server-seq-num}$
  - ▶ First data packet must contain puzzle solution
    - ▶ Otherwise TCP connection is closed
- ▶ SSL handshake DoS: (SD'03)
  - ▶ Challenge C based on TLS session ID
  - ▶ Server: check puzzle solution before RSA decrypt.
- ▶ Same for application layer DoS and payment DoS.

28

## Benefits and limitations

- ▶ Hardness of challenge:  $n$ 
  - ▶ Decided based on DoS attack volume.
- ▶ Limitations:
  - ▶ Requires changes to both clients and servers
  - ▶ Hurts low power legitimate clients during attack:
    - ▶ Clients on cell phones, PDAs cannot connect

29

## Memory-bound functions

- ▶ CPU power ratio:
  - ▶ high end server / low end cell phone = 8000
  - ⇒ Impossible to scale to hard puzzles
- ▶ Interesting observation:
  - ▶ Main memory access time ratio:
    - ▶ high end server / low end cell phone = 2
- ▶ Better puzzles:
  - ▶ Solution requires many main memory accesses
    - ▶ Dwork-Goldberg-Naor, Crypto '03
    - ▶ Abadi-Burrows-Manasse-Wobber, ACM ToIT '05

30

## 2. CAPTCHAs

- ▶ Idea: verify that connection is from a human



- ▶ Applies to application layer DDoS [Killbots '05]
  - ▶ During attack: generate CAPTCHAs and process request only if valid solution
  - ▶ Present one CAPTCHA per source IP address.

31

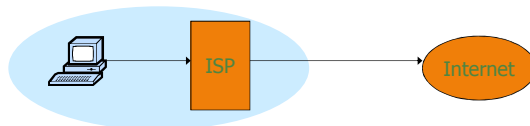
## 3. Source identification

Goal: identify packet source  
Ultimate goal: block attack at the source

32

## 1. Ingress filtering (RFC 2827, 2000)

- ▶ Big problem: DDoS with spoofed source IPs
- ▶ Question: how to find packet origin?

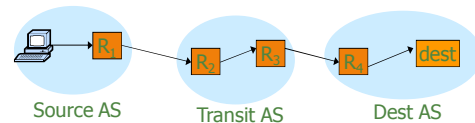


- ▶ Ingress filtering policy: ISP only forwards packets with legitimate source IP.

33

## Implementation problems

- ▶ ALL ISPs must do this. Requires global trust.
  - ▶ If 10% of ISPs do not implement  $\Rightarrow$  no defense
- ▶ Another non-solution: enforce source IP at peer AS



- ▶ Can transit AS validate packet source IP? No ...

34

## 2. Traceback [Savage et al. '00]

- ▶ Goal:
  - ▶ Given set of attack packets
  - ▶ Determine path to source
- ▶ How: change routers to record info in packets
- ▶ Assumptions:
  - ▶ Most routers remain uncompromised
  - ▶ Attacker sends many packets
  - ▶ Route from attacker to victim remains relatively stable

35

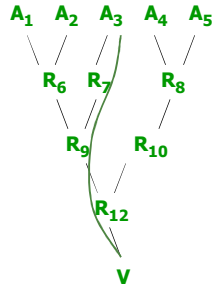
## Simple method

- ▶ Write network path into packet
  - ▶ Each router adds its own IP address to packet
  - ▶ Victim reads path from packet
- ◆ Problem:
  - Requires space in packet
    - ◆ Path can be long
    - ◆ No extra fields in current IP format
      - Changes to packet format too much to expect

36

## Better idea

- DDoS involves many packets on same path
- Store one link in each packet
  - Each router probabilistically stores own address
  - Fixed space regardless of path length



37

## Edge Sampling

- Data fields written to packet:
  - Edge: *start* and *end* IP addresses
  - Distance: number of hops since edge stored
- Marking procedure for router R
  - if coin turns up heads (with probability p) then
    - write R into start address
    - write 0 into distance field
  - else
    - if distance == 0 write R into end field
    - increment distance field

38

## Edge Sampling: picture

- Packet received
  - R<sub>1</sub> receives packet from source or another router
  - Packet contains space for start, end, distance



39

## Edge Sampling: picture

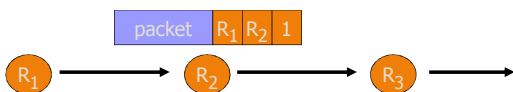
- Begin writing edge
  - R<sub>1</sub> chooses to write start of edge
  - Sets distance to 0



40

## Edge Sampling

- Finish writing edge
  - R<sub>2</sub> chooses not to overwrite edge
  - Distance is 0
    - Write end of edge, increment distance to 1



41

## Edge Sampling

- Increment distance
  - R<sub>3</sub> chooses not to overwrite edge
  - Distance > 0
    - Increment distance to 2



42

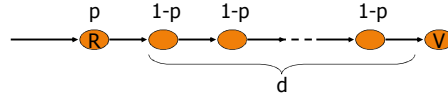
## Path reconstruction

- ▶ Extract information from attack packets
- ▶ Build graph rooted at victim
  - ▶ Each (start,end,distance) tuple provides an edge
- ▶ # packets needed to reconstruct path
 
$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}$$
 where  $p$  is marking probability,  $d$  is length of path

43

## Node Sampling?

- ▶ Less data than edge sampling
  - ▶ Each router writes own address with probability  $p$
- ▶ Infer order by number of packets
  - ▶ Router at distance  $d$  has probability  $p(1-p)^d$  of showing up in marked packet

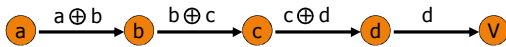


- ◆ Problems
  - Need many packets to infer path order
  - Does not work well if many paths

44

## Reduce Space Requirement

- ▶ XOR edge IP addresses
  - ▶ Store edge as  $\text{start} \oplus \text{end}$
  - ▶ Work backwards to get path:
 
$$(\text{start} \oplus \text{end}) \oplus \text{end} = \text{start}$$
- ▶ Sample attack path



45

## Details: where to store edge

- ▶ Identification field
  - ▶ Used for fragmentation
  - ▶ Fragmentation is rare
  - ▶ 16 bits
- ▶ Store edge in 16 bits?
 

offset	distance	edge chunk
0	2 3	7 8
		15

  - ▶ Break into chunks
  - ▶ Store  $\text{start} \oplus \text{end}$

Version	Header Length
Type of Service	
Total Length	
Identification	
Flags	Fragment Offset
Time to Live	
Protocol	
Header Checksum	
Source Address of Originating Host	
Destination Address of Target Host	
Options	
Padding	
IP Data	

46

## More traceback proposals

- ▶ Advanced and Authenticated Marking Schemes for IP Traceback
  - ▶ Song, Perrig. IEEE Infocomm '01
  - ▶ Reduces noisy data and time to reconstruct paths
- ▶ An algebraic approach to IP traceback
  - ▶ Stubblefield, Dean, Franklin. NDSS '02
- ▶ Hash-Based IP Traceback
  - ▶ Snoeren, Partridge, Sanchez, Jones, Tchakountio, Kent, Strayer. SIGCOMM '01

47

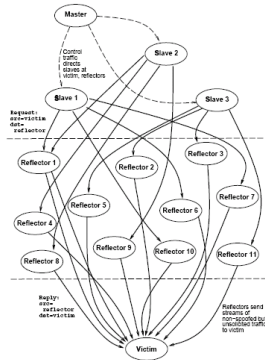
## Problem: Reflector attacks [Paxson '01]

- ▶ **Reflector:**
  - ▶ A network component that responds to packets
  - ▶ Response sent to victim (spoofed source IP)
- ▶ **Examples:**
  - ▶ DNS Resolvers: UDP 53 with victim.com source
    - ▶ At victim: DNS response
  - ▶ Web servers: TCP SYN 80 with victim.com source
    - ▶ At victim: TCP SYN ACK packet
  - ▶ Gnutella servers

48

## DoS Attack

- ▶ Single Master
- ▶ Many bots to generate flood
- ▶ Zillions of reflectors to hide bots
  - ▶ Kills traceback and pushback methods



## Capability based defense

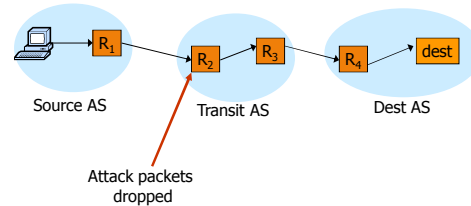
## Capability based defense

- ▶ Basic idea:
  - ▶ Receivers can specify what packets they want
- ▶ How:
  - ▶ Sender requests capability in SYN packet
    - ▶ Path identifier used to limit # reqs from one source
  - ▶ Receiver responds with capability
  - ▶ Sender includes capability in all future packets
  - ▶ **Main point:** Routers only forward:
    - ▶ Request packets, and
    - ▶ Packets with valid capability

51

## Capability based defense

- ▶ Capabilities can be revoked if source is attacking
  - ▶ Blocks attack packets close to source

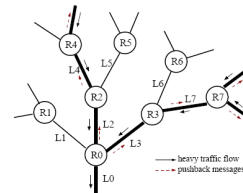


52

## Pushback Traffic Filtering

## Pushback Traffic Filtering

- ▶ Assumption: DoS attack from few sources



- ▶ Iteratively block attacking network segments.

54

53

### Take home message:

---

- ▶ Denial of Service attacks are real.  
Must be considered at design time.
  
- ▶ Sad truth:
  - ▶ Current Internet is ill-equipped to handle DDoS attacks
  
- ▶ Many good proposals for core redesign.