COMP 290-040
Network Intrusion Detection

Denial of Service
Brief History & Introduction to Defenses

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
jeffay@cs.unc.edu
January 26, 2005

http://www.cs.unc.edu/~jeffay/courses/nidsS05

Denial of Service
Timeline

- Early 1980s viruses spread through floppy disks
  - » Virus: A piece of code that adds itself to other programs, including operating systems.
  - » A virus cannot execute independently— it requires that its "host" program be executed
- 1988: Robert T. Morris releases first major Internet worm
  - » (Worm: a program that can run by itself and can propagate a fully working version of itself to other machines)
  - » Most Sun & VAX workstations disabled (~6,000 victims)
  - » Damage estimate: 1- 98 million dollars (*i.e.*, noone knows)
  - » Computer Emergency Response Team (CERT) founded at CMU's Software Engineering Institute

Quick Case Study:
The '88 RTM Internet Worm

- The worm acquired a shell on a remote machine
  - » Exploited well-known holes in system calls and services
- It attempted to masquerade as other users
  - » Attempted to crack passwords using (really) simple strategies
- For each user, the worm attempted to replicate itself on peer machines
  - » Connect to machines in *.forward* and *.rsh* files
- The worm tried to persist and hide its tracks
  - » Periodically fork itself and delete its parent

Quick Case Study:
The '88 RTM Internet Worm

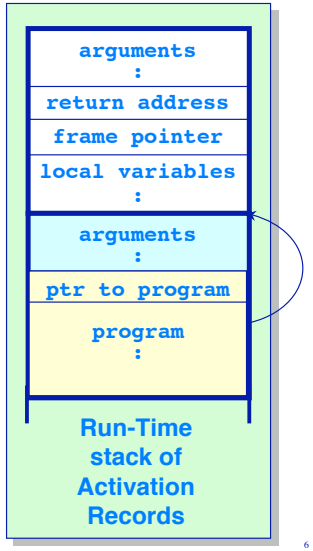
- The worm exploited "misconfigured" *sendmail* installations and ancient string overflow problems in standard I/O library
- *sendmail* was a notoriously hard program to manage
  - » "Stories are often related about how system administrators will attempt to write new device drivers or otherwise modify the kernel of the OS, yet they will not willingly attempt to modify sendmail or its configuration files" [Spafford 88]
- In "debug" mode, *sendmail* would execute commands "mailed" to the system
  - » Remote user could obtain a shell on victim machine and send it commands over the *sendmail* connection

Quick Case Study:
The '88 RTM Internet Worm

- Alternatively, an attacking machine would attempt to gain a shell on the victim machine via the *finger* daemon
  - » A (very) simple socket interface to the local *finger* program (a program to get information about local users)
- By passing the finger daemon a carefully crafted string (containing an executable program), a buffer overflow and transfer of control to the executable program was possible
  - » The hidden program could then start a shell and wreak havoc

The '88 RTM Internet Worm
The buffer overflow exploit

- On x86 machines stacks grow from high addresses to low addresses
- If a buffer is allocated as a local variable and overwritten, the stack frame can be corrupted
- If the contents of the buffer contain executable code and you overwrite the buffer to replace the return address…
- …on function return the machine will execute the program contained in the buffer



The '88 RTM Internet Worm
Worm structure

- Worm consisted of a main program and a bootstrap or vector program
- Using *fingerd* or *sendmail* exploit, attacking machine creates a shell on the victim machine
  - » Shell creates a socket back to attacker (*fingerd* case)
  - » Shell receives commands from attacker via SMTP (*sendmail* case)
- Vector program copied to victim and compiled
- Vector connects back to "server" and copies Sun & VAX binary versions of worm plus vector program source
  - » Challenge/response exchange used to authenticate vector and server

The '88 RTM Internet Worm
Worm structure (2)

- Vector program attempts to execute the worm binaries
- Worm then unlinks its binary and kills its parent process…
  - » Munges its *argv* string
- …reads the Sun & VAX worm binaries into memory
  - » Encrypts the files in memory
  - » Deletes the files from disk
- …and attempts to infect other machines
  - » Get information about network interfaces
  - » Contact machines thought to be on same LAN via *rsh*, *fingerd*, or *sendmail*
- Read */etc/passwd*, attempt simple crack, goto step 3

The '88 RTM Internet Worm
Denial-of-Service

- Machines quickly saturated with worm processes
  - » Machines on the local network continually re-infected each other ("bug" in the worm!)
- To ensure processes were not "niced," worm would periodically *fork* itself and kill its parent
  - » Also served to generate a fresh process ID
- Each worm process flushed its list of infected hosts every 12 hours and started anew
- Worm spread exponentially
- Machines ran out of processes or swap space and either crashed or were inaccessible

The '88 RTM Internet Worm
Killing the worm

- Strange activity was discovered nearly immediately
  - » Graduate students are the best IDS!
- Within 12 hours the CSRG folks at Berkeley had captured a copy of the worm and published workarounds
  - » Many folks remained in the dark because their computers were unusable or disconnected from the network
- Simplest stopgap measure: rename the C compiler and loader
  - » Patches to *sendmail*, *fingerd*, and *libc.a* followed within 24 hours

Denial of Service
Timeline (Continued)

- 1992: Widespread use of "root kits" appear
  - » Log in to system as real user, use exploit to become root
    - ❖ Gain password by "sniffing," or "social engineering"
  - » Replace system binaries with compromised versions to enable remote access
- 1996: First TCP SYN attacks appear
- 1997:
  - » Large scale IRC DoS attacks occur
    - ❖ *teardrop*, *boink*, *bonk* programs released
    - ❖ All exploited bugs in the Windows TCP/IP stack
    - ❖ (Bugs fixed but are PDAs/cell phones still vulnerable?)
  - » Internet taken down by 1 (non-maliciously) misconfigured router
  - » First pure flooding attacks (*smurf*) appear

Denial of Service
Timeline (3)

- 1998 (High-speed Internet access becoming common):
  - » Automated DoS tools appear
  - » New TCP/IP exploits appear (fragmented packets)
    - ❖ Entire organizations experience simultaneous BSoD
- 1999:
  - » Large-scale usage of DDoS tools (*trinoo*, *TFN*, *stacheldraht*)
    - ❖ All used handler/agent architecture
    - ❖ IRC clients/servers primary targets
    - ❖ Univ. of Minnesota offline for 3 days by attack from $O(2{,}500)$ machines
  - » CERT holds first workshop on "Distributed System Intruder Tools"
  - » First attacks seen using encrypted communications and load balancing

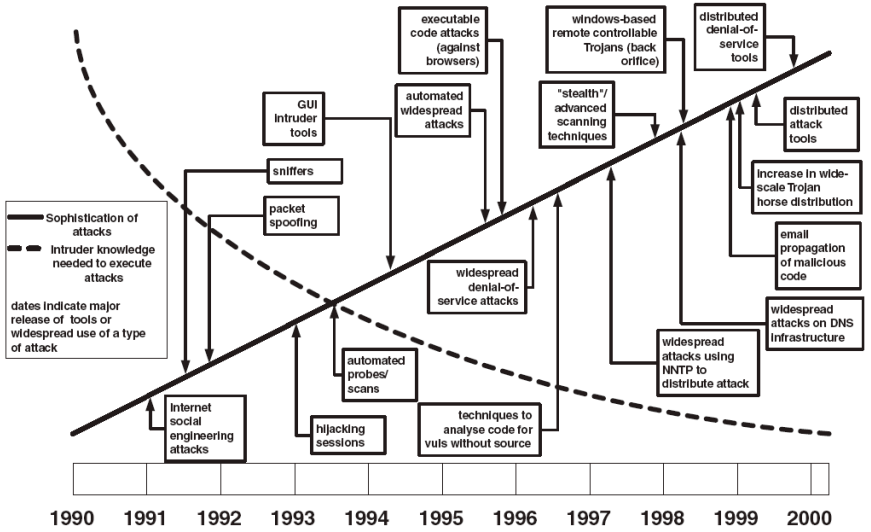
Denial of Service
Timeline (3)

- 2000: (Fear of Y2K bug heightens anxiety over DoS)
  - » DDoS aimed at routers observed
    - ❖ Most major web sites targeted
  - » Demonstrated that overprovisioning of bandwidth not a panacea
- 2001:
  - » First reflected DDoS attack using DNS observed
    - ❖ One attack lasted for 1 week
    - ❖ Demonstrated that simple filtering not a panacea
  - » First Internet measurement studies conducted
    - ❖ Low estimate: 4,000 attacks per week
  - » Microsoft taken offline because of attack on an edge router
  - » Return of the worms
    - ❖ Code Red (v1, v2, II), Nimba, …

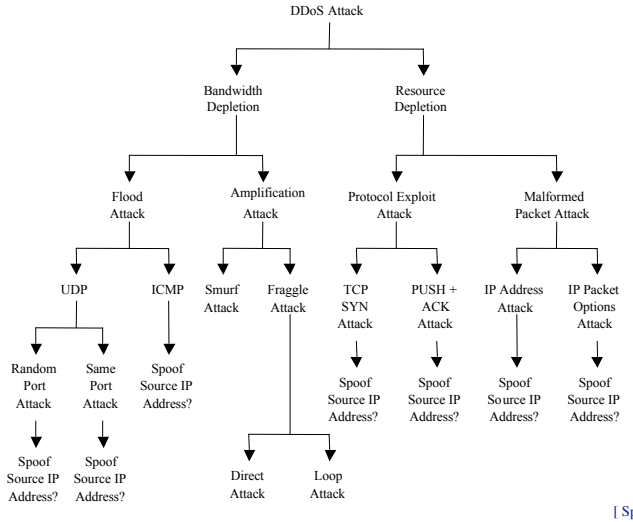
Denial of Service
Timeline (4)

- 2002: DDoS attack targets DNS root (TLD) servers
  - » Attack lasts 1 hour but knocks out 9 of the 13 TLD servers
  - » Most people see little effect
- 2003: Worms meet DDoS (Slammer/Sapphire)
  - » Spam used to disseminate worms that launch DDoS
  - » Financial crimes (extortion) using DDoS reported
  - » Anti-spam, anti-DDoS sites attacked
  - » Al-Jazeera taken offline for two days early in Iraq war
  - » SCO sustains prolonged DDoS attack (blamed on open source commnity)
- 2004: From worm to bot
  - » *Agobot* and *Phatbot* implicated in spam delivery and DDoS
  - » Programmable bot armies controlled via IRC channels widespread
  - » Bot networks offered for sale on the black market

Denial-of-Service
Timeline [McHugh 01]

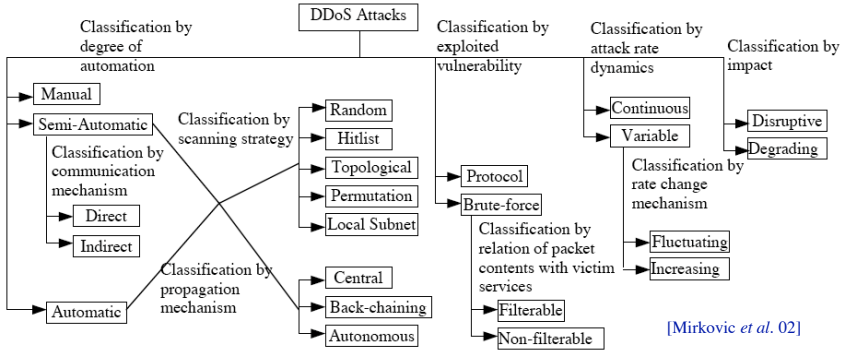


Distributed Denial-of-Service
Taxonomy of attacks (1)



[Specht & Lee 04]

Distributed Denial-of-Service
Taxonomy of attacks (2)



[Mirkovic *et al.* 02]

Denial-of-Service
Defenses!

- No silver bullet!
  - » Remember that this is fundamentally a cat-and-mouse game
  - » (Ignore prevention for now)
- Overprovisioning of bandwidth and server cycles ("resource multiplication")
  - » Works great (if you can afford it) but overprovisioning just moves the primary bottleneck to some other component (if it moves at all)
  - » Dovetails nicely with research into load-adaptive server clusters

Denial-of-Service
Defenses (2)

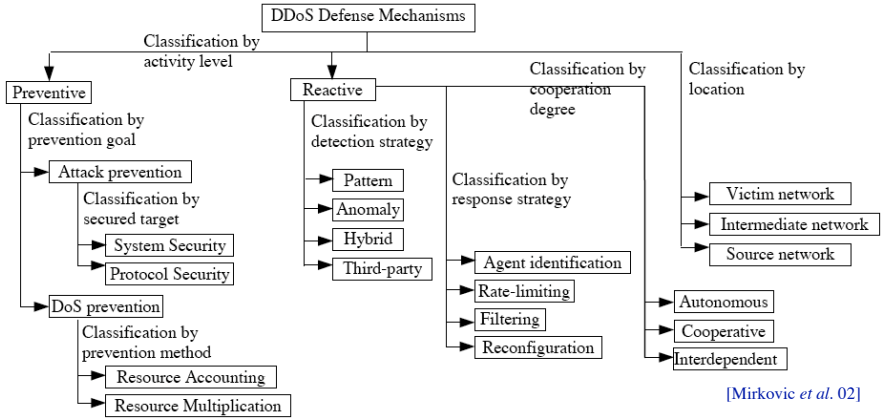
- Re-engineering protocols to be (at least) symmetric in work required at client and server
  - » SYN floods effective because server initially does more work than client
  - » Client-required puzzle solving makes client demonstrate a commitment to communicate
  - » Pricing models?
- Stopping DDoS attacks at the source: Ingress/Egress filtering
  - » Test outgoing or localized incoming traffic for consistency
  - » But subnet spoofing is now more common…
- Also ingress/egress rate limiting ("penalty boxing")
  - » Raise the stakes for a DDoS attack to be effective
  - » But rate limiting can form the basis for an attack!

Denial-of-Service
Defenses (3)

- Detection and reaction
  - » What we'll study
- Cooperative prevention/detection
  - » Need to figure out a way to enable competing businesses to cooperate without losing existing competitive advantages
    - ❖ This may be easier to getting programmers to write better code!
  - » Protocols for inter-domain IDS data exchange being standardized

Distributed Denial-of-Service

Taxonomy of detection schemes



[Mirkovic *et al.* 02]