Worms

CSC 154

Adapted from V. Smatikov, UT Austin,
John Mitchell, Stanford
with slides borrowed from various (noted) sources
,

Morris Worm

- 1988: No malicious payload, but bogged down infected machines by uncontrolled spawning
  - Infected 10% of all Internet hosts at the time
- Multiple propagation vectors
  - Remote execution using rsh and cracked passwords
    - Tried to crack passwords using small dictionary and publicly readable password file; targeted hosts from /etc/hosts.equiv
  - Buffer overflow in fingerd on VAX
    - Standard stack smashing exploit
  - DEBUG command in Sendmail
    - In early Sendmail versions, possible to execute a command on a remote machine by sending an SMTP (mail transfer) message

Dictionary attack

Buffer overflow attack

Remote shell

- Unix trust information
  - /etc/host.equiv – system wide trusted hosts file
  - /.rhosts and ~/.rhosts – users' trusted hosts file
- Worm exploited trust information
  - Examining files that listed trusted machines
  - Assume reciprocal trust
    - If X trusts Y, then maybe Y trusts X
- Password cracking
    - Worm was running as daemon (not root) so needed to break into accounts to use .rhosts feature
    - Dictionary attack
    - Read /etc/passwd, used ~400 common password strings

fingerd

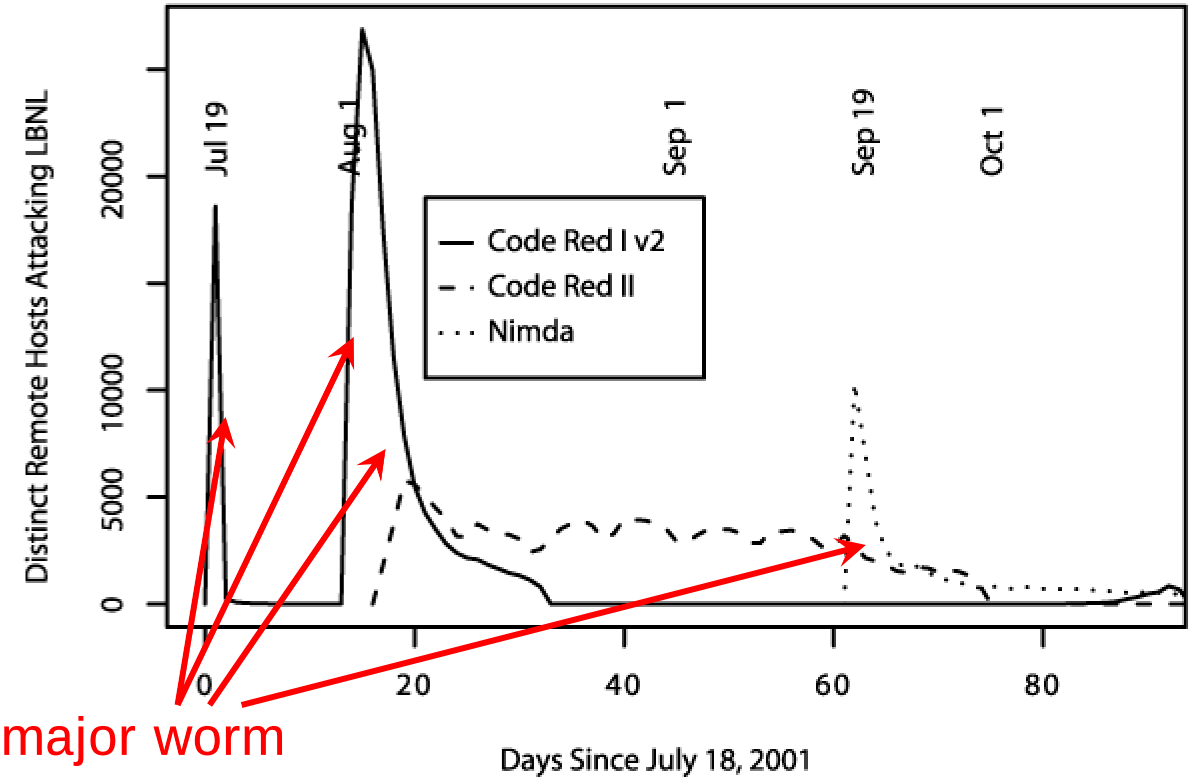
- Written in C and runs continuously
- Array bounds attack
  - Fingerd expects an input string
  - Worm writes long string to internal 512-byte buffer
- Attack string
  - Includes machine instructions
  - Overwrites return address
  - Invokes a remote shell
  - Executes privileged commands

sendmail

- Worm used debug feature
  - Opens TCP connection to machine's SMTP port
  - Invokes debug mode
  - Sends a RCPT TO that pipes data through shell
  - Shell script retrieves worm main program
    - places 40-line C program in temporary file called x$$,l1.c where $$ is current process ID
    - Compiles and executes this program
    - Opens socket to machine that sent script
    - Retrieves worm main program, compiles it and runs

Summer of 2001

[from "How to 0wn the Internet in Your Spare Time"]



Three major worm outbreaks

Code Red I

- July 13, 2001: First worm of the modern era
- Exploited buffer overflow in Microsoft's Internet Information Server (IIS)
- 1st through 20th of each month: spread
  - Find new targets by random scan of IP address space
    - Spawn 99 threads to generate addresses and look for IIS
  - Creator forgot to seed the random number generator, and every copy scanned the same set of addresses ☺
- 21st through the end of each month: attack
  - Deface websites with **`"HELLO! Welcome to http://www.worm.com! Hacked by Chinese!"`**

Usurped Exception Handling In IIS

[See Chien and Szor, “Blended Attacks...”]

- Overflow in a rarely used URL decoding routine
  - A malformed URL is supplied to vulnerable routine...
  - ... another routine notices that stack has been smashed and raises an exception. Exception handler is invoked...
  - ... the pointer to exception handler is located on stack. It has been overwritten to point to a certain instruction inside the routine that noticed the overflow...
  - ... that instruction is CALL EBX. At that moment, EBX is pointing into the overwritten buffer...
  - ... the buffer contains the code that finds the worm’s main body on the heap and executes it!

Code Red I v2

- July 19, 2001: Same codebase as Code Red I, but fixed the bug in random IP address generation
  - Compromised all vulnerable IIS servers on the Internet
  - Large vulnerable population meant fast worm spread
    - Scanned address space grew exponentially
    - 350,000 hosts infected in 14 hours!!
- Payload: distributed packet flooding (denial of service) attack on *www.whitehouse.gov*
  - Coding bug causes it to die on the 20th of each month... but if victim's clock is wrong, resurrects on the 1st!
- Still alive in the wild!

Code Red II

- August 4, 2001: Same IIS vulnerability, completely different code, kills Code Red I
  - Known as "Code Red II" because of comment in code
  - Worked only on Windows 2000, crashed NT
- Scanning algorithm preferred nearby addresses
  - Chose addresses from same class A with probability ½, same class B with probability 3/8, and randomly from the entire Internet with probability 1/8
- Payload: installed root backdoor in IIS servers for unrestricted remote access
- Died by design on October 1, 2001

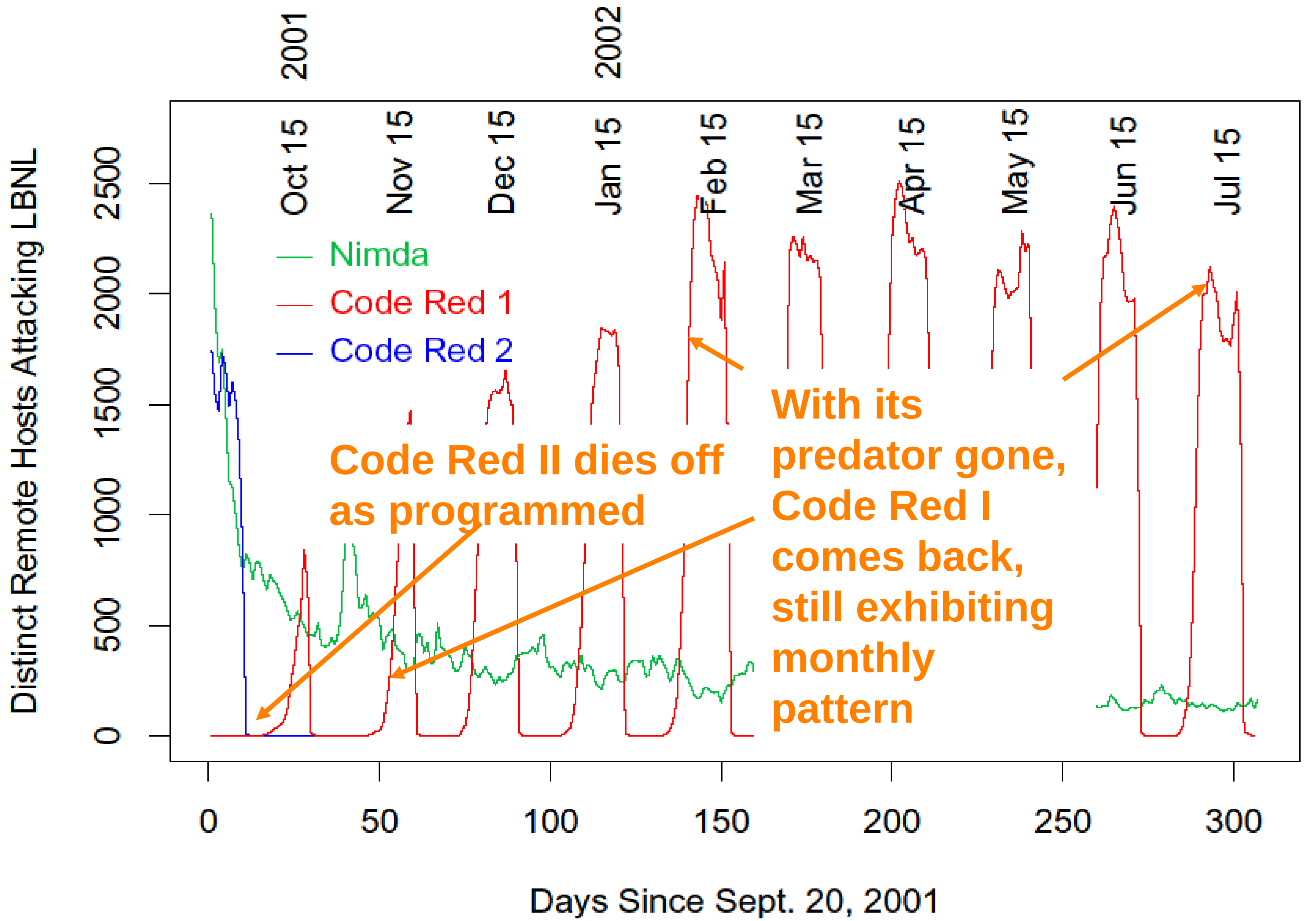
Nimda

- September 18, 2001: Multi-modal worm using several propagation vectors
  - Exploit same IIS buffer overflow as Code Red I and II
  - Bulk-email itself as an attachment to email addresses harvested from infected machines
  - Copy itself across open network shares
  - Add exploit code to Web pages on compromised sites to infect visiting browsers
  - Scan for backdoors left by Code Red II
- Payload: code deleting all data on hard drives of infected machines

Signature-Based Defenses Don’t Help

- Nimda leaped firewalls!
- Many firewalls pass mail untouched, relying on mail servers to filter out infections
  - Most filters simply scan attachments for signatures (code snippets) of known viruses and worms
- Nimda was a brand-new infection with unknown signature, and scanners could not detect it
- Big challenge: detection of zero-day attacks
  - When a worm first appears in the wild, signature is not extracted until minutes or hours later

Code Red I and II (due to Vern Paxson)



Slammer (Sapphire) Worm

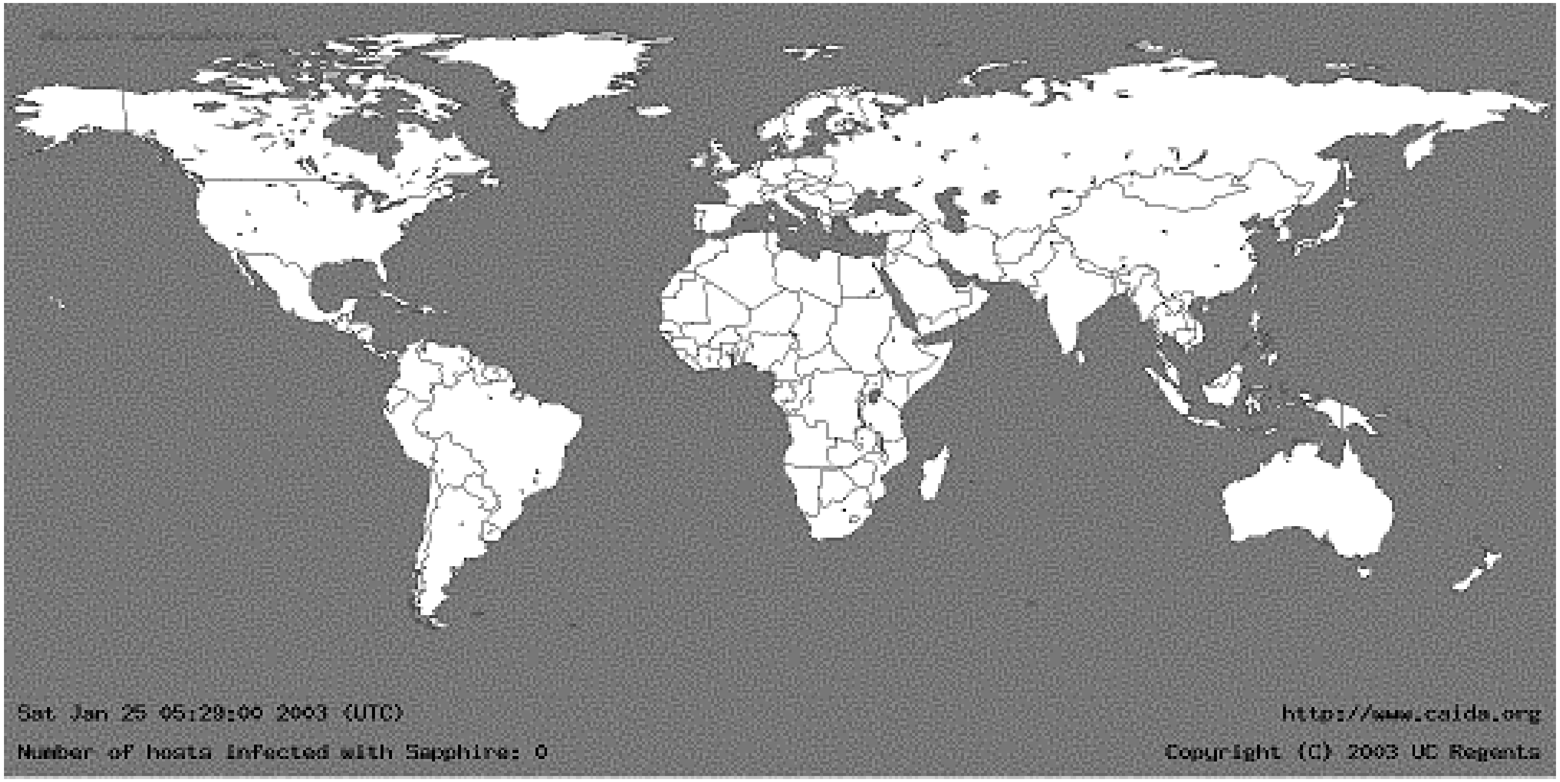
- January 24/25, 2003: UDP worm exploiting buffer overflow in Microsoft's SQL Server
  - Overflow was already known and patched by Microsoft... but not everybody installed the patch
- Entire code fits into a single 404-byte UDP packet
  - Worm binary followed by overflow pointer back to itself
- Classic buffer overflow combined with random scanning: once control is passed to worm code, it randomly generates IP addresses and attempts to send a copy of itself to port 1434
  - MS-SQL listens at port 1434

Slammer Propagation

- Scan rate of 55,000,000 addresses per second
  - Scan rate = rate at which worm generates IP addresses of potential targets
  - Up to 30,000 single-packet worm copies per second
- Initial infection was doubling in 8.5 seconds (!!)
  - Doubling time of Code Red was 37 minutes
- Worm-generated packets <u>saturated carrying capacity</u> of the Internet in 10 minutes
  - 75,000 SQL servers compromised

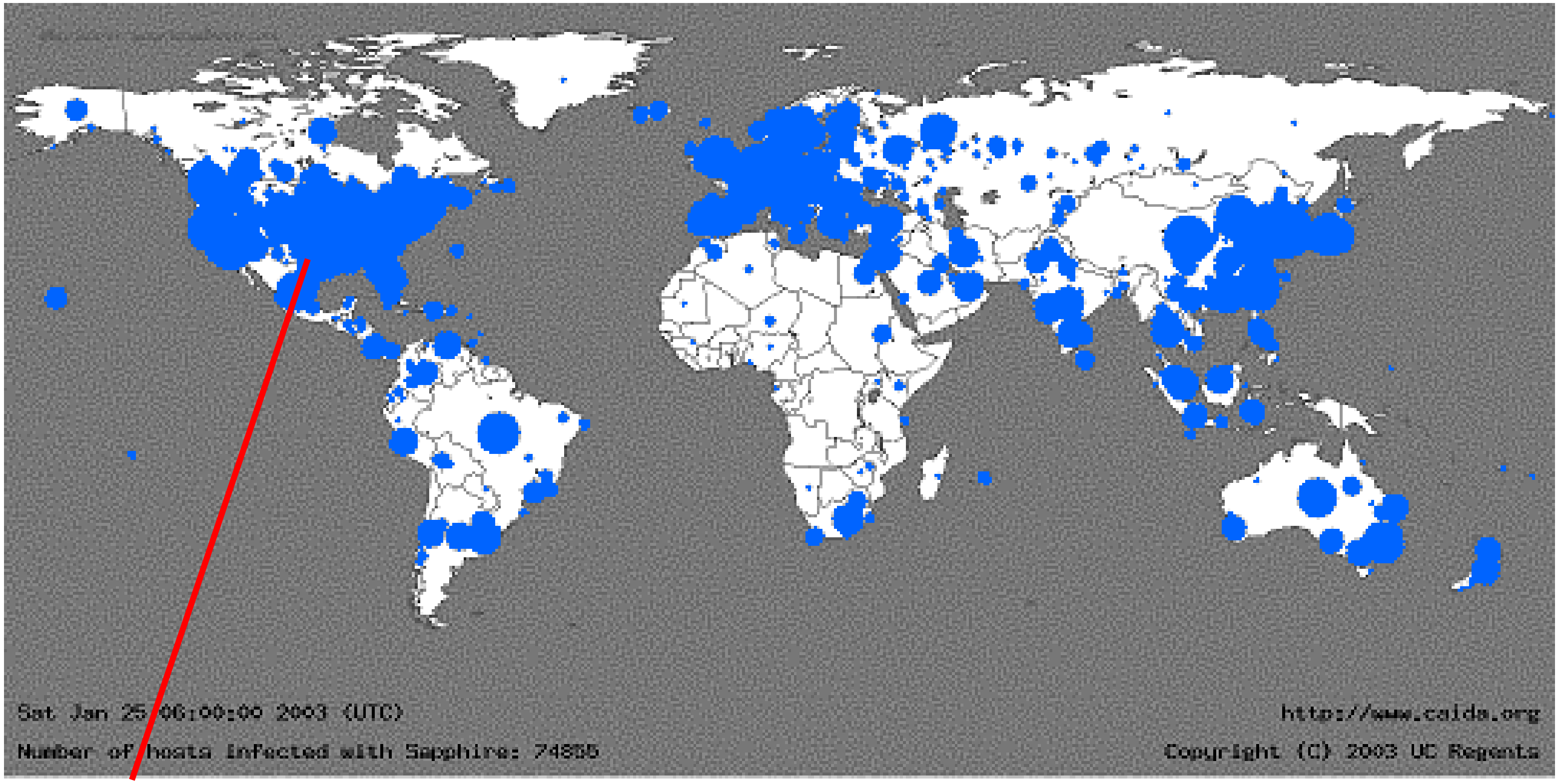
05:29:00 UTC, January 25, 2003

[from Moore et al. "The Spread of the Sapphire/Slammer Worm"]



30 Minutes Later

[from Moore et al. "The Spread of the Sapphire/Slammer Worm"]



Size of circles is **logarithmic** in the number of infected machines

Slammer Impact

- $1.25 Billion of damage
- Temporarily knocked out many elements of critical infrastructure
  - Bank of America ATM network
  - Entire cell phone network in South Korea
  - Five root DNS servers
  - Continental Airlines' ticket processing software
- The worm did not even have malicious payload... simply bandwidth exhaustion on the network and resource exhaustion on infected machines

Secret of Slammer's Speed

- Old-style worms (Code Red) spawn a new thread which tries to establish a TCP connection and, if successful, send a copy of itself over TCP
  - Limited by latency of the network
- Slammer was a connectionless UDP worm
  - No connection establishment, simply send 404-byte UDP packet to randomly generated IP addresses
  - Limited only by bandwidth of the network
- A TCP worm can scan even faster
  - Dump zillions of 40-byte TCP-SYN packets into link layer, send worm copy only if SYN-ACK comes back

Blaster and Welchia/Nachia

- August 11, 2003: Scanning worm exploiting RPC service in Microsoft Windows XP and 2000
  - First address at random, then sequential upward scan
    - Easy to detect, yet propagated widely and leaped firewalls
- Payload: denial of service against MS Windows Update + installing remotely accessible backdoor
- Welchia/Nachia was intended as a counter-worm
  - Random-start sequential scan, use ICMP to determine if address is live, then copy itself over, patch RPC vulnerability, remove Blaster if found
  - Did **more** damage by flooding networks with traffic

How to Build a Super-Worm?

[from "How to 0wn the Internet in Your Spare Time"]

- Objective: Warhol worm
  - Worm that reaches saturation (infection of **all** potentially vulnerable targets) in 15 minutes
  - Faster than any possible human-mediated respose
- Previous worms suffered from suboptimal design
  - Slammer copies ended competing with themselves for bandwidth
  - Broken address generation algorithms
  - Buggy payloads (premature death, failed DDoS, etc.)

Better Target Address Generation

- Pre-compute hit-list of vulnerable hosts
  - Very slow, stealthy scan for known vulnerabilities over several months prior to worm release
    - To cover your tracks, do it from hacked “zombie” machines
  - Web-crawling spiders
  - Listen for responses to other attacks
    - E.g., every IIS infected with Code Red announced its presence by dumping large amounts of traffic to random addresses
- Even imperfect hit-list will greatly speed up initial infection (slowest part of worm propagation)
  - Start with a single host; every time the worm divides, it “outsources” half of its hit-list to the new copy

Coordinated Scanning

- Random address generation is inefficient
  - Many addresses are probed multiple times, worm copies flood the bandwidth
- Permutation scan: each copy starts to scan from a random point in IP address space; if encounters another copy, randomly picks another point
  - Worm needs to recognize its own presence on the target machine
- Divide-and-conquer: split target address space in half each time a new copy is created
  - Probably can infect 1,000,000 hosts in 2 seconds

Exploit Existing Networks

- Use network topology
  - Morris worm looked for new targets in hosts.equiv
  - Peer-to-peer networks are perfect targets
    - Instead of generating random addresses, just spread to peers
- Get initial hit-list from meta-servers
  - Use online directories to find potential victims
  - Paxson's example: Google for "powered by phpbb" to find websites running PHP
- Piggyback on existing network traffic
  - E.g., worm inserts itself into Kazaa or BitTorrent traffic
  - Virtually undetectable (no unusual network activity!)