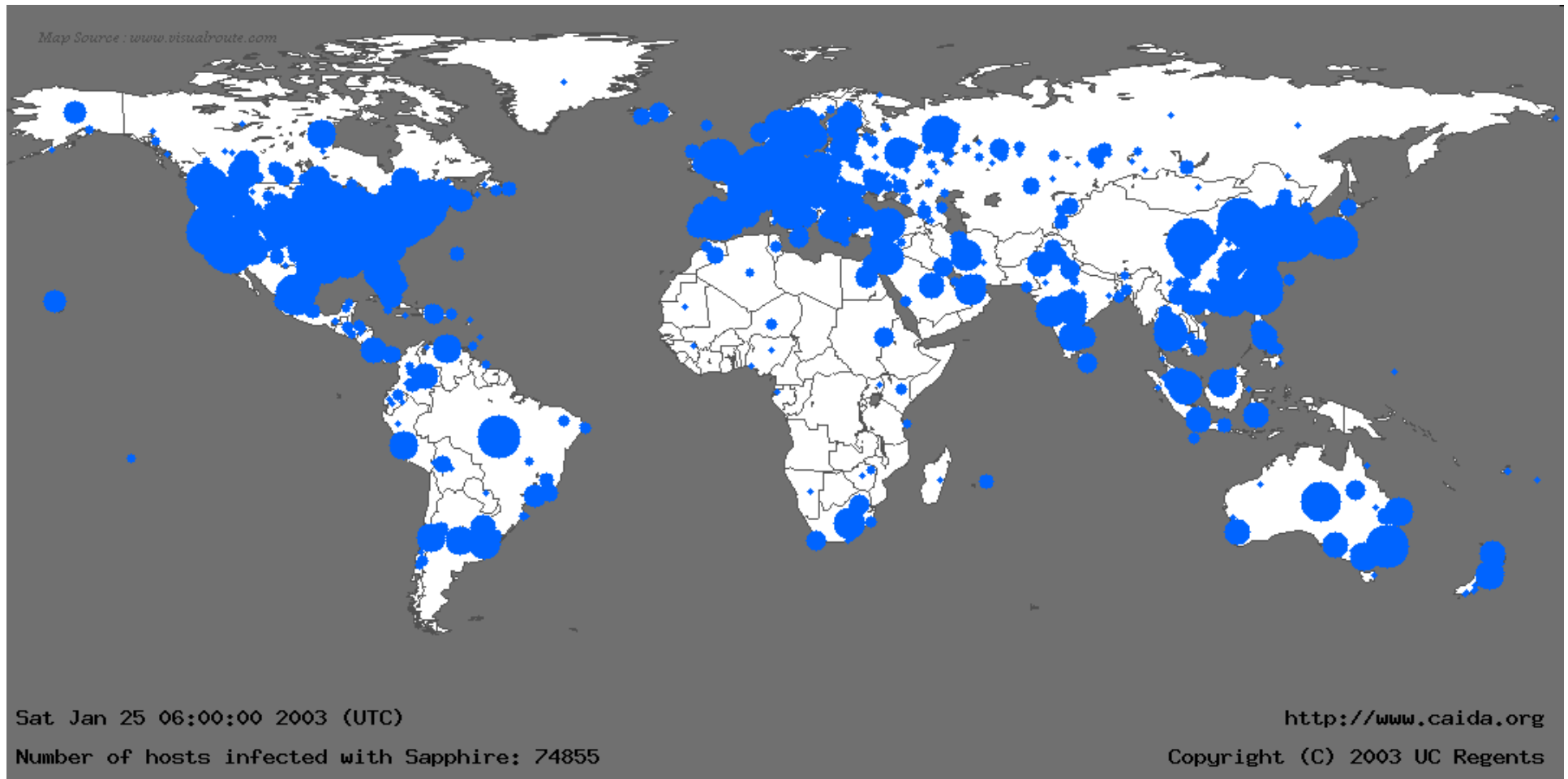


Internet Worms

Internet Worms

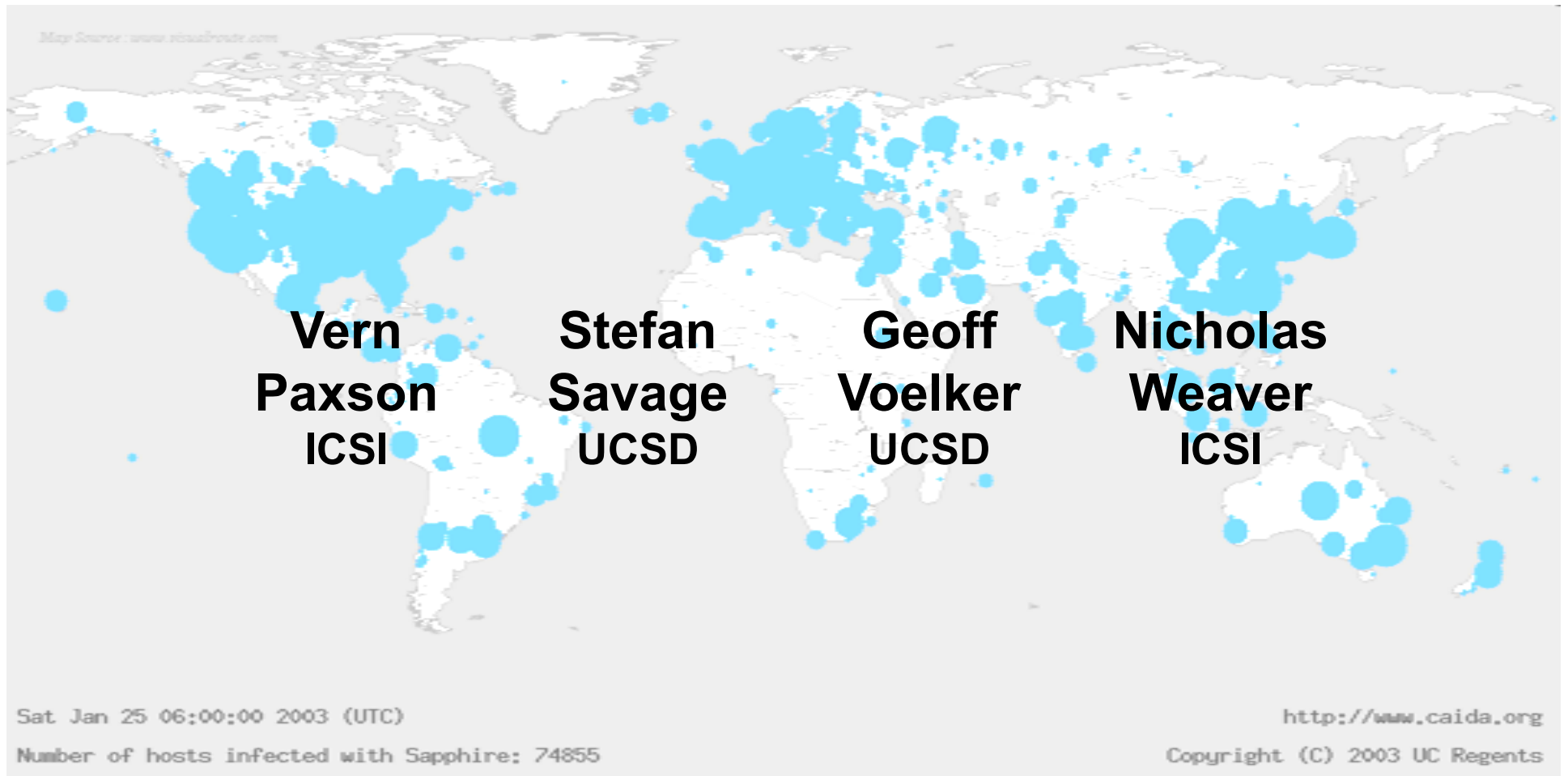
Paxson, Savage, Voelker, Weaver



Internet Worms

Internet Worms

Paxson, Savage, Voelker, Weaver



Introductions



Vern Paxson
ICSI



Stefan Savage
UCSD



Nicholas Weaver
ICSI



Geoff Voelker
UCSD

Together we run the Cooperative Center for
Internet Epidemiology and Defenses

Today

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

- Afternoon session (defenses)
 - Overview
 - Detection
 - Signature-based
 - Behavioral
 - Mitigation



The 10,000 foot questions

- Why is this a big issue **now**?
 - Didn't we have worms and viruses back in the 80s?
- Isn't ~~spyware~~
~~phishing~~
~~botnets~~ the **real** problem now?
something new
- Isn't all this Internet threat stuff overhyped?
 - My computer continues to work after all...

The sky is falling! The sky is falling!

Internet Worms
Paxson, Savage, Voelker, Weaver

The New York Times **Technology**

NYTimes: [Home](#) - [Site Index](#) - [Archive](#) - [Help](#)

Go to a Section Quotes: Site Search:



 Access to more rooms in London than any airline.

[NYTimes.com](#) > [Technology](#)

Attacks on Windows PC's Grew in First Half of 2004

By **JOHN MARKOFF**
Published: September 20, 2004

SAN FRANCISCO, Sept. 19 - A survey of Internet vulnerabilities

MSNBC News Print | Email | Alerts | Newslett

TECHNOLOGY & SCIENCE

Spam, Scams & Viruses

Info - Message (Plain Text) - Western European (Windows) [P] [R] [Q]

From: [redacted]
To: [redacted]
Subject: Hello

[The message contains Unicode characters and has been sent as a binary attachment.]

Mydoom threat still high; Microsoft offers reward

Software giant puts up \$250,000 for information leading to arrest

Trend Micro Inc An infected e-mail includes a simple invitation to click on an a

CNN.com International Edition | Netscape

MEMBER SERVICES MAKE CNN.com YOUR HOME PAGE

SEARCH Powered by **YAHOO!** search

Items Page
World
U.S.
Weather
Business
Sports & More
Politics
Law
Technology
Science & Space
Health
Entertainment
Travel
Education
Special Reports

TECHNOLOGY

'Phishing' scams reel in your identity

Feds pursue culprits, warn consumers

WASHINGTON, Sept. 19 (CNN) — The Web sites look real and the information sought seems justified. But it's really the latest form of e-mail scam, called "brand spoofing," "carding" or "phishing."

The often a luring message that asks for credit card numbers, bank account numbers, billing information and other sensitive information. The scam is usually conducted through e-mail messages and messages on e-commerce Web sites, notes a report



Washington Post's Chief Privacy Officer, Suzanne and the company's tough on "phishing" scams.

BusinessWeek

SEPTEMBER 14, 2004

FORD
A LOT IS RIDING ON THE NEW F-150 PICKUP

HEINEKEN
WAKING UP AN OLD WORLD BREWER

BROADBAND
HOW THE U.S. CAN CATCH UP

RESEARCH
A RECCA FOR BIO-MEDICINE

DRESSING SMART
THE NEW LOOK IN MEN'S FASHION

MSNBC News Print

TECHNOLOGY & SCIENCE

Spam, Scams & Viruses

EPIDEMIC

Crippling computer viruses threaten the info economy. Can they be stopped?



Experts fret over online extortion

armies capable of toppling big sites, some

Sullivan
logy correspondent

d: 8:34 p.m. ET Nov. 10, 2004

e 21st century's equivalent of a ransom note: Pay up or a massive denial of service attack on your Web site ed by thousands of hijacked "zombie" computers.

washingtonpost.com

A year of spam, spyware and worms

Dark side of Internet boiled over in 2003

By **Rob Pegoraro**
[washingtonpost.com](#)
Updated: 8:32 p.m. ET Dec. 30, 2003

MSNBC News Print

TECHNOLOGY & SCIENCE

Spam, Scams & Viruses

Experts fret over online extortion

armies capable of toppling big sites, some

Sullivan
logy correspondent

d: 8:34 p.m. ET Nov. 10, 2004

e 21st century's equivalent of a ransom note: Pay up or a massive denial of service attack on your Web site ed by thousands of hijacked "zombie" computers.

Chicken Little is a naïve optimist

- Imagine the following species:
 - Poor genetic diversity; heavily inbred
 - Lives in “hot zone”; thriving ecosystem of infectious pathogens
 - Instantaneous transmission of disease
 - Immune response 10-1M times slower
 - Poor hygiene practices
- **What would its long-term prognosis be?**
- What if diseases were designed...
 - Trivial to create a *new* disease
 - Highly profitable to do so

Fundamental threat transformation

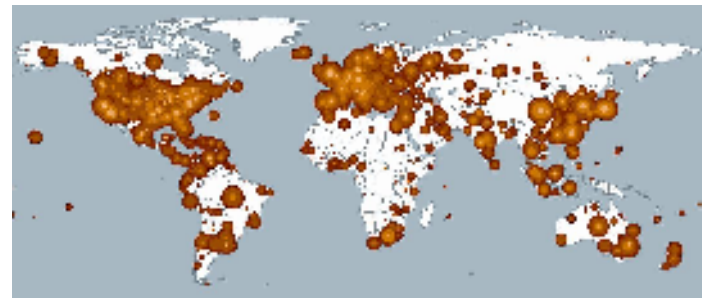
- **Traditional threats**

- Attacker manually targets high-value system/resource
- Defender increases cost to compromise high-value systems
- Biggest threat: insider attacker



- **Modern threats**

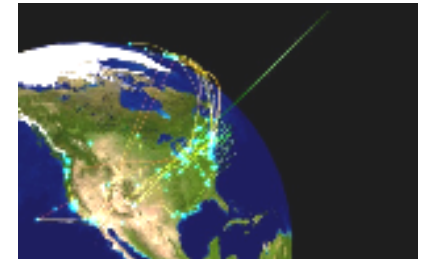
- Attacker uses automation to target **all** systems at once (can filter later)
- Defender must defend **all** systems at once
- Biggest threats: software vulnerabilities & naïve users



Technical Enablers

- **Wide-open communications architecture**

- IP model: anyone can send anything to anyone
- Federated management, minimal authentication



- **Vulnerable computing platforms**

- One software bug -> millions of compromised hosts
- Naïve users -> don't even need software bugs

- **Lack of meaningful deterrence**

- Little forensic attribution/audit capability
- Inefficient investigatory mechanisms/prosecutorial incentives



"On the Internet, nobody knows you're a dog."

Economic Drivers

- In last five years, emergence of profit-making malware
 - SPAM forwarding (MyDoom.A backdoor, SoBig), Credit Card theft (Korgo), DDoS extortion, (many) etc...
 - “**Virtuous**” economic cycle **transforms** nature of threat
- Commoditization of compromised hosts
 - *Fluid* third-party exchange market (**millions of hosts**)
 - Raw bots, ~\$.05/host, Special orders (up to \$50/bot for .mil)
 - Value added tier: SPAM proxying 3 -10 cents/host/week
 - **Compromised hosts** have become a criminal **platform**
- Innovation in both host substrate and its uses
 - Sophisticated infection and command/control networks
 - DDoS, SPAM, piracy, phishing, identity theft are all **applications**

Botnet Spammer Rental Rates

>20-30k always online SOCKs4, url is de-duped and updated every
>10 minutes. 900/weekly, Samples will be sent on request.
>Monthly payments arranged at discount prices.

- 3.6 cents per bot week

>\$350.00/weekly - \$1,000/monthly (USD)
>Type of service: Exclusive (One slot only)
>Always Online: 5,000 - 6,000
>Updated every: 10 minutes


- 6 cents per bot week

>\$220.00/weekly - \$800.00/monthly (USD)
>Type of service: Shared (4 slots)
>Always Online: 9,000 - 10,000
>Updated every: 5 minutes

- 2.5 cents per bot week

September 2004 postings to SpecialHam.com, Spamforum.biz

What service-oriented computing really means...

▼ Subject: I offer the DDoS attack service !
From: ddos@safe-mail.net <DDOS Service> 
Date: 3/3/05 10:54
Newsgroups: alt.2600.cardz

HI,

I offer the DDOS attack service, I offer estimate of expense on hour base. Free demonstration (10 minutes).

The price is based on the difficulty to pull down the target website, for the free demonstration or information please contact :

DDOS Service at : ddos@safe-mail.net

Next:

large-scale *Information* Exploitation

- The true value in a host is the *information* it holds
 - Spreadsheets, e-mail, presentations, source code, etc
- It is *easy* to implement distributed queries:
 - Find all spreadsheets containing “10-Q” & “2007”
 - Find all e-mail containing “From:” and “microsoft.com”
 - Find any computer containing an interface on **this** network
- Cast a wide net and reel in...
 - *This is already happening for commodity info*
(PayPal/Credit Card info, CD-Keys, passwords/PII)

What does this have to do with worms?

- All of it depends on automated mechanisms for subverting large numbers of hosts
- Self-propagating programs continue to be the most effective mechanism for doing this
- The difference between a worm and a scripted bot is vanishingly small

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



Overview and Taxonomy

- Viruses
- Mail Viruses/Mail worms
- Worms
 - Target-discovery based taxonomy
- Payloads
 - Bots
 - Control Network Taxonomy
 - Rendezvous Points
- Exploit Overview
- Conceptual overview of defenses
 - Aggregation

Mobile Malcode Overview

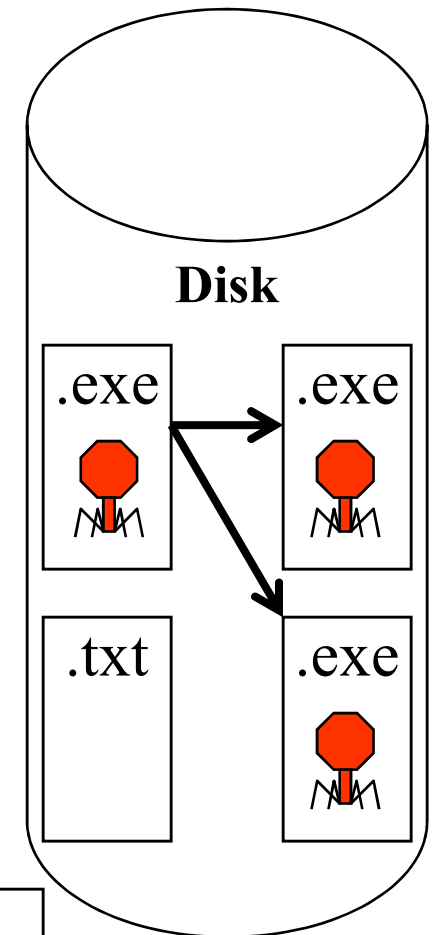
- Malicious programs which spread from machine to machine without the consent of the owners/operators/users
 - However, consent can be obtained with false pretenses, deception, erroneous expectations...
- Many strains of Mobile Malcode possible
 - Viruses
 - Infect Files, spread across file systems or users within a files system with human activation
 - Email-borne Viruses/Worms
 - Propagate through email, usually requires human activation
 - (Autonomous) Worms
 - Processes that can cause a like process to run on a remote host without user activation, very dangerous
 - Compromised Auto-updates
 - No user action required, very dangerous
 - Mix and Match
 - Bots
 - Control networks (often associated with other malcode)
- This tutorial primarily focuses on worms and worm defenses

Mobile Malcode Anatomy

- **Target Discovery/Reconnaissance**
 - How to discover new targets
- **Attack Vector**
 - Mechanism to attack the target
- **Infection Vector**
 - Mechanism to transfer over a copy of the worm
- **Activation**
 - Getting the remote copy of the worm to start running
- **Payload**
 - Whatever the attacker desires to accomplish his goals

Virus Overview

- Viruses are mobile, self-replicating subroutines which infect executable (or interpretable) files
 - First documented "virus" was written by Ken Thompson [T84]
 - Term coined by Len Adelman to describe Fred Cohen's graduate work [C84]
- On execution:
 - Search for valid target files (target discovery)
 - Usually executable files
 - Often only infect uninfected files
 - Inject a copy into targeted files (attack & infection)
 - When the target is executed, the virus starts running (activation)
- Only spread when contaminated files are moved from machine to machine
 - Some infected "auto-start" sections of floppy disks
- Mature defenses available



Virus Infectables

- Executable files
 - COM, EXE, BAT
 - .PDF, .PS, other markup languages have executable content
- Macros
 - Many application environments
- Source code viruses
 - Ken Thompson's
- System sector viruses
 - Infect control sectors on a disk
 - DOS boot sectors
 - Partition (MBR) sectors
- Companion viruses
 - Creates a .com file for each infected .exe file
 - Relies on OS (i.e., DOS) executing .com before .exe
- Cluster viruses
 - Capture directory traversals, infect all subordinates

A Simple Virus Example

- Capture flow control
- Insert code
- On some condition, find other files to infect
- On some condition, execute payload
- Infection changes program behavior, content, and (sometimes) size

main:

goto Virus

goto Foo

goto Bar

Foo:

...

Bar:

...

Virus:

if(cond 1) find & infect files

if(cond 2) execute payload

Variable Viruses

- Polymorphic
 - Change disk footprint at the byte level
 - Encrypt with nonconstant key and random sets of cryptographic primitives
 - Executable virus code changing (macros: var names, line spacing, null functions)
 - Control flow permutations (rearrange code with goto's)
 - Also called hardened
- Metamorphic [SF01]
 - “Metamorphics are body-polymorphics”, Igor Muttik
 - Change footprint on disk and in memory, at the semantic level, but with same consequences
 - Preserve consequences
 - No common byte streams between generations
 - No encryption
 - Integrates with victim code
- Polymorphic and Metamorphic code can also be incorporated into other forms of malware
 - Polymorphic worms and bots are possible

Virus Detection/Evasion

- Detection (examples)
 - Look for changes in file
 - size, time stamp, hash/checksum, type
 - Look for patterns (byte streams) in virus code that are unique
 - Look for bad behavior
 - Modification of executable files
 - changing program behavior
 - Detection schemes also useful for discovering worms/bots on end-host systems
 - Note that detecting viruses is undecidable
 - Proved by Fred Cohen in his dissertation
- Evasion (examples)
 - Compression of virus and target code
 - Modify time stamp to original
 - Change patterns
 - Polymorphism
 - Metamorphism
 - Don't touch disk
 - Disable AV programs
 - Don't do much that's *bad*

Virus Recovery

- Extricate the virus from the infected file to leave the original (uninfected file) behind
- Remove the redirection to the virus code
 - Virus exists but not triggered
 - Can cause false-positives during subsequent scans
- Recover the file from backup
- Delete the file and move on with life

Virus Prevention

- DON'T EXECUTE THE VIRUS!
- On shared file systems there may be many shared executables
 - Don't execute another unprivileged user's files
 - If root gets infected, all bets are off
 - root is difficult to protect
- Scan all files from (even trusted) sources that you put/pull inside your perimeter
 - Email, disks, web, cvs, etc. -- scan all files
 - Prohibit classes of files from crossing the perimeter

Macro Viruses

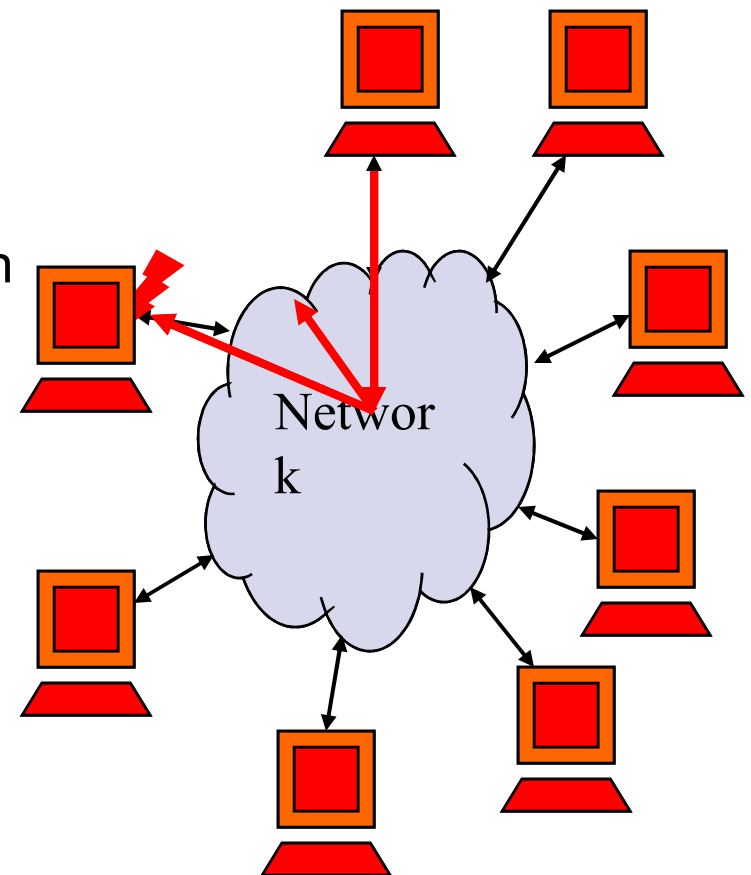
- Word/office macros are a full programming language (many passive documents have macros)
 - Can be carried in arbitrary word documents
 - Word documents contain “executable” content
 - Can move from document to document
 - General danger in mingling code and mobile data
 - Not unique to Microsoft, but Office was a big offender at mingling code and data
- Was a big problem until the Melissa mail worm:
 - A word macro which, rather than infecting files, emailed out copies of itself
 - Word macros now have much more restricted semantics when moving from file to file

Email-Borne Viruses/Worms

- The logic/code is contained in an email, usually as an executable attachment
 - Some mail agents execute content/attachments automatically
 - Upon execution:
 - Search for email addresses in address book, emails (target discovery)
 - Send out copies of itself to all the addresses it can find (attack)
- Activation:
 - Hope that the sucker runs you
 - Exploit vulnerability in the mail reader
- Taxonomy: Are they viruses or worms?
 - "Email Viruses" when humans or human-agents asynchronously trigger the worm
 - "Email Worms" when they are fully autonomous
 - In general, there is no good overall taxonomy for malware

Worms

- Autonomous, active code that can replicate to remote hosts without any triggering
 - Typically, a worm is a process not a file
- Because they propagate autonomously, they can spread much more quickly than viruses
- This tutorial primarily focuses on worms
 - Speed and general lack of user interaction make them a harder research problem
 - Reactive defenses against worms must be automatic



Target Discovery Taxonomy [WPSC03]

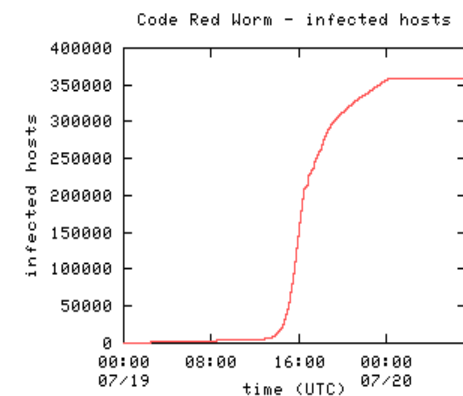
- For active worms, the worm **must** find new targets to infect
 - Use this to define classes of worm behavior
- Scanning
 - Pick “random” addresses and try to attack (lots)
 - Many variations on “random”, but all try addresses which are not **a priori** suspected to be vulnerable
- Target lists
 - **Hit lists**: attacker pregenerates a list of victims (Witty)
 - How is up to the attacker
 - **Meta-server**: worm queries a server which has information
 - E.g., queries Google to find vulnerable PHP-containing servers (Santy worm)
 - **Topological**: Uses local information (Morris worm, mail worms)
 - E.g., hosts file, email addresses
- All these techniques have been used by worm authors

Target Discovery: Scanning

- Pick random addresses, attempt to infect
 - Highly anomalous:
 - Detect scanning
 - Possible to perform worm containment
 - Take advantage of scanning's randomness
 - Lots of optimizations possible
 - Local subnet scanning
 - Bandwidth-limited scanner
 - Permutation scanning
 - Fully coordinated scanning
 - Spread follows the logistic equation
 - Exponential start, then exponential die-off
 - Speed is a function of the vulnerable population and the speed of the scanner
 - Generic strategy

$$a = \frac{e^{K(t-T)}}{1 + e^{K(t-T)}}$$

$$K = \frac{\text{Scan Rate} * \text{Vuln Machines}}{\text{Address Space Size}}$$



Target Discovery: Target Lists

- Worm obtains a list of probable targets
 - External: **Meta-server** worm
 - Queries a third party server (eg Google, Active Directory Server, Game Matchmaker server)
 - Pregenerated: **Hit list** and **Flash** Worm
 - A generic strategy, attacker scans/acquires a list in advance
 - Internal: **Topological** worm
 - Look in local disk/memory for new targets
 - Basis of the Morris worm
- These worms are far less anomalous
 - But may not be purely “normal”
- Naturally produces fast worms

Contagion/Passive Worms

- Worm waits for behavior
 - Web requests
 - Connection requests
 - File sharing
 - Attacks by other worms
- Responds with an infection attempt
 - Possibly as part of the normal response
- Possibly highly stealthy
 - Target discovery is quiet on the network
 - Infection attempt may be disguised as normal communication
- Speed is highly target-dependent
 - Driven by the protocol and usage

Target Discovery: Passive Worms

- Worm waits for event
 - Passive Target Discovery **cannot** be detected in the network.
- Nimda's firewall-crossing mode
 - When vulnerable web browser visits, respond with contagion attack
 - Web browser attacks have been exploited heavily by single-stage trojans (non-self-propagating) malware
- CRClean Code Red 2 "anti"-worm
 - Wait for Code Red 2 attack
 - Respond by exploiting the Code Red 2 backdoor
 - Install copy on remote machine
 - Disable Code Red
 - Disable the Code Red vulnerability
 - Remove Code Red 2 Backdoor
 - Written but not released

Attack Vector

- The worm needs to take control of the remote computer
 - Works like any other security attack
- Many, **but not all**, rely on C/C++'s weak memory and type semantics
 - Buffer overflows
 - Integer overflows
 - Memory allocation/deallocation errors
- Others are often bad policy
 - Access control, authorization
- Perhaps the most difficult to counter
 - Social engineering & human idiocy
 - Transitive trust & escalation (discussed later)

Infection Vector: Secondary Communication Channel

- Once the worm has taken control of the target, it needs to copy itself over
 - Just controlling the remote machine is insufficient, the worm itself needs to be transferred
- Simplest mechanism: Open a second channel
 - Using TFTP or a similar protocol, transfer over the worm body from the infecting machine
 - Get the main body from a webpage or secondary source
 - Once the transfer is complete, then run the worm
- May be blocked
 - Worm successfully attacks the target, but can't complete the infection process due to network restrictions
 - TFTP not allowed through the firewall
 - TFTP/TFTPD unavailable
 - Blaster had to rely on the system having a TFTP that it could run to infect other systems
 - Secondary source shut down

Infection Vector: Embedded Worm

- After worm attacks the target, transfer the body over in the same communication channel
 - No need to open a second channel, no additional programs required
 - If the attack was not blocked, the infection probably will succeed
- Commonly used with memory-resident worms such as **Code Red, Slammer**
 - Injected code uses a (usually large) buffer
 - Worm body is then written into the buffer, injected code jumps to the start of the worm
 - Works best with position-independent code
 - Code gets a pointer to the current PC, uses this to calculate offsets for constants and related values
 - All jumps in the code are PC relative (no relinking required)
 - Either write it thyself in assembly or massage the output of gcc

Activation

- Just transferring the worm body is insufficient, it needs to start running on the victim
 - Start as part of infection
 - Worm will spread fully autonomously
 - Start on event or user action
 - May slow down the worm's spread
 - A copy of the worm infects a lot of machines, but they don't contribute to the worm's spread until later in time
- Activation largely depends on exploit/attack vector
 - Automatic: Write code in memory, start it running (**Code Red**)
 - Automatic: Write code onto disk, start it running (**Blaster**)
 - Human/Timer: Write to an open file share, when machine reset or some other event happens ...
 - "Open Shares" Worms rely on human/event-based activation

Hybrid Malcode

- Mobile Malcode can mix propagation vectors
- Open Shares Worm/Virus
 - Look for remote machines (Worm)
 - When discovered, open the c:/ drive, infect executables (virus-like)
 - Plant copy of self in the startup directory (more worm-like)
 - When those executables are executed, the worm starts to spread
 - Human (or boot) activation but autonomous spread
- **Nimda**: an autonomous worm with firewall crossing
 - Used a mail mode and a passive web-browser exploit to achieve an initial foothold
 - Used Open Shares as an additional mechanism for spreading in internal networks
- Contagion worm: hijacks normal behavior to spread
 - Is it a virus infecting the network traffic? A worm?

Compromised Auto-updates

- Auto-update program: At a given time, call to central server
 - If new programs are available, download and install
 - Probably a good thing, when correctly implemented
- Hijack the auto-update
 - May be trivial:
 - DNS cache poisoning or router redirection
 - May be very difficult:
 - Capture the code-signing private-key and the server's private key
- Present a fake update to the world
 - Clients will now automatically download, install, and run the new code
- Unlike other mobile malware, it only travels a single step
 - From the update server to the clients
 - Lots of opportunities, little (known) exploitation
 - Auto-updaters are very common in spyware/scumware programs

Payload

- Whatever the attacker desires and can construct
 - So limited by skill and imagination
 - Payload can do anything with the privileges of the worm
- Do Nothing
 - Either makes a statement (see what I could have done) or a bug
- Destroy/corrupt data
 - Massive denial-of-service attack
- Search for data
 - Generic searches
 - “How do you find a needle in a haystack?”
 - “With a match”
- Internet-scale DDoS attacks
 - Disrupt major information sources, update sources
- Cryptovirology or extortion
 - “Pay us or your data dies”
- Data manipulation
 - Blackmail with porn
- Patch the system
 - Why let others get in the same way?
- **Rootkits**: Stealth Software
 - Hide the malcode from other programs
- **Botnets**: Command and Control
 - Create a distributed system for issuing further commands
 - Use authentication to prevent hijacking
 - Provide updates to the worm
 - Leverage for further attacks

Rootkits: Stealth Software

- A **rootkit** is a piece of software designed to hide functionality from all other system elements
 - Usually by patching the functions used to verify system integrity
 - File system calls, registry examination, etc
 - Several possible places to hook into the system
 - Can even run the system in a VM [KCW06]
 - From within a system, you **can't** reliably determine system integrity
- Only certain way: Examine the system from outside
 - Best method: Compare the view within the system from an exterior viewpoint [YVR04]
 - Any code which is persistent and stealthy **must appear** or detect that the check is being performed (AV heuristic halting problem)
- Heuristic that usually works: Compare from two or more different internal viewpoints, with different code paths (but being countered now [S06])
 - EG, access the windows registry normally, and also examine the file in detail manually [CR05] [F06]

[KCW06]

Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, Jacob R. Lorch, **SubVirt: Implementing malware with virtual machines**, IEEE Security and Privacy conference, 2006

[YVR04] Yi-Min Wang, Binh Vo, Roussi Roussev, Chad Verbowski, and Aaron Johnson, **Strider GhostBuster: Why It's A Bad Idea For Stealth Software To Hide Files**, Microsoft Research Technical Report MSR-TR-2004-71, July 2004.

[S06] http://www.symantec.com/enterprise/security_response/weblog/2006/06/raising_the_bar_rustocka_advan.html

[CR05] Bryce Cogswell and Mark Russinovich, **RootkitRevealer**, <http://www.sysinternals.com/Utilities/RootkitRevealer.html>

[F06] F-Secure **BlackLight Rootkit Detector**, <https://europe.f-secure.com/blacklight/>

Bots: A Global Distributed Computing Platform

- The most powerful payload is a control network
 - Giving the attacker, at a single point, the ability to control some or all of the infected systems
- Functionality may include
 - Execute arbitrary commands (shell)
 - Authenticated commands
 - Only the attacker or an authorized agent can run a command
 - Update the software
 - Enables flexibility
 - Attack tools
 - Scan for more victims and attack them
 - DDoS tools
 - Profit tools
 - Spam relays

Botnet Structure: Centralized Servers

- The bot contacts a centralized server
 - Commonly an IRC server
 - Bot then waits for commands to be broadcast on the server
- IRC server is a point of vulnerability
 - Can be detected and tracked
 - Server can be shut down
 - But criminally-important IRC servers, when shut down, have resulted in massive DDoS retaliation
- Why is it still used?
 - Its simple to do
 - Common bot source code implements this already
 - Its good enough for today

Botnet Structure: Polling Dead Drops

- Rather than connect to a server, poll some “dead drop”
 - Web site, newsgroup, or other place where arbitrary users can add content
- Much harder to detect and block
 - Can mimic user behavior and use steganography
- Less responsive
 - No longer real-time
- Botmaster may be somewhat vulnerable to discovery
 - Depending on dead-drop technique and use of stepping-stones

Botnet Structure: Peer To Peer

- Each bot has links to other active bots
 - When a command is sent, the bot broadcasts it to its peers
- Much harder to engineer
 - There have been some small P2P botnets
 - We don't know of large P2P botnets
- Much harder to trace
 - Bot network can act as a stepping-stone system
 - Makes it very difficult to trace the source of commands
- Harder to counter
 - Can be a stealthy P2P network (a'la **Skype**)

The Botnet Rendezvous Problem

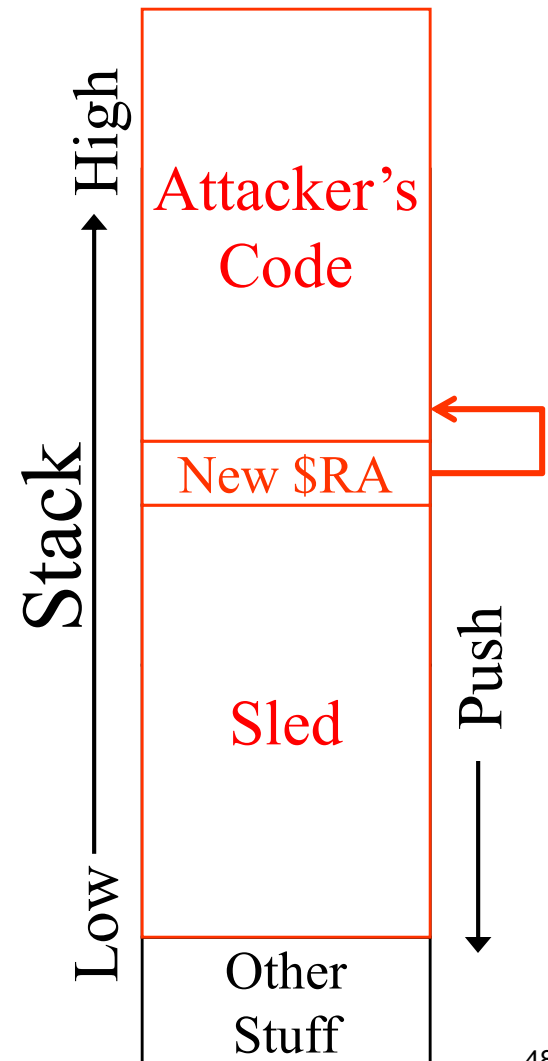
- Just controlling the bots is only part of the problem: the bots need to join the control channel
 - Some piece of information in the bot
- Can be a fixed name or IP
 - Vulnerable to hijacking/recognition
- For P2P, can be the IP of the infecting host
 - Used to enter into the P2P network
 - Needs to cache some information to reconnect later

Exploit Primer

- Worms need to attack the victim system in order to take control
 - How can they do this?
- Buffer Overflows
- Stack Overflows
- Authentication/Trust Attacks
- File Injection Attacks
- Social Engineering
- There are many other types of attacks not covered!
 - These just cover the low-lying fruit for attackers

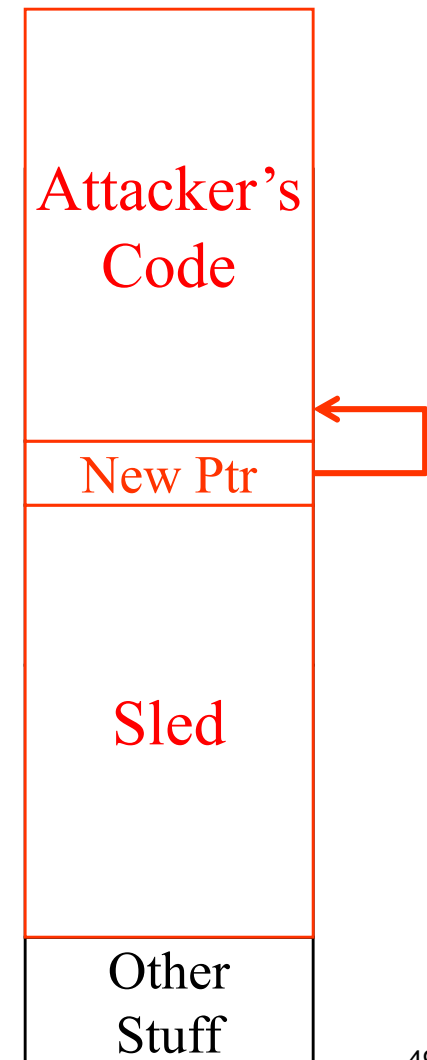
Exploit: Stack Overflow

- Classic “Stack Smashing” attack: Overwrite data past the end of an unchecked buffer
 - Sled: Pre-payload information
 - Payload: Code to be executed
 - Return address: Pointer to the payload
- Overwrites the return address pointer
 - Function now returns into the payload code
- Many defenses based on protecting the return address pointer from being overwritten (without detection):
 - StackGuard, ProPolice, /gc flag
 - On by default in Win2K3 and OpenBSD
 - Also, “Safe” languages prevent this attack



Exploit: Heap Overflow

- Heap contains lots of data
 - Not just buffers, but function pointers in data structures, etc.
 - Overflow (or other piece) corrupts a function pointer to change control flow
- Harder to engineer
 - But not that much harder
- Fewer defenses currently available
 - Requires more checks to prevent exploitation
 - Thus, PointGuard, etc., are not widely used
 - Obfuscation/randomization helps
 - OpenBSD randomized memory allocator
 - Layout constantly changing, harder to exploit with confidence
 - Only makes the jump more difficult, does not prevent code injection
 - Nonexecutable data memory (W xor X)
 - Prevents injected code from running, but doesn't prevent the forced jump from taking over the program



Exploit: User Authentication

- If a worm controls a user's account
 - It can know anything the user tells the machine
 - It can do anything the user can do
- If a worm controls a machine
 - It can control all current users, as well as the machine itself
- Worms can leverage this to spread
 - Morris worm: attempt to rlogin/rsh as the currently infected user
 - Takes advantage of .rhosts
 - Would work as well for ssh with agent forwarding enabled
 - Open Shares worms: Connect as the current user as well as anonymous
 - Especially effective if a network administrator is infected
 - Can capture passwords, or active agent-forwarding
 - Commonly used by human attackers to leverage a single exploit into an institution-wide attack

Exploit: File Injection

- Open Shares
 - World readable/writable file space
 - Some shares are executed automatically
- Other network services allow remote users to read/write to disk space
 - Web server permissions (write to /cgi-bin/)
 - FTP servers can allow remote users to write to disk
 - Root on one machine \Rightarrow homedirs for all users
- Ways to use a writable disk to attack
 - Overwrite /etc/{passwd, hosts.equiv, inetd.conf} .ssh/authorized_keys
 - Write to startup menu, registry, etc.

Exploit: Social Engineering

- Social Engineering deserves its own tutorial!
- How do you persuade a victim to execute the attack on himself?
 - “Please click here...” (porn, figure enhancers, performance enhancers, free cable, monitor spouse, cool tool, freeware, ...)
 - Humans are incredibly gullible creatures!
 - They will even run an executable contained in a zip file in an attachment/message called “test”!
 - They will even decrypt a zip file using a password in a captcha in the email to run an attachment!
- How do you remove the user’s ability to hurt himself and still allow him to do useful work?

Near-Term Worm Defense Overview

- Automated Detection: Determine that a worm is operating on the Internet
 - What strategies does a worm use, what services are targeted, and what systems are vulnerable (a vulnerability signature)?
 - If possible, an attack signature
- Automated Analysis: Given numerous sensors and other devices, create an understanding of the worm
 - How virulent? Are current defenses effective?
- Automated Response: Change the systems to resist further infection
 - Need multiple levels of response to minimize disruption
- Prevention: Make the system infection-resistant
 - Not just patching, but topological changes to the network
 - But must preserve usability
- Recovery: With human involvement, restore service after an attack
- Tolerance: Can we stand some degree of compromise for a limited time?
- Attribution & Retribution: Who did this? Can we prosecute/punish?

Worm Defense: Automatic Detection

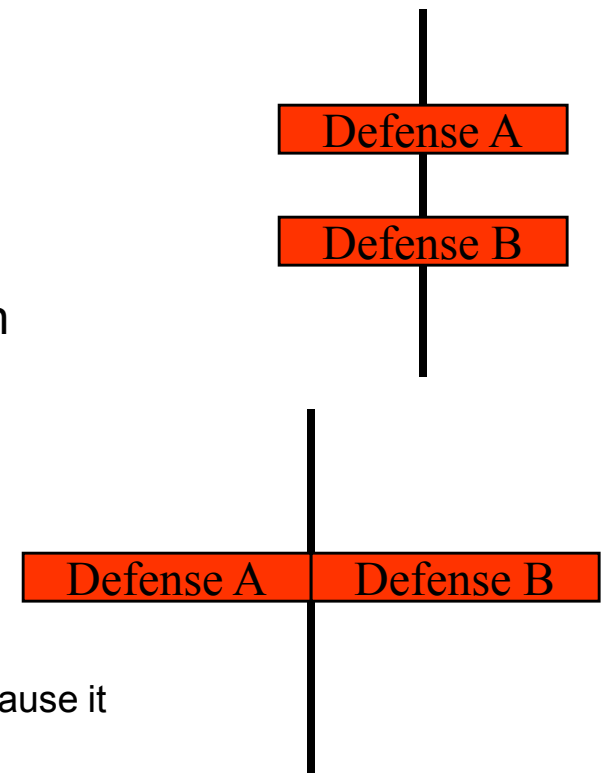
- Automatically detect new worms
 - Old worms are trivial to discover, just listen on a large network
- Without reliable detection, can't build a dynamic defense!
 - Detection → Analysis → Response cycle needs to be faster than the worm can spread
 - May be a problem for topological & hitlist worms
- Detection must detect behavior
 - Propagation behavior of worms
 - E.g., scanning, ability to infect machines, etc
 - Attempts to exploit machines
- What should detection provide:
 - Behavioral signature: Behavior needed for the worm to spread
 - Reconfigure network to block this behavior
 - Vulnerability signatures: what types of systems are vulnerable
 - Ideally, including patch-level
 - Attack signatures: An actual attack-string representing the worm

False Positives and Negatives In Automatic Detectors

- We primarily consider malicious and accidental false positives
- False Negatives
 - Made the attacker work more to achieve the same result
 - Bypassing defenses may require reduced virulence
- False Positive Vulnerabilities
 - Attacker must find a vulnerability in the defense
 - Can then generate a single/few false-positive events
- Systemic False Positives
 - Attacker can generate a false positive at will
 - Defenses get removed
- Our worry: **Network Autoimmune Disease** and **Induced Network Autoimmune Disease**

Composing Detectors: Defense In Depth

- Given two detectors A and B
 - Independent false positive rates P_{fpa} and P_{fpb}
 - Independent false negative rates P_{fna} and P_{fnb}
 - The composition of the two detectors
- Serial composition (defense in depth)
 - Either A or B triggers an alarm
 - $P_{fp} = P_{fpa} + P_{fpb}(1-P_{fpa}) = P_{fpa} + P_{fpb} - P_{fpa} * P_{fpb}$
 - $P_{fn} = P_{fna} * P_{fnb}$
- Parallel composition (reduce false positives through aggregation)
 - Both A and B need to trigger an alarm
 - $P_{fp} = P_{fpa} * P_{fpb}$
 - $P_{fn} = P_{fna} + P_{fnb}(1-P_{fna}) = P_{fna} + P_{fnb} - P_{fna} * P_{fnb}$
- Composing defenses offers no “free lunch”
 - Will either increase false positives or false negatives
 - Our research favors defenses with low false positives because it enables serial composition
 - Additionally, worms are rare events



Worm Defense: Automatic Analysis

- This capability is very limited
 - Offline analysis is even a limited capability
- Two areas:
 - Automatic analysis of worms:
 - Source/binary analysis
 - Look at the definition of the pathogen itself
 - Undecidable in general case
 - Very difficult in the average case
 - Armoring the code makes it nigh impossible
 - Behavioral analysis/black box testing
 - Look at the behavior in an (isolated/live) environment
 - Difficult to extract or characterize (any) all behaviors
 - Network level analysis:
 - Are the defenses working?

Worm Defense: Automatic Response

- Given the results of detection and analysis, change the network to resist further infection
 - Block worm-behavior (behavior signatures):
 - e.g. TFTP may be required for spreading, so block that service completely in the internal network
 - Protect vulnerable machines (vulnerability signatures):
 - Constantly maintain an inventory of systems
 - Using nmap or other continual monitoring tools
 - When a worm is detected, block communication to the vulnerable systems
 - Or inspect those systems more closely for signs of trouble
 - Block attack strings (attack signatures):
 - If an attack signature is available, block it at the network/host level
- Easiest automatic component to implement
 - But still nontrivial to develop
- If responses prevent infections, they can be placed on the end hosts
- If responses contain infections, they must be outside of the end host:
 - In the network, on the NICs, or similar locations

Worm Defense: Prevention

- Prevention requires modifying
 - Visibility: Complete isolationism
 - Vulnerability: Perfectly secure software
 - Attackability: No effects to attacks
 - Infectability: Code can't propagate, execute remotely
- Restrictions are based on the necessary conditions needed for a worm to propagate
 - There's a natural tradeoff between the security of a system and its utility. If there is any general utility, then worm infection is possible.

Worm Defense: Recovery

- Things will go wrong, how can the overall system be restored?
 - Keep data from being lost
 - Be able to resume normal operation
- Restoring systems in a hostile environment:
 - Newly reinstalled systems may be vulnerable to immediate reinfection
- Restoring systems quickly:
 - Restore operation in seconds to minutes rather than hours (or days)
- Restoring systems in parallel:
 - $O(1)$ task, not $O(n)$ task to restore a large number of systems
- Physical disasters are different but related
 - Worms (shouldn't) destroy hardware, so recovery should be easier
 - But worms aren't geographically isolated, making recovery harder

Worm Defense: Tolerance

- Can you accept some degree of infection?
 - What percentage of machines can be compromised?
 - Can you treat all machines the same?
- Defenses are substantially easier when you can tolerate some infections
 - Automatically quarantine infected machines
- Can you tolerate complete infection for a limited time?
 - If so, optimize recovery and data-preservation today and, after that, just don't worry

Worm Defense: Attribution

- Who actually did this? And Why?
- Often a very hard problem:
 - Technically very hard to track back to the initial point of infection
 - Even then, where did THAT come from
- Other hard legal problems:
 - How do you PROVE that the attacker did this
 - When the attacker can wipe his systems before you can arrest him
- Successful attribution is usually through non-technical means

Successful Attribution

- The Morris Worm
 - “Oops” message
 - Now a very successful professor at MIT working in networking and distributed systems
- Blaster copycat
 - Renamed the executable “penis32.exe”, with an added phone-home to the server in his basement
 - Sentenced to 18 months in prison + 3 years probation & community service
- Sasser
 - Turned in by his “friends” for \$250,000
 - Suspended sentence, hired into the security industry
- Melissa
 - Initial infection was a newsgroup posting:
Traced back to a stolen AOL account
AOL account accessed by suspect
Verified by Microsoft Word’s unique ID feature
- ILoveYou
 - One author first tried to submit the worm as a masters report



Failed Attribution

- Witty:
 - Attacker was likely an insider. They apparently:
 - Had a **10 day warning** on the vulnerability
 - Outsiders had < 36 hours
 - **Knew a set of targets** to release the worm
 - Hit list required insider knowledge
 - Had a **grudge**
 - Malicious payload
 - We even **know the IP of the system** it was launched from
 - Traced back to a European ISP
 - Probably only a couple of suspects, but how to prove it?
 - The worm was released through a (presumably compromised) system, who's identity wasn't discovered until months later
 - The attacker was smart: probably eliminated any records
- Others: Blaster, Slammer, Nimda, Code red *, Welchia
 - No clue who wrote them

The Breadth of the Problem

- Worms target communicating network monocultures, not just Windows based PCs
 - Have seen plenty of Unix worms
 - Minority installations are not “safe”
 - 12,000 victims can enable a viable worm (Witty)
- ATMs and nuclear powerplant data acquisition systems have been infected by worms
 - Embedded systems built on a COTS design:
Windows 2K/XP, on an x86, communicating with IP
- Mobile devices (cell phones, PDAs) attractive targets
 - Devices can contact other devices
 - Criminals know how to make money (pay-line scams)
 - OSs becoming more general purpose, becoming monocultures
 - Difficult to upgrade/repair if compromised