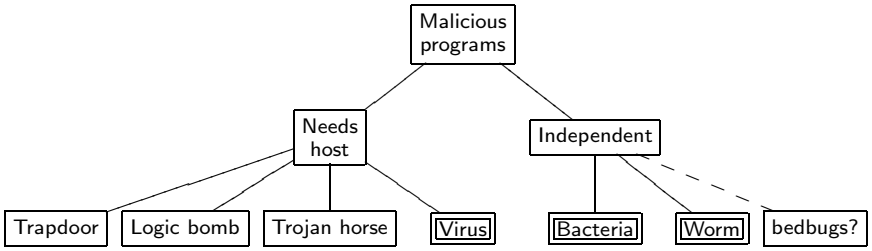
Die Würmer!

CSC 790

Fall 2014

Malicious Program Categories



- Can categorize based on independence
  - **Host program** - fragments of programs that cannot exist alone
  - **Independent** - self-contained programs that can be scheduled and run by the operating system
- Can also differentiate based on program replication
  - Some programs may produces copies of itself

Worms

- A worm is a *self-replicating* program
  - Does not require another program to exist or replicate
  - Typically does not require human-action to execute

    So what?

- Four stages of the worm *life-cycle*
  1. **Target selection**, find a potential host (target)
  2. **Exploitation**, compromise the target using a vulnerability
  3. **Infection**, replicate (copy) itself to the target
  4. **Propagation**, find a potential host (target)

     The difference between target selection and propagation?

Worm Life-Cycle Examples

- A victim of the L10n worm would see
  1. **Target selection**, connection attempt on port 53
  2. **Exploitation**, a BIND exploit on port 53
  3. **Infection**, outbound connection to a web-site to download the worm and install a backdoor *Why not load from attacker?*
  4. **Propagation**, series of scans for port 53
- A victim of the Ramen worm would see
  1. Connection attempts on ports 21, 111, and 515
  2. Exploit `wu-ftp` and/or `rpc.statd`
  3. Download worm and install backdoor
  4. Scan for ports 21, 111, and 515

Famous Worms

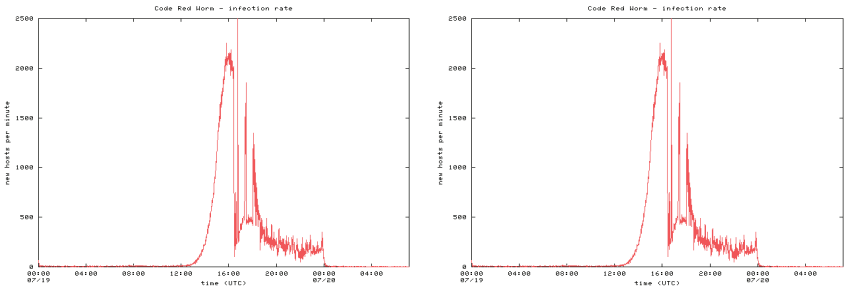
Name	Date	Distinction
Morris	11/88	Used multiple vulnerabilities, looked local
ADM	5/98	Random scanning for addresses
Ramen	1/01	Exploited three vulnerabilities
L10n	3/01	Rootkit worm
Cheese	6/01	Vigilante worm, that smoked your stash
Code Red	7/01	First significant Windows worm, memory resident and delicious
Walk	8/01	Recompiled at the host
Nimda	9/01	Windows worm, C2S, C2C, S2S, ...
Scalper	6/02	11 days after announcement, P2P
Slammer	1/03	Only requires single UDP packet, messed w/ yo GF/BF
Witty	5/04	Attacks ISS security software (not funny)
Zotob	8/05	"covered live on CNN, as the network's computers got infected"
Stuxnet	9/10	Targets SCADA using USB devices for propagation (*hippynet?*)

Code Red

- Initial version released July 13, 2001
  - Vulnerability with Microsoft Index Server 2.0
  - Sends its code as an HTTP request
  - HTTP request exploits buffer overflow

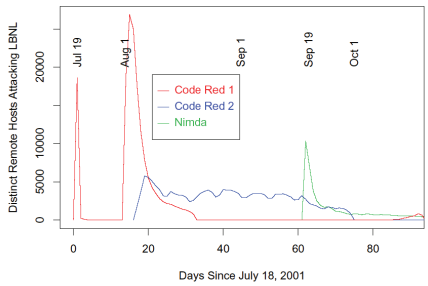
```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090
%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=aHTTP/1.0
```

  - Malicious code exists in memory
- When executed it...
  - Checks is worm already exists (`c:/Notworm`), if it exists the worm goes into infinite sleep state
  - Creates new threads, depending on the date the threads...



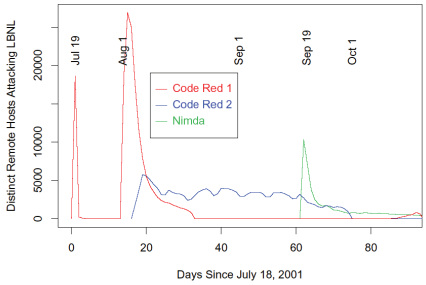
- Initial release
  - 1^{st} through 20^{th}, 99 threads created for random IP search and spread
  - 20^{th} through the end of the month, flood `www.whitehouse.gov`
  - Failure of random seed generator resulted in linear growth
- Vulnerable population (360,000 servers) infected in 14 hours
- *Only requires one instance to start it again...* (August 1^{st} above)

Code Red Part 2



- Released August 4, 2001
  - Commented in code as "`Code Red 2`", but entirely new code base
- Payload is a rootkit, can withstand reboots
  - Contains a bug, crashes NT, only works on Windows 2000
  - Performed *localized scanning* and kills Code Red 1
  - Safety valve added, program terminates October 1, 2001

Nimda



- Released September 18, 2001, multiple forms to spread
  - Attack IIS servers via infected clients and emails copies
- Worm creates a type of *ecosystem*
  - Copies itself on network shares and modify web pages with exploit
  - Scanned for Code Red 2 backdoor (*whoze messin w/ yo GF now?*)
- Leaped across firewalls (*yes, physically leaped*)

Nimda Target Acquisition

- Nimda uses probabilities for generating target IP addresss
  - 50% of the time the first 16 bits of the address fixed, remaining least significant bits random
  - 25% of the time the first 8 bits of the address fixed, remaining least significant bits random
  - 25% of the time the entire address was random
- Localizes network propagation and finds local hosts

  Why? What is the advantage?

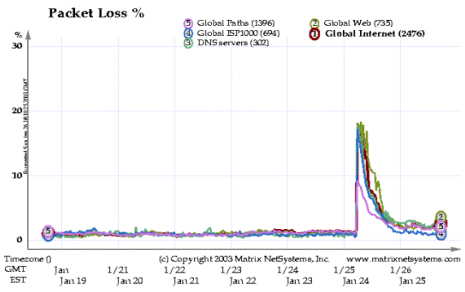
Slammer (Sapphire) Worm

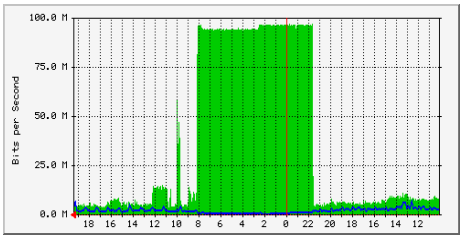
- January 2003: UDP worm exploiting overflow in Microsaft SQL Server
  - Overflow was already known and patched by Microsoft
  - But not everybody installed the patch
- Entire code fits into a single 404 byte UDP packet
  - Normal buffer overflow
  - Packet contents, worm binary then overflow pointer back to itself

> *"The worm is so small that it does not contain code to write itself to disk, so it only stays in memory, and it is easy to remove. For example, Symantec provides a free removal utility, or it can even be removed by restarting SQL Server (although the machine would likely be immediately reinfected)."*

- Random scanning used to propagate
  - Randomly generates IP addresses and sends itself to port 1434
  - MS-SQL listens at port 1434

Slammer Impact





- Approximately scan rate 55,000,000 addresses per second
- Initial infection was **doubling** in 8.5 seconds
  - *Doubling time of Code Red was 37 minutes*
- Saturated carrying capacity of the Internet in 10 minutes
  - 75,000 SQL servers compromised
  - Broken pseudo-random number generator used for IP address

generation had no impact...

- No malicious payload, but it stopped several critical infrastructures
  - Bank of America ATM network, entire cell phone network in South Korea, five root DNS servers...

  Why was Slammer so successful?

- A really bad estimation can provide some insight
  - Packet sent to propagate was 3232 bits
  - There are 2^{32} possible address in the Intertubes (IPv4)
  - Need to send about 1.38×10^{13} bits total to contact everyone
  - Gigabit Ethernet data rate is 1×10^{9} bps
  - Used UDP protocol

  What are some of the horribly wrong assumptions?

Target Selection

- A worm must find a target before it can infect
  - Several different methods for finding potential targets
- **Scanning**, testing a set of addresses, two simple approaches
  - Sequential scanning tries *blocks* of addresses
  - Random scanning tries addresses outside of blocks
- **Pre-generated lists**, testing predefined addresses (*hit-list*)
  - Allows the existence of *flash-worms*
- **Externally generated lists**, testing predefined addresses
  - List is maintained by a separate server (*metaserver*) or source
  - For example, use Google to discover web-servers
- **Contagion**, use normal communications
  - Parasitically use communications initiated by the user

- **Internal target list**, testing predefined *local* addresses
  - Many hosts keep certain addresses local (`/etc/hosts` file)
  - Such target lists can be used to create *topological worms* which discover the topology of the network
  - Warhol or Flash worms generally use target lists

    So what?

- **Passive**, worm does not seek new hosts
  - Wait for a legitimate connection to the current host
  - Has no anomalous traffic patterns during target selection phase, so very difficult to find

Nimda Target Acquisition

- Nimda uses probabilities for generating target IP addresses
  - 50% of the time the first 16 bits of the address fixed, remaining least significant bits random
  - 25% of the time the first 8 bits of the address fixed, remaining least significant bits random
  - 25% of the time the entire address was random
- Localizes network propagation and finds local hosts

  Why? What is the advantage?

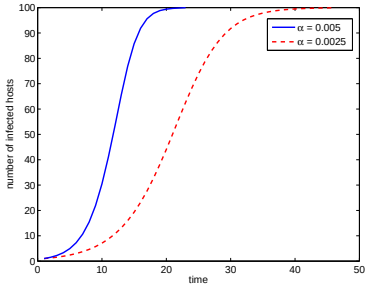
Infection Rate

- Consider a discrete-time epidemic model

$$I_t = (1 + \alpha)I_{t-1} - \beta I_{t-1}^2$$

  - Where I_t is the infected population at time t, α infection rate per infected host, β is the pairwise rate of infection

- Assume all hosts are vulnerable and a simple uniform scan worm



  - The curve depicts *slow start*, *fast spread*, and *slow finish* phases

Worm Distribution

- *How does the worm propagate to the victim?*
  - Again, several different approaches
- **Self carried**, worm actively transmits code
- **Second channel**, worm uses a different channel to transmit code
  - Blaster required a secondary channel to send the code (`TFTP`)
- **Embedded**, worm transmits code during **normal** communication
  - Appending to or replacing normal messages
  - As a result, *propagation does not appear anomalous*
  - Only useful if target selection is also stealthy

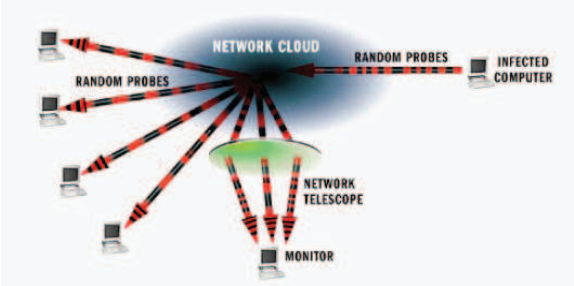
Using Search Engines

- Random address searching can be detected
  - Typically associated with malware, attempts to propagate
- Possible to use search engine to find vulnerable servers
  - Used by MyDoom.0 and Santy
  - Santy exploited phpBB and used Google to ...

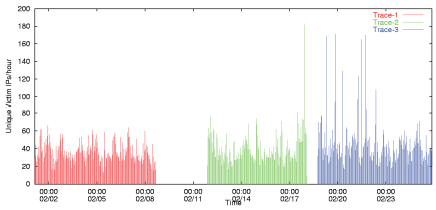
    "exploits bug in the phpBB bulletin system that allows an adversary to run arbitrary code on the web server. To find vulnerable servers to infect, it uses Google to search for URLs that con-tain the string viewtopic.php"
- *Is this a more intelligent way of propagating the worm?*

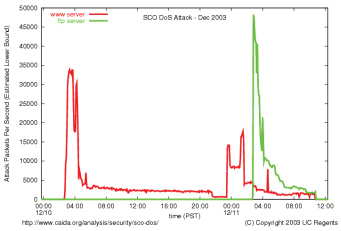
  "Although, the number of infected machines stayed in the low thousands during the outbreak, the actual query traffic was larger as infected web servers were often well connected and ended up running multiple instances of the worm for each vulnerable virtual host."

Network Telescopes for Detecting an Attack



- Designed to observe large-scale events in a network (Internet)
  - Observe dark portion of the network space and extrapolate
  - All traffic in this space should be suspicious
- For example, attacker floods victim with spoofed source addresses
  - Victim believes requests are legitimate, and responds





  - CAIDA's dark network observes $\frac{1}{256}$ of the responses
    How big is the dark address space in this case? How big does the dark address need to be?

- Resolution is dependent on the number of monitored
  - Larger the dark address space, the better the estimate
  - *Dependent on the random component in the attack...*

  Does this apply strictly to worm attacks?

Defenses for the First Three Stages

- Address hopping
  - Change addresses if time between reconn and exploit
- Detect target selection, notice port scans across IP addresses
  - Test for x events across a y-size time interval
  - Notice one source IP scanning different destination IP

    When does this type of detection not work?

- Detecting or preventing exploits
  - Keep systems updated with latest patches
  - Most worms in existence rely on *known* vulnerabilities

    So patches solve this problem? What about IDS signatures?

- Develop signatures
  - Developed via honeypots
  - Signatures need to be determined very quickly...

Signature Inference

- Want to quickly learn a signature for a worm
  - "Automated Worm Fingerprinting," Singh, OSDI 2004
  - "Autograph: Toward Automated, Distributed Worm Signature Detection," Kim, USENIX 2004
  - "Can I win at Keno? Really? Yes, it is Possible," Cody, 2014
  - "Rick Rolled: Life as Fulp's Student," Chukar, 2014
- Monitor the network and look for strings in worm-like traffic
  - Signatures can be used for content filtering
  - *Or at least that is the hope...*

Content Sifting

- Assume there exists some unique invariant bitstring w across all instances of a worm, there are two consequences
  - Content prevalence, w will be more common in traffic than other bit-strings of the same length
  - Address dispersion, set of packets containing w will address a disproportionate number of distinct sources and destinations
- Content sifting is an attempt to find w with a high content prevalence and high address dispersion and drop that traffic
  - Only 0.6% of the 40 byte substrings repeat more than 3 times in a minute [Savage, UCSD]