Detecting Internet Worms

David Moore

University of California, San Diego

Research Exam – March 17, 2005
Motivation for Worm Defense

- **Speed** - Slammer spread in 10 minutes
- **Virulence** - Blaster infected millions of hosts
- **Malice** - Witty destroyed hard drive data
- **Opportunity** - 1000s of vulnerabilities yearly
Outline

• Motivation
• **Background**
• Detection schemes
• Tying things together
What is a Network Worm?

• Self-propagating self-replicating network program
  – Exploits some vulnerability to infect remote machines
    • No human intervention necessary
  – Infected machines continue propagating infection
A Brief History…

- Brunner describes “tapeworm” program in novel “Shockwave Rider” (1972)
- Shoch&Hupp co-opt idea; coin term “worm” (1982)
  - Key idea: programs that self-propagate through network to accomplish some task
  - Benign; didn’t replicate
- Fred Cohen demonstrates power and threat of self-replicating viruses (1984)
- Morris worm exploits buffer overflow vulnerabilities & infects a few thousand hosts (1988)

Mostly a hiatus for 13 years…
Wake-Up Call: Code-Red
(July 19, 2001) [MSB02]

- 360,000 hosts infected in *ten hours*
- No effective patching response
- More than $1.2 billion in economic damage in the first ten days
- Collateral damage: printers, routers, network traffic
Surprising Speed: SQL Slammer
(aka Sapphire) – January 24, 2003 [MPSSW03]

Before 9:30PM (PST)
• >100,000 hosts infected in ten minutes
• Sent more than 55 million probes per second world wide
• Collateral damage: Bank of America ATMs, 911 disruptions, Continental Airlines cancelled flights

After 9:40PM (PST)
• Unstoppable; relatively benign to hosts
Huge Population: MSBlast
August 11, 2003 [L04]

• Microsoft estimates 8-16 million hosts infected

• Note: this count includes hosts behind NATs, firewalls, and internal networks

• Designed to launch a denial-of-service (DoS) attack against Microsoft
More Novelty: Witty
March 19, 2004 [SM04]

• First wide-spread Internet worm with destructive payload
  • writes 64k blocks to disk at random location, repeatedly

• Launched from a large set of ground-zero hosts
  • >100 hosts

• Shortest interval from vulnerability disclosure to worm release
  • 1 day

• Witty infected firewall/security software
  • i.e. proactive user base

• Spread quickly even with a small population
  • ~12,000 total hosts, 45 minutes to peak of infection
Ability to Defend: Who vs. What

• There are two primary methods of blocking malicious traffic
  – Hosts sending the traffic (who)
  – Content of the traffic (what)

• Advantage of knowing who
  – Anything sent by a malicious host is suspect

• Advantage of knowing what
  – Able to prevent the malicious activity from any host
Monitor Placement of Detection System

- Directly on an end-host:
  - Greatest ability to know a compromise has occurred
  - Least ability to see what other hosts are doing

- On a backbone link:
  - Difficult to know if traffic is malicious or legitimate
  - High visibility of traffic from many distributed hosts

- Between:
  - Sharp transition of knowledge of compromise events
  - Gradual transition of visibility of multiple hosts
A Worm’s Raison d’Être

• As a collective whole a worm wishes to infect as many vulnerable machines as possible

• To achieve this goal, instances of the worm must:
  – Spread - find other vulnerable hosts
  – Replicate - create new instances on those hosts
Exploiting a Worm’s Fundamental Behaviors

• To spread, a worm instance needs to:
  – Chose potential targets
  – Send network packets to the target

  Detection strategy: Find hosts which are unexpectedly connecting to many other hosts

• To replicate, a worm instance sends data to:
  – Exploit the vulnerability
  – Transfer the worm code

  Detection strategy: Find a signature, a portion of worm payload, which identifies the malicious traffic but does not match legitimate network traffic
General Detection Guidelines

Detection results must:

• have few false-positives, to avoid affecting legitimate traffic

• have few false-negatives, to avoid continued worm spread

• be generated rapidly, to contain fast worms [MSVS03]

• be simple enough to check against traffic in near real-time

• be readily distributable in a trustworthy manner
Outline

- Motivation
- Background
- Detection schemes:
  - Content Signature - Network
  - Content Signature – Host/Honeypot
  - Scanning Activity
  - Population Dynamics
- Tying things together
Content Signature - Network

• To replicate, worm must send data to:
  – Exploit the vulnerability
  – Transfer the worm code

• Successful worm ⇒ lots of copies of data
  – Assumption that even with polymorphism, some portion of the data will not change

• ⇒ look for frequently occurring substrings

• However, do not match common legitimate strings
  – “GET / HTTP/1.0” or “@ucsd.edu”
Content Signature - Network

- Autograph [KK04] and Earlybird [SEVS04] both look for frequently occurring substrings in packet payloads.

- Primary differences:
  - Which substrings are sampled and checked
  - Heuristics to minimize generating signatures for legitimate traffic
  - Speed of basic algorithm (online vs. batch)
Outline

• Motivation
• Background

• Detection schemes:
  – Content Signature - Network
  – **Content Signature – Host/Honeypot**
  – Scanning Activity
  – Population Dynamics

• Tying things together
Content Signature – Host/Honeypot

• A honeypot [Spitzner] is a special host dedicated for the purpose of being attacked, compromised or infected

• Honeypot approaches:
  – Normal host running a normal software distribution
  – Virtualized host
  – Specialized light-weight responder software emulating portions of other services
Content Signature – Host/Honeypot

• Since a honeypot has no legitimate users, its behavior is determined by its setup and the influence of unsolicited network traffic

• Steps to generating a signature:
  – Detect that the honeypot has been compromised
  – Determine which network packets were responsible
  – Generate a content signature from packet data

• Honeycomb [KC03] is an example system to create intrusion signatures from the honeyd honeypot
Content Signature – Host/Honeypot

• Patching problems:
  – Bugs often understood before a patch can be made
  – Testing patches is time consuming
  – Miscreants reverse-engineer patches to discover bugs

• The Shield [WGSZ04] project:
  – Proactively protect hosts before patches are available
  – Use vulnerability signatures

• Vulnerability signature:
  – sufficient information to check that traffic does not exploit a bug
Content Signature – Host/Honeypot

- Vulnerability signatures can require a large amount of state to emulate protocols, libraries and applications

- Vulnerability signatures are generally specific to an exact set of software installed on a machine
  - The union of all vulnerability signatures might match a large fraction of legitimate traffic

- ⇒ best suited for use directly on hosts
Content Signature – Host/Honeypot

- Hosts running shield can act similarly to honeypots
  - When traffic arrives which is deemed an exploit, generate a content signature from the data in those packets
  
    - While the vulnerability signature is host specific, the resultant content signature is shareable
  
    - Note, the content signature can not exist until there is an actual, observed exploit
Outline

• Motivation
• Background

• Detection schemes:
  – Content Signature - Network
  – Content Signature – Host/Honeypot
  – Scanning Activity
  – Population Dynamics

• Tying things together
Scanning Activity

• To spread, a worm instance needs to:
  – Chose potential targets
  – Send network packets to the target

• Successful worm ⇒ talks to many distinct hosts
  – For a random scan worm, many connections are to addresses with no actual host

⇒ look for hosts with large “outdegree”

• However, do not match legitimate servers
  – “www.cs.ucsd.edu” or “ns0.ucsd.edu”
Scanning Activity: Connection Tracking

- Measure successful/unsuccessful communication attempts over time
  - Note: judging success generally requires being near the host

- The Williamson algorithm \([W02]\) uses a leaky-bucket approach to limit the rate of connections to hosts not recently seen

- Sequential hypothesis testing \([SJB04,WSP04]\) makes a Bayesian decision by comparing the observed sequence with a random walk
Scanning Activity: Large Outdegree

- Examine hosts which communicate with a large number of other hosts (ignoring success)

- Can be deployed deeper in the network and requires less state than connection tracking techniques

- Other non-worm activity can look similar:
  - Flash crowd to normally quiet web server (legitimate)
  - Port scanning source (malicious)
  - Port scanning victim (legitimate response to malicious traffic)
  - Backscatter from DDoS (legitimate response to malicious traffic)
Scanning Activity: Large Outdegree

• Superspreader algorithm [VSGB05]
  – Designed specifically to solve this problem
  – Can immediately report hosts which cross a threshold

• Traffic summaries [KME05]
  – Large degree report one of many “heavy-hitter” reports
  – Designed to generate reports at fixed time intervals
Outline

• Motivation
• Background
• Detection schemes:
  – Content Signature - Network
  – Content Signature – Host/Honeypot
  – Scanning Activity
  – Population Dynamics
• Tying things together
Population Dynamics

- Random scan worms:
  - Pick targets randomly
  - Common in practice
  - Characteristic infection curve

- Look for traffic matching this growth pattern [ZGGT03]

\[ i(t) = \frac{1}{1 + e^{-\beta(t-T)}} \]
Outline

• Motivation
• Background
• Detection schemes:
  – Content Signature - Network
  – Content Signature – Host/Honeypot
  – Scanning Activity
  – Population Dynamics
• Tying things together
**Tying Things Together**

- Lots of progress on “basic” detection algorithms
  - Efficient, fairly effective, combinable

- Many depend on wide or random scanning
  - Including the content signature algorithms

- Recent work in worm design avoids random scans
  - Can we use the existing building blocks to solve?
  - What new techniques are needed?

- Combinations of host- and network-based detectors could provide best of both worlds
Questions?