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**

UCSD CSE Computer Science and Engineering
INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Mitigation Strategies

- Goals of Response Strategies
- Containment vs. Blocking
- Graduated responses
  - Filtering
  - Throttling
  - Lockdown
- Cell-based responses
  - Arbor Networks Peakflow X
- “White” worms and auto-patching

Objectives of Responses

- To change the network or end host to
  - Prevent the further spread of the worm
    - Stopping it from infecting others: Containment
    - Keep it from entering a system: Blocking
  - …. while minimizing disruptive effects on legitimate activity
- Tradeoff: more effective responses may be more disruptive
  - E.g.: complete system power-down ⇒ perfectly effective at blocking worm's spread, but also completely disruptive
- Tradeoffs require site-by-site weighing.
  - Non-linearity of downtime costs for many networks:
    Down for 5 minutes? Often, no one notices enough to care.
    Down for an hour? Annoying
    Down for a day? Bad
    Down for a week? Bankrupt
  - Disruption of state may be worse than disruption of availability

Containment

- Containment focuses on keeping the worm from getting out of an infected system
  - Often coupled with a local (end-host or in-line switch) detector: Don't just detect the worm, detect and stop it
- Requires universal deployment in the network
  - Tenable in enterprise networks
  - Impractical in the Internet

Blocking

- Focused on keeping worm from getting in
  - Usually requires externally specified signature
    - As a way of knowing what to block
- Can benefit from partial deployment
  - Networks running blocking benefit directly, even w/o broad participation by others
- Distinction (keeping out vs. keeping in) is important…

Differing Requirements For Containment vs. Blocking

Containment:

- Universal Deployment required
  - Thus containment strategies are essentially unworkable for the global Internet
- Not reliable if in end-host
  - Generally requires network deployment
  - In future, can be in VM hypervisor (discussed later)
- Can be purely local
  - Detect and contain a common strategy
    - Thus for scanning worms, for example, it can be very simple

Blocking:

- Partial Deployment effective
- Can be in networks or part of the end-host
  - E.g., integrated into conventional AV
- But requires distributed input
  - Can't generally block with just local information
    - Exception: If "local" network spans many systems, can contain one system and then block infection on others
- Usually requires sophisticated analysis to generate signatures

Filtering

- Define representation of the problem
- Drop traffic that matches it
- One representation: who is infected (address blacklisting)
  - except worm's exponential growth will often outrace it
- Another: what the infection looks like
  - Usually defined as a signature
  - Text / regular expression of payload (hopefully) unique to worm
  - Vulnerability signature
    - Description of the vulnerability the worm exploits
  - Behavior signature
    - Description of (hopefully unique) behavior worm exhibits

Moore, Shannon, Voelker and Savage, *Internet Quarantine: Requirements for Containing Self-Propagating Code*, INFOCOM 2003.

Filtering, con't

- End-host filtering (blocking):
  - Easy to implement, but only protects each system individually
    - Can't effectively contain, only block, without a TPM/VM due to potential subversion of mechanism by the worm
- Network-level filtering (blocking & containment):
  - Can protects large groups of diverse systems
  - But can be hard to implement
    - TCP stream reassembly
    - May require application parsing
    - Inline

Vulnerability Signatures

- Observation: injected code might be polymorphic, but exploit is (partially) fixed
  - **DACODA** formulation [CSWC05]: ε, γ, π model of exploitation
    - ε: Input to force the target server to the exploitable point
    - γ: The change in control flow
    - π: The actual payload
  - Rather than describe attack, describe process of exploitation (ε)
    - As the other parts can be highly variable
- In the network:
  - Describe string/expression/app.-elements which capture ε
- On the end-host:
  - Describe a string or control-flow path
  - Describe a change in the host program

[CSWC05] Jedidiah R. Crandall, Zhendong Su, S. Felix Wu, and Frederic T. Chong. **On Deriving Unknown Vulnerabilities from Zero-Day Polymorphic and Metamorphic Worm Exploits**. CCS 2005

Network-Based Vulnerability Signatures

- Use automated analysis to create a regular expression to describe ε
- End-host analysis (**Vigilante** [CCR04], **Sting** [NS05], **DACODA**)
  - Guarantees completeness
  - May be overly broad for the actual worm
- Network-level analysis of multiple instances of the attack (**Polygraph** [NKS05])
  - No completeness guarantee; can be overtrained
  - But captures the practice of the worm

[CCCR04]
Manuel Costa, Jon Crowcroft, Miguel Castro, and Antony Rowstron. **Can we contain Internet worms?** Hotnets 2005
[NS05] J. Newsome and D. Song. **Dynamic Taint Analysis: Automatic Detection, Analysis, and Signature Generation of Exploit Attacks on Commodity Software**. NDSS 2005.
[NKS05] J. Newsome, B. Karp, and D. Song. **Polygraph: Automatically**

Network-Based Vulnerability Signatures, con't

- Open question: by how much can ε vary by at the textual level?
  - It depends on the exploit
- The rest of the attack can be arbitrarily metamorphic
- Code Red: if observe "get *.{ida|idb} *?*", and exceeding a given length:
  - Likely actionable because .ida / .idb with? argument is rare
- Slammer: UDP port, one byte, exceeds given length, γ in limited range
  - Likely only actionable if you aren't using that port

End-Host Based Vulnerability Signatures

- At the end-host, defender has more information
  - Can monitor the program
  - Can perform significantly more computation
    - No need for separate TCP stream reassembly
      - Though still might need application parsers
  - Can afford much more state
- Model the vulnerability as
  - A state machine on input (**Shield** [WGSZ04])
  - Use the program as the state machine (**Vigilante**)
    - Dynamically patch the vulnerable point in the program
- Much more precise model should yield substantially fewer false positives; but requires much broader deployment

[WGSZ04]
Helen J. Wang, Chuanxiong Guo, Daniel R. Simon, and Alf Zugenmaier, **Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits**, SIGCOMM 2004

Addressing Fragility Using Virtualization

- A general problem for end-system defenses: when the system is corrupted, all bets are off
  - But current x86 systems now support much better virtualization
- A general theme-in-development: Place security primitives in a hypervisor layer below the OS
  - All potentially damaging communication must go through the hypervisor
    - Can monitor all disk writes, network traffic, and other behaviors
- Also very useful for rapid recovery: rollback and restart the VM

Throttling

- Idea: trade off uncertainties in detection for less drastic response ….
- …. But one that still impedes the worm.
  - E.g., on detection, limit source to 1 TCP SYN/minute
    - Slows potential worm by one to two orders of magnitude
- Can't halt the worm, but can buy time
  - For some more extensive external analysis process to make a higher-confidence decision
- Can also consider routinely injecting delay to allow real-time analysis procedure to get ahead of the worm
  - E.g., delay all SYNs in a LAN by 20 msec so that (non-delayed) communications between local sensors can form aggregate decision about possible worm spread

Lockdown

- Simply block all connections which could be infectious
  - All traffic from a suspicious host
  - All traffic on a particular port
  - All traffic to hosts of a particular type (OS or server)
- Very draconian response
  - But if correct and timely, very effective response
    - No network → no network propagation
  - Delayed forgiveness may be necessary to handle false positives

Cell-Based Containment

- Break network into distinct regions (cells) [S04], monitor boundaries between them
- Goal: keep worm contained inside its cells
- More cells $\Rightarrow$ more effective containment
  - Can see more infection attempts (finer-grained cell boundaries)
  - When cell compromised, assume all hosts within it compromised
  - But more cells costs more

[S04] S. Staniford. **Containment of Scanning Worms in Enterprise Networks**. Journal of Computer Science.

Cell-Based Containment and Epidemic Threshold

- Detection and containment may not be perfect
  - Allow some possibly-infectious traffic to escape a cell
- If worm instance expected to find >1 new victim
  - The worm will still spread exponentially
- If worm instance expected to find <1 new victim
  - Worm spreads logarithmically and will halt its spread

Enhancing Containment

- For scanning worms, make address space more sparse
  - Takes more scans to find victims
    - Buys detector more room to keep worm below epidemic threshold
  - Could use NAT on network border to enable large 10/8 private address space internally
- Cooperative containment: [WSP04]
  - When a cell detects and blocks an infection, it notifies other cells
  - Response: other cells become more sensitive
    - Goal: converge below epidemic threshold
  - Important question to explore: could this cause cooperative collapse?
    - Single false positive (perhaps malicious) → increased sensitivity → more false positives → increased sensitivity → …

[WSP04] N. Weaver, S. Staniford, and V. Paxson. **Very Fast Containment of Scanning Worms**. USENIX Security 2004.

Arbor Networks Peakflow X

- Peakflow X is an internal network monitoring / response suite from Arbor Networks
  - Out-of-band network monitoring based on NetFlow and related analyses
    - Focused on anomalies
- Response: change router / switch configurations
  - Centered around white graph of learned behavior
    - Who talks to whom using what ports
  - Change switch / router configurations to block malicious traffic while still enabling communication specified by white graph
    - Designed to be "safe": bias towards minimizing disruption
    - Cell size a function of switch/router topology

“White” Worms

- Why not use a worm to stop a worm?
- Shock & Hupp’s experiments: controlling the worm a big issue
- Code Green: a passive anti-Code-Red-2 worm
  - Code Red 2 left an open backdoor
  - Upon receipt of a Code Red 2 scan probe, Code Green:
    - Attacks infected system
    - Removes Code Red 2
    - Patches vulnerability
    - Resets system
  - Apparently never released into the wild

“White” Worms, con’t

- Welchia, a “Good” anti-Blaster worm
  - Spread through the same vulnerability
  - Removed Blaster
  - Patched system
  - But NOT a good worm:
    - Ping scanner disrupted major networks (including US Navy/Marine networks)
    - Opened backdoor on infected systems
  - Goodness was simply self-preservation:
    - Remove a competing worm
    - Prevent another competing worm from arising
    - Prevent multiple infections from slowing/destabilizing systems

"White" Worms - Bad Idea Magnet

- Although attractive, they don't work!
- Can't outrace a spreading worm
  - Unless spreading worm is poorly engineered
- Can't displace an existing worm
  - Unless worm fails to patch behind itself
- Cure can be as bad as the disease
  - An anti-Slammer would still cause the same network disruption while it spreads
  - Or can be even worse: Welchia vs Blaster
- Potentially huge legal issues
  - If it gets out of control

UCSDCSE Computer Science and Engineering

ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Frank Castaneda, Emre Can Sezer, Jun Xu. **WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism.** WORM '04.

An Alternative: Reactive Patch Management

- Most attacks are for vulnerabilities where a patch exists
  - But QA to ensure patch non-disruptive takes time
- Idea: Reactive Patch Management
  - While patch undergoing QA, ship copy to all systems
    - If outbreak occurs, automated system triggers immediate installation
    - Otherwise, wait for the regression testing to complete
- Superior to white worms:
  - Faster: trigger can propagate via multicast, patch has already propagated
  - No legal/control issues
- Can even possibly do this for zero-day exploits!

Sidiroglou and Keromytis. **Countering Network Worms Through Automatic Patch Generation**. *IEEE S&P 3(6),* Nov/Dec 2005