Outline

- Morning session (understanding)
  - The 10,000 foot issues
  - Overview and taxonomy
  - Worm history
  - Epidemiological modeling
- Afternoon session (defenses)
  - **Detection**
    - Signature-based
    - **Behavioral**
  - Mitigation

UCSDCSE Computer Science and Engineering

ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Behavioral Worm Detectors

- Principle of Behavioral Detection
  - Detecting structural manifestations of classes of worms rather than their particular attack
- Scan Detection
  - How worm instance locates further victims
- Contact Graphs
  - Who talks to whom
- User Intent
  - How does network activity relate to local user's actions and desires?

The Core Idea of Behavioral Detection

- A worm must propagate
  - Otherwise it is obviously not a worm
- What is a class of behavior that a worm must exhibit in order to propagate?
  - For a particular category of worm
    - There can be an infinite number of worm instances, but the number of worm classes is finite
  - Or for an implementation strategy common to a number of worms
  - And is this different from how the network normally behaves?
- Detect and/or block this behavior
  - Allows robust defenses against classes of worms
- Generally exploit-independent

Scan Detection

- Core idea: detect the worm's attempts to find new victims
- Naïve algorithm: for each host, track number of new connections (or remote addresses) contacted
  - Alarm if exceeds threshold N within an interval ΔT
  - For large values of *N*, clever algorithms can reduce req'd state [V05]
- Problem: how to pick N, ΔT ?
  - No natural values crisply distinguish hostile scanning from hosts w/ high "fan-out" (e.g., email servers, P2P clients, RSS client updates, multi-user systems)
  - Whatever values you pick, worm can still spread by staying under them
    - Since thresholds have to be large, this can be easy to evade by slowing down

[V05] Venkataraman et al, *New Streaming Algorithms for Fast Detection of Superspreaders*, NDSS '05

Scan Suppression - Williamson Virus Throttle

- Idea: benign hosts don't often contact a lot of different, newly visited remote hosts all at once
  - Whereas a scanning worm does exactly this
- End-host element maintains cache of K last destinations visited, rate-limits connections to new destinations to N per second [TW03]
  - Suggestion: N = 1, K = 5
    - Enhancement: when a connection is acknowledged, remove it from the list of pending connections
  - If queue backlog reaches 100 new destinations, block
  - Need to white-list some servers (e.g., SMTP, DNS)
  - Somewhat problematic: Web clients with fanout bursts
  - Potential problem: Still vulnerable to subthreshold scanning

UCSDCSE Computer Science and Engineering

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

[TW03] J. Twycross and M. Williamson. *Implementing and testing a virus throttle*. Proc. USENIX Security 2003

Scan Detection - “Landmines”

- Idea: benign hosts shouldn’t connect to “dark addresses” (unused/unallocated), so presume such access indicates a scanner
- Implemented in some commercial products
  - Forescout, Mirage Networks
    - Looking at the ARP requests as well as SYNs
    - Block behavior by switch changes and/or ARP cache poisoning
- Not clear (= not studied in the literature) what thresholds to use
  - Benign people do make mistakes, after all …
  - … or access stale data
- Similar in spirit to TRW (discussed shortly)

Scan Detection - (lack of) Associated DNS Traffic

- Idea: legitimate connections are preceded by DNS lookups by which the client finds the server [WKO05]
- Therefore, consider connections that lack such a lookup as suspect
- Usual false-positive/evasion issues arise in deciding thresholds
  - Note, problematic for protocols that pass around IP addresses for rendezvous (e.g., multi-homed FTP servers/clients, some URLs, many P2P programs)
    - Small lab deployment: 52 alerts in 1 week. 36 were HTTP related:
      - One problem: Clients don't respect very short DNS TTLs
  - So perhaps use as input to further anomaly detection

[WKO05] D. Whyte, E. Kranakis, and P. C. van Oorschot. *DNS-based detection of scanning worms in an enterprise network*. Proc. NDSS 2005

Scan Detection - ICMP Backscatter

- Idea: attain visibility into worm propagation by analyzing clusters of ICMP Unreachable's [BBM03]
  - Extract port being scanned from embedded transport payload
  - Look for patterns of many sources *plus* many individual destinations receiving probes from many sources
- However: significant issues (for any global detection) due to background radiation [RGL05]
- Spoofed packets can create a malicious false positive

[BBM03] V. Berk, G. Bakos, and R. Morris. *Designing a framework for active worm detection on global networks*. Proc. IWIA '03
[RGL05] D. Richardson, S. Gribble and E. Lazowska. *The Limits of Global Scanning Worm Detectors in the Presence of Background Noise.* Proc. WORM '05

Scan Detection - Threshold Random Walk

- Idea: scanners more likely to fail in connection attempts (to new destinations) than legit sources. Suppose:
  - Legit sources succeed to new dests. $\geq \theta_0$ of the time
  - Scanners succeed $\leq \theta_1$ of the time ($\theta_1 < \theta_0$)
- For each traffic source, formulate two hypothesis, H0 (legit source) and H1 (scanner)
- As source makes new-destination connection attempts, use success/failure of attempt to update a Sequential Hypothesis Testing [JPBB04] statistical model to decide between H0 and H1

UCSD CSE Computer Science and Engineering ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

[JPBB04] J. Jung, V. Paxson, A. Berger, and H Balakrishnan. *Fast portscan detection using sequential hypothesis testing*. Proc. IEEE Symposium on Security and Privacy, 2004.

Scan Detection - Threshold Random Walk, con't

- Statistical model formulated as a variable associated with each source:
  - Initialize to zero
  - On successful connection, increment
  - On failure, decrement
- Progression of variable describes a random walk
- If walk progresses …
  - … High enough above zero, declare H_0 (**legit** source)
  - … Low enough below, declare H_1 (**scanner**)
  - Hence name ***Threshold Random Walk***
- How do you decide what is high or low enough?
  - Can directly derive from values of θ_0 and θ_1 along with *desired false positive/negative rate*

Threshold Random Walk in practice

- From trace analysis, (quite) conservative parameters for an institute's border traffic:
  - $\theta_0 = 0.8$ (i.e. legit sources succeed $\geq$ 80% of the time)
  - $\theta_1 = 0.2$ (i.e. scanners succeed $\leq$ 20% of the time)
- TRW generally detects scanners after 4-5 attempts to connect to new destinations, with a suitably low false-positive/false-negative rate
- Note, for worm detection, application includes a subtle consideration:
  - Worm infectees act normal until they become infected.
  - Thus, their random walk can travel far in the "legit" direction before veering the other direction
  - Solution: "Reverse" TRW. See:

Jung, Schechter, and Berger. **Fast Detection of Scanning Worm Infections**. Proc. RAID 2004.

Threshold Random Walk in practice, con't

- Deployment internal to an enterprise requires different priors, perhaps conditioned on type of application (TBD)
  - (Subtleties also arise due to ARPs, Ethernet broadcasts vs. switching)

- Implementing TRW in hardware: Approximate Caches (AC-TRW) which use fixed memory

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

UCSDCSE Computer Science and Engineering

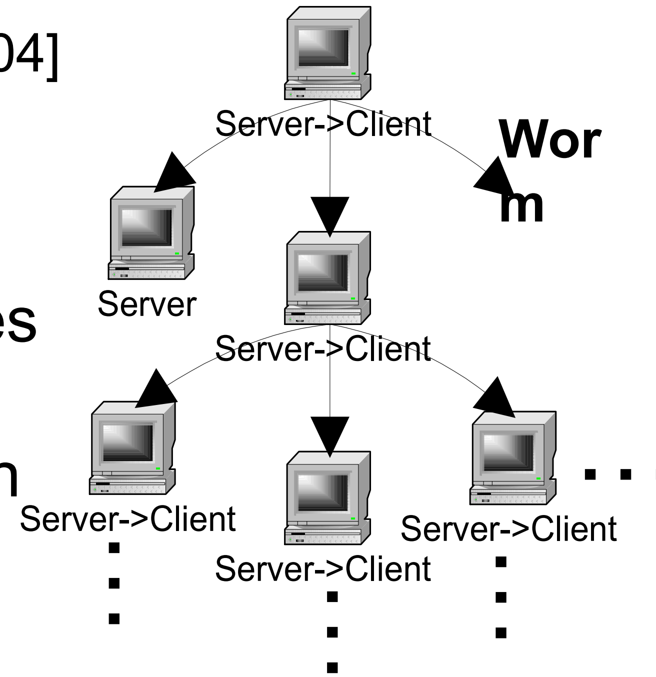
INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Issues with Scan Detection

- In their basic form, most schemes predicated on worm spreading via random address scanning
  - As opposed to email/topological/meta-server/contagion worms
- Most schemes include a threshold. If worm is *efficient* (scans tend to succeed), it might be able to spread while remaining undetected.
  - Some detectors have a subthreshold scanning rate which is sufficient to allow evasion while still being reasonably fast within an enterprise
- Works best for detecting local infections.
  - If the global Internet is infected and scanning, then eventually a source will succeed on its very first attempt -- no opportunity to suppress as a "scan"

Behavioral Detection Based on Communication Patterns

- Idea: worm-spread has unusual *en masse* communication graph [EAAS03,X04]
- What's unusual:
  - Servers become clients
  - The same data/pattern propagates
    - E.g.: Common ports or data
  - Secondary channels may be seen
    - E.g.: TFTP-based infection
  - High fanout sources
- Amenable to headers-only traffic analysis
  - Such as available from NetFlow



[EAAS03] D. Ellis, J. Aiken, K. Attwood and S. Tenaglia. **A behavioral approach to worm detection**. Proc. WORM 2003
[X04] J. Xiong. **ACT: Attachment chain tracing scheme for email virus detection and control**. Proc. WORM 2004.

UCSDCSE Computer Science and Engineering
ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Evaluating Contact Graphs

- Evaluated by monitoring MITRE's internal network at ~10 sensors and recording ~1 year worth of traces
  - Overlay worm-type behavior using worm-emulation
    - Program which accepts XML description of how to communicate in a worm-like manner
- Accuracy/sensitivity (after tuning):
  - ~2 false alarm periods/day, ~20 minutes total alarm time
  - Detection of worms after 4 generations
  - Apt for enterprise-wide defense, but not individual host/small network defense
    - Also benefits from network construction: a clear distinction between clients and servers
  - Can't detect worms in some P2P systems
    - P2P systems may create contact graphs indistinguishable from worms in the P2P system

Behavioral Detection Based On User Intent

- Idea: (most) legit desktop use arises from the local user issuing commands
  - Or from a few, already defined applications
- Track causality between user input and subsequent network traffic [CK05]
  - Keystrokes, mouse clicks
  - Process trees (e.g., desktop click on IE spawns browser)
- Alarm if network activity arises independent of previous user activity
- Symantec uses a similar hack: alarm if email processing leads to certain forms of network connection
  - Heuristic which can detect many email worms

[CK05] W. Cui and R. Katz, *BINDER: An Extrusion-Based Break-In Detector for Personal Computers,* Proc. USENIX ATC 2005

BINDER System Issues

- Issue: automated follow-on activity
  - Refreshing of Web pages, polling for email
  - Solution: allow subsequent connections if linked to previous user-intended one
- Issue: how much time do you allow to lapse between user input & network activity?
  - Solution: default values are 10s of secs for initial connection, 10s of mins for repeats
  - Can use per-user training to sharpen these

BINDER System Issues, con't

- Issue: what about activity because the user was fooled into clicking/typing?
  - Notion: catch follow-on activity after system boots
- Issue: Autonomous benign activity
  - System daemons, auto-update, start-up activity
  - Solution: white-list, as number is not large

BINDER Efficacy

- User study (modest) finds very low false-alarm rate (~ 1-2/user/week)
  - Could be improved by
    - Better tracking of inter-process event sharing
    - Better whitelisting
- Detected several actual instances of spyware installed on test-user machines
- In testbed, detected email worm
- Pending evasion issues (usual arms race). Malware:
  - Injects apparent user-input
  - Tricks user into entering input
  - Leverages single connection created upon (user-mediated) infection, keeps it open indefinitely
  - Hides inside other processes or subverts whitelisted processes