Summary From the Last Lecture
- Well-publicized worms
- Worm propagation curve
- Scanning strategies (uniform, permutation, hitlist, subnet)

Worm Defense
- Three factors define worm spread:
  - Size of vulnerable population
    - Prevention – patch vulnerabilities, increase heterogeneity
  - Rate of infection (scanning and propagation strategy)
    - Deploy firewalls
    - Distribute worm signatures
  - Length of infectious period
    - Patch vulnerabilities after the outbreak

How Well Can Containment Do?
- This depends on several factors:
  - Reaction time
  - Containment strategy – address blacklisting and content filtering
  - Deployment scenario – where is response deployed
- Evaluate effect of containment 24 hours after the onset


How Well Can Containment Do? Code Red
- Idealized deployment: everyone deploys defenses after given period

How Well Can Containment Do? Depending on Worm Aggressiveness
- Idealized deployment: everyone deploys defenses after given period

How Well Can Containment Do? Depending on Deployment Pattern
- Fig. 4: Containment effectiveness as a function of deployment scenario.
How Well Can Containment Do?

- Reaction time needs to be within minutes, if not seconds
- We need to use content filtering
- We need to have extensive deployment on key points in the Internet

Detecting and Stopping Worm Spread

- Monitor outgoing connection attempts to new hosts
- When rate exceeds 5 per second, put the remaining requests in a queue
- When number of requests in a queue exceeds 100 stop all communication


Detecting and Stopping Worm Spread

Cooperative Strategies for Worm Defense

- Organizations share alerts and worm signatures with their “friends”
  - Severity of alerts is increased as more infection attempts are detected
  - Each host has a severity threshold after which it deploys response
- Alerts spread just like worm does
  - Must be faster to overtake worm spread
  - After some time of no new infection detections, alerts will be removed

"Cooperative Response Strategies for Large-Scale Attack Mitigation", Proceedings of DISCEX 2003, D. Netijit, J. Rowe, K. Lewis

Cooperative Strategies for Worm Defense

- As number of friends increases, response is faster
- Propagating false alarms is a problem

"Cooperative Response Strategies for Large-Scale Attack Mitigation", Proceedings of DISCEX 2003, D. Netijit, J. Rowe, K. Lewis
Early Worm Detection

- Early detection would give time to react until the infection has spread
- The goal of this paper is to devise techniques that detect new worms as they just start spreading
- Monitoring:
  - Monitor and collect worm scan traffic
  - Observation data is very noisy so we have to filter new scans from
    - Old worms’ scans
    - Port scans by hacking toolkits


Assumptions

- Worms uniformly scan the Internet
- No hit lists but subnet scanning is allowed
- Address space scanned is IPv4

Monitoring System

- Provides comprehensive observation data on a worm’s activities for the early detection of the worm
- Consists of:
  - Malware Warning Center (MWC)
  - Distributed monitors
    - Ingress scan monitors – monitor incoming traffic going to unused addresses
    - Egress scan monitors – monitor outgoing traffic

Worm Propagation Model

- Simple epidemic model: \( \frac{dI}{dt} = \beta I (N - I) \)

![Graph showing worm propagation model]
**Monitoring System**
- Ingress monitors collect:
  - Number of scans received in an interval
  - IP addresses of infected hosts that have sent scans to the monitors
- Egress monitors collect:
  - Average worm scan rate
- Malware Warning Center (MWC) monitors:
  - Worm’s average scan rate
  - Total number of scans monitored
  - Number of infected hosts observed

**Worm Detection**
- MWC collects and aggregates reports from distributed monitors
- If total number of scans is over a threshold for several consecutive intervals, MWC activates the Kalman filter and begins to test the hypothesis that the number of infected hosts follows exponential distribution

**Code Red Simulation**
- Population: \( N = 360,000 \), Infection rate: \( \alpha = 1.8 \) / hour,
- Scan rate \( \eta = 358 \) / min, Initially infected: \( I_0 = 10 \)
- Monitored IP space \( 2^{20} \), Monitoring interval: \( \Delta = 1 \) minute

**Slammer Simulation**
- Population: \( N = 100,000 \)
- Scan rate \( \eta = 4000 \) / sec, Initially infected: \( I_0 = 10 \)
- Monitored IP space \( 2^{20} \), Monitoring interval: \( \Delta = 1 \) second

**Dynamic Quarantine**
- Worms spread very fast (minutes, seconds)
  - Need automatic mitigation
  - If this is a new worm, no signature exists
  - Must apply behaviour-based anomaly detection
  - But this has a false-positive problem! We don’t want to drop legitimate connections!
  - Dynamic quarantine
    - “Assume guilty until proven innocent”
    - Forbid network access to suspicious hosts for a short time
    - This significantly slows down the worm spread

**Dynamic Quarantine**
- Behaviour-based anomaly detection can point out suspicious hosts
  - Need a technique that slows down worm spread but doesn’t hurt legitimate traffic much
  - “Assume guilty until proven innocent” technique will briefly drop all outgoing connection attempts (for a specific service) from a suspicious host
  - After a while just assume that host is healthy, even if not proven so
  - This should slow down worms but cause only transient interruption of legitimate traffic

---

Dynamic Quarantine
- Assume we have some anomaly detection program that flags a host as suspicious
  - Quarantine this host
  - Release it after time T
  - The host may be quarantined multiple times if the anomaly detection raises an alarm
  - Since this doesn’t affect healthy hosts’ operation a lot we can have more sensitive anomaly detection technique

Simulation Setup
- Initially 75,000 vulnerable hosts
- Quarantine rate of infectious host is $\lambda_1 = 0.2$ per second
  - Infectious host will be quarantined after 5 seconds on average
- Quarantine rate of susceptible host is $\lambda_2 = 0.00002315$ per second
  - There are 2 false alarms per host per day
- $T = 10$ seconds

Dynamic Quarantine
- An infectious host is quarantined after $\frac{1}{\lambda_1}$ time units
- A susceptible host is falsely quarantined after $\frac{1}{\lambda_2}$ time units
- Quarantine time is T, after that we release the host
- A few new categories:
  - Quarantined infectious $R(t)$
  - Quarantined susceptible $Q(t)$

Slammer With DQ

DQ With Large T?

DQ And Patching?
Patch Only Quarantined Hosts

Cleaning $I(t)$

Cleaning $R(t)$