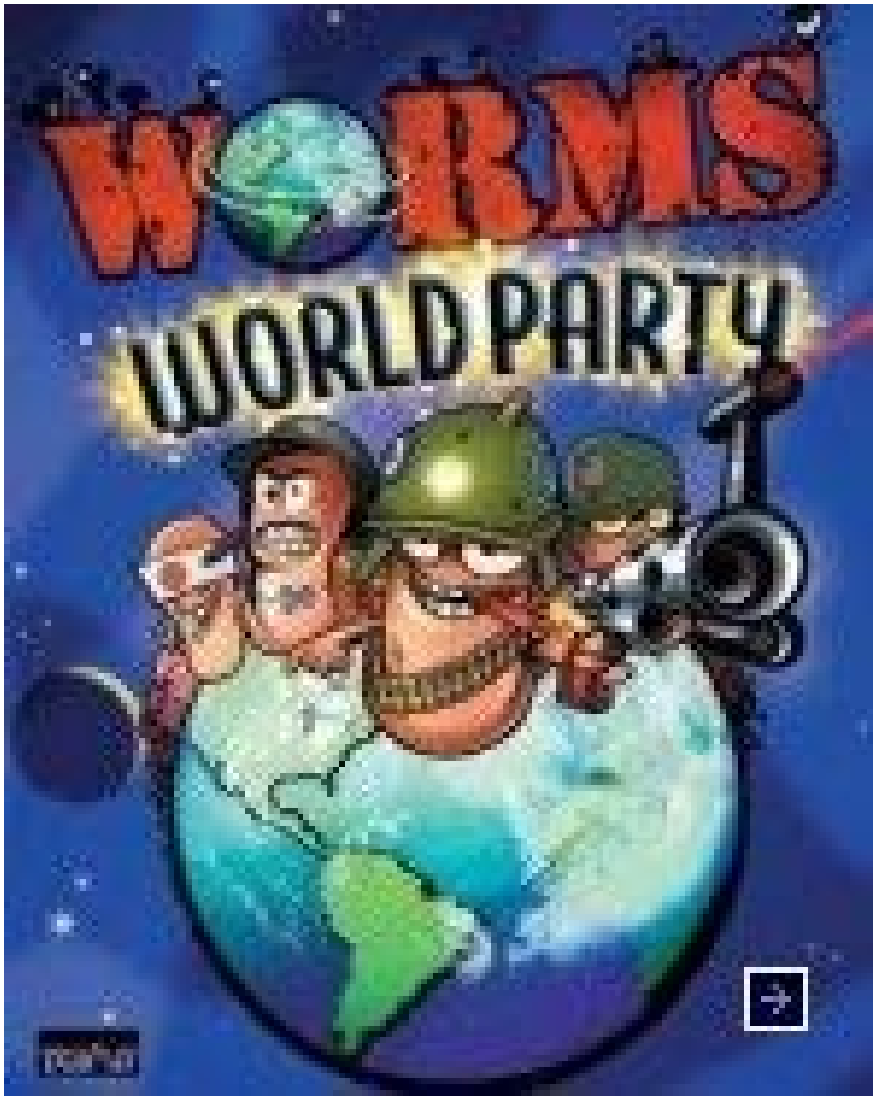
Security: Worms

Presenter: AJ Fink

Nov. 4, 2004

It’s a War Out There



Analogy between Biological and Computational Mechanisms

- The spread of self-replicating program within computer systems is just like the transition of smallpox several centuries ago

Worm Classification

- Spreading Media
  - Scan-based & self-propagation
  - Email
  - Windows File Sharing
  - Hybrid
- Target Acquisition
  - Random Scanning
  - Subnet Scanning
  - Routing Worm
  - Pre-generated Hit List
  - Topological
  - Stealth / Passive

Improvement in Scan Methods

- Subnet Scanning
  - The first goal may be a /24 or /16 enterprise network instead of the whole internet
- Routing Worm
  - Some IP addresses are not allocated
- Pre-generated Hit List Scanning
  - Speedup the propagation and the whole address pace can be equally divided for each zombie

Ways to mitigate the threat of worms

- Prevention
  - Prevent the worm from spreading by reducing the size of vulnerable hosts
- Treatment
  - Neutralize the worm by removing the vulnerability it is trying to exploit
- Containment
  - Prevent the worm from spreading from infected systems to the unaffected, but vulnerable hosts

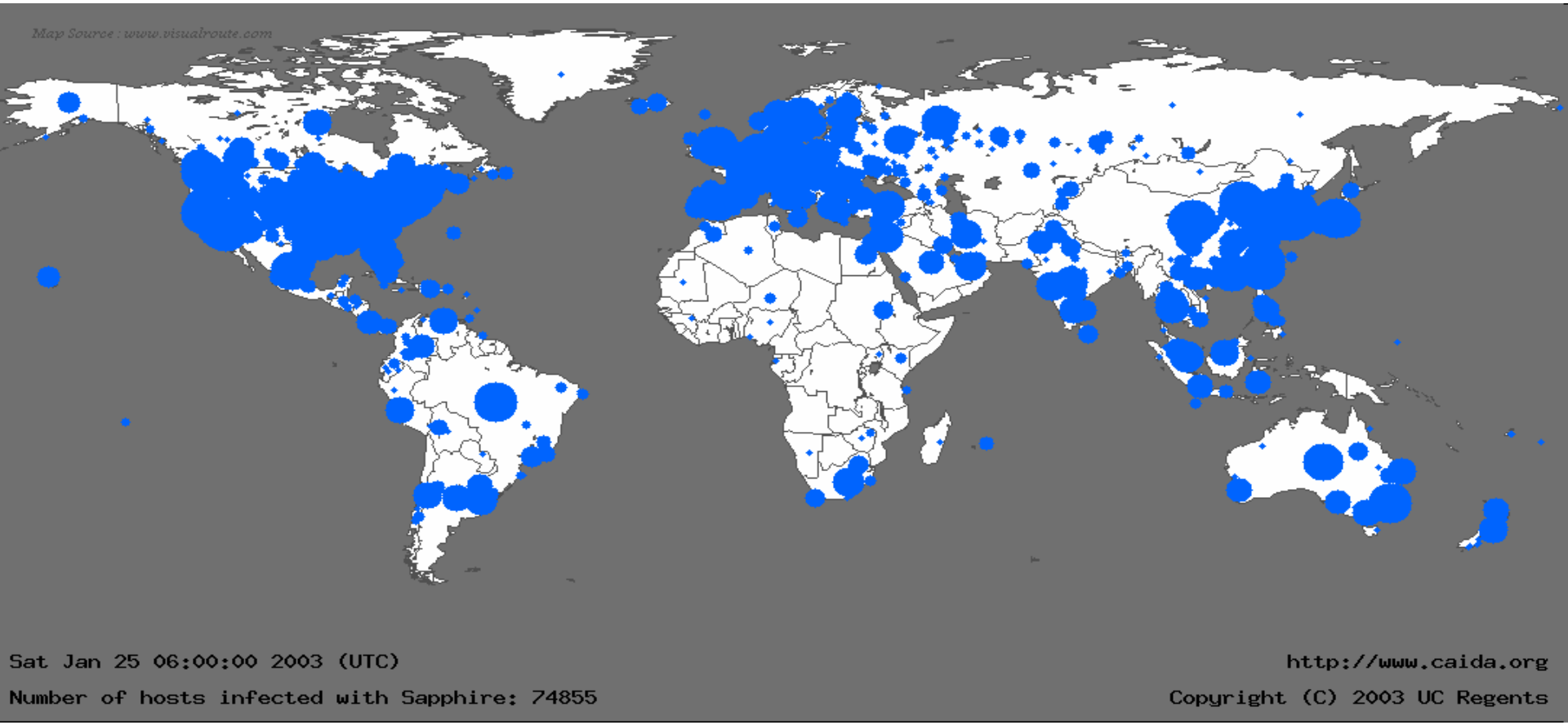
Case Study

Slammer / Sapphire Worm

The Warhol Effect

"In the future, everybody will be world-famous for 15 minutes." –Andy Warhol

- Warhol worms
  - worms that infect most of the targets in <15 minutes.
- Slammer the first Warhol worm
- Infected >90% of vulnerable machines in <10 minutes
- Infected >75,000 machines on Jan 25, 2003
- Doubled in size every 8.5 seconds


Map Source : www.visualroute.com
Sat Jan 25 06:00:00 2003 (UTC)
Number of hosts infected with Sapphire: 74855
http://www.caida.org
Copyright (C) 2003 UC Regents

How?

- Exploit: Microsoft(surprise surprise) SQL Server (buffer overflow)
- Payload contained in a 404 byte UDP packet. All it needed to do was send a single UDP packet to port 1434
- Slammer's scanning only limited by bandwidth
- Largest probe rate was 26,000 scans/sec

What Did Slammer Do?

- Payload was not explicitly malicious
- Infected hosts saturated the network
- DoS attacks caused by Slammer's aggressive scanning
- Crashed network hardware due to such high load

Slammer's Mistakes

according to paper

- Errors in the pseudo random number generator
- Tried to use a linear congruent parameterization:
  - x' = (x * 214013 + 2531011) mod 2^{32}
  - Substituted 0xFFD9613C for 2531011

    (which is equivalent to –2531012)
  - Conversion to a negative number was an error (taking the negative in twos-complement involves flipping all the bits and adding 1, not just flipping all the bits)
  - Also author intended to use SUB instead of ADD since tried to use a negative number

Slammer's Mistakes (con't)

according to paper

- Used XOR instead of OR to clear a register
- Had the effect of XORing with the contents of pointer contained in SqlSort's import address table

Slammer's Mistakes (con't)

according to paper

- The negative number in the pseudo random number generator caused the output to always be even
- The XOR caused the result to be 32-bit aligned
- These combined cause the upper octet of generated addresses to remain constant in any worm execution instance
- As a result each worm instance cycled through a list of addresses much smaller than the actual Internet address space

Slammer's Mistakes Actual Mistakes?

- If the author was intending to use the linear congruent generator:

  $x' = (x * 214013 + 2531011) \bmod 2^{32}$

- First the author had to decide to take the twos-complement to get a negative number
- Then had to mess up the taking of the twos-complement
- Then had to mess up and use ADD instead of SUB
- Its hard to believe that the programmer messed up that much

Slammer's Mistakes Actual Mistakes?

- The paper mentions that they can't tell how many machines were infected due to the "bugs" in the worm
- So the author butchered a linear congruent generator in such a way that we can't tell how many machines were infected
- Or the author new the consequences of the "bugs" and created it such that monitoring it would be difficult?

Very Fast Containment of Scanning Worms

Weaver, Staniford, Paxson

Outline

- Scanning
- Suppression Algorithm
- Cooperation
- Attacks
- Conclusion

What is Scanning?

- Probes from adjacent remote addresses?
- Distributed probes that cover local addresses?
- Horizontal vs. Vertical
  - Scanning for particular services or scanning for all services on a machine
- How to infer intent?
  - Some scans are benign, locating peers in a peer to peer system

Scanning Worms

- Blaster, Code Red, CR II, Nimda, Slammer

- Does not apply to:
  - Hit lists (flash worms)
  - Meta-servers (online list)
  - Topology detectors
  - Contagion worms

Scanning Detection

- Key properties of scans:
  - Most scanning attempts fail
  - Infected machines attempt many scans
- Containment is based on worm *behavior*, not signatures (content)
- Containment by address blocking (blacklisting)
- Blocking can lead to DoS if false positive rate is high

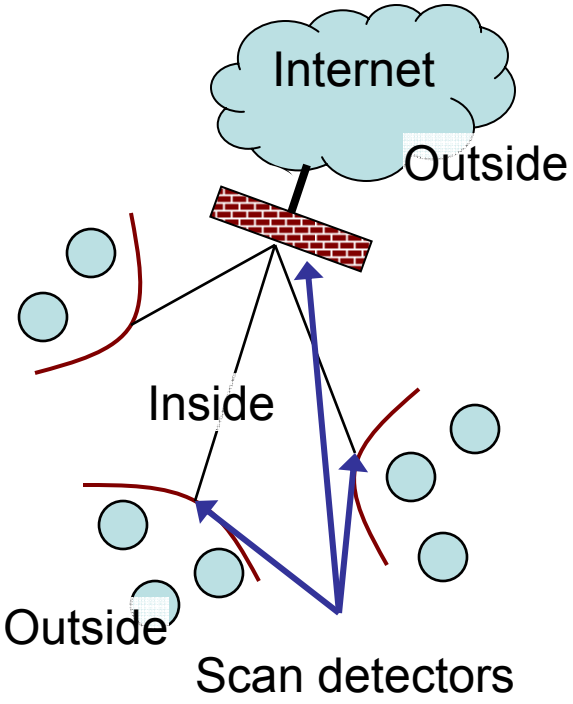
Scan Suppression

- **Goal 1:** protect the enterprise; forget the Internet
- **Goal 2:** keep worm below epidemic threshold, or slow it down so humans notice

- Divide enterprise network into cells
- Each is guarded by a filter employing the scan detection algorithm

Inside, Outside, Upside Down

- Preventing scans from Internet is too hard
- If inside node is infected, filter sees all traffic
- Cell (LAN) is “outside”, Enterprise network is “inside”
- Can also treat entire enterprise as cell, Internet as outside



Scan Suppression

- Assumption: benign traffic has a higher probability of success than attack traffic
- Strategy:
  - Count connection establishment messages in each direction
  - Block when *misses* – *hits* > *threshold*
  - Allow messages for existing connections, to reduce impact of false positives

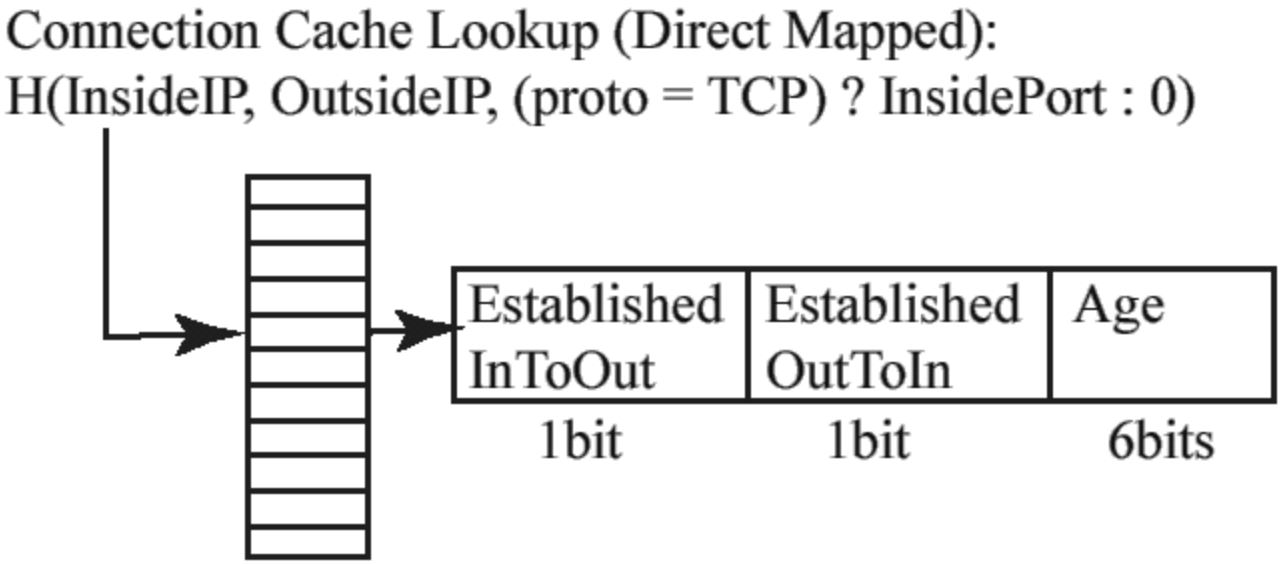
Constraints

- For line-speed hardware operation, must be efficient:
  - Memory access speed
    - On duplex gigabit ethernet, can only access DRAM 4 times
  - Memory size
    - Attempt to keep footprint under 16MB
  - Algorithm complexity
    - Want to implement entirely in hardware

Mechanisms

- Approximate caches
  - Fixed memory available
  - Allow collisions to cause aliasing
  - Err on the side of false negative
- Cryptographic hashes
  - Prevent attackers from controlling collisions
  - Encrypt hash input to give tag
  - For associative cache, split and save only part as tag in table

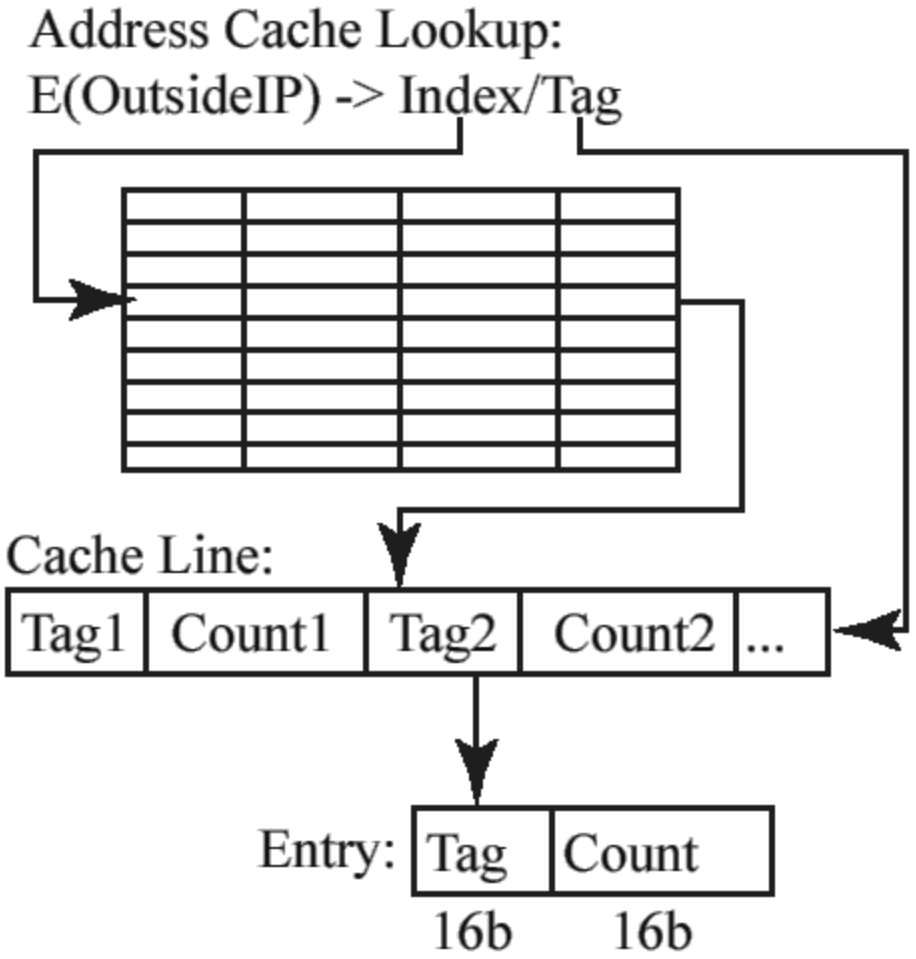
Connection Cache



- Remember if we've seen a packet in each direction
- Aliasing turns failed attempt into success (biases to false negative)
- Age is reset on each forwarded packet
- Every minute it purges entries older than D_{conn}

Address Cache

- Track “outside” addresses
- Counter keeps difference between successes and failures
- Counts are decremented every D_{miss} seconds



Algorithm Pseudo-code

Condition:
SrcIP = InsideIP

```
If(!EstablishedInToOut)
   if(EstablishedOutToIn)
      # Was previously
      # recorded as a miss
      # but is now a hit
      Count <- Count - 2
   EstablishedInToOut <- True
Age <- 0
Forward packet
```

Condition:
SrcIP = OutsideIP &
Count < Threshhold

```
If(!EstablishedOutToIn)
   if(EstablishedInToOut)
      # Record as a hit
      Count <- Count - 1
      EstablishedOutToIn <- True
   else if(hygiene_drop)
      Drop packet
   else
      # A possible miss
      Count <- Count + 1
      EstablishedOutToIn <- True
if(!DroppedPacket)
   Age <- 0
   Forward packet
```

Condition:
SrcIP = OutsideIP &
Count >= Threshhold

```
# Address is being blocked
if(EstablishedInToOut)
   if(isSYN | isUDP)
      # No matter what, drop
      Drop packet
   else if(!EstablishedOutToIn){
      # Record as a hit
      Count <- Count - 1
      EstablishedOutToIn <- True
   # Internally requested or old
   # connection, so allow
   Age <- 0
   Forward packet
else
   Drop packet
```

Performance

- For 6000-host enterprise trace:
  - 1MB connection cache, 4MB 4-way address cache = 5MB total
  - At most 4 memory accesses per packet
  - Operated at gigabit line-speed
  - Detects scanning at rates over 1 per minute
  - Low false positive rate
  - **About 20% false negative rate**
    - Is this 20% false negative rate low enough?
    - Should the connection cache be made larger, less aliasing?

Scan Suppression – Tuning

- Parameters:
  - T: miss-hit difference that causes block
  - C_{min}: minimum allowed count
  - C_{max}: maximum allowed count
  - D_{miss}: decay rate for misses
  - D_{conn}: decay rate for idle connections
  - Cache size and associativity

Scan Suppression Parameters

- T, C_{min}, C_{max}, D_{miss}, D_{conn} need to be set correctly to make the containment effective
- C_{min} too small and previously good addresses that are recently infected will be allowed to many connections before being blocked
- D_{miss} too small and forgive too many failed connections

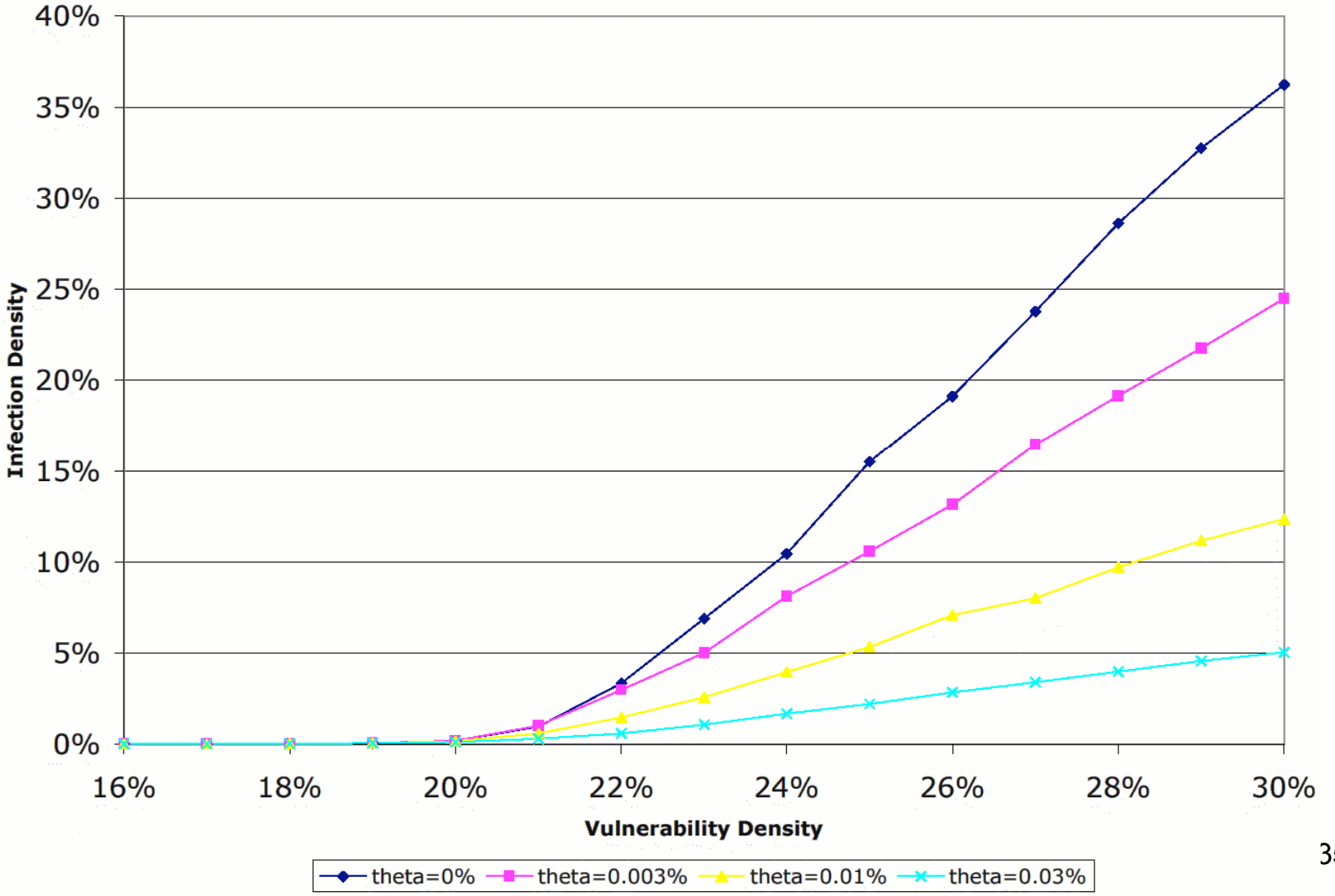
Scan Suppression Parameters

- The parameters effect the hosts flagged by their algorithm, yet there is no discussion on how to set the parameters
- All they did was present 2 different tests with different sized caches and different parameter values

Cooperation

- Divide enterprise into small cells
- Connect all cells via low-latency channel
- A cell's detector notifies others when it blocks an address ("kill message")
- Blocking threshold dynamically adapts to number of blocks in enterprise:
  - $T' = T - \theta^X$, for very small θ
  - Changing θ changes the infection density

Cooperation – Effect of θ



Cooperation Issues

- Poor choice of θ could cause collapse
- Lower thresholds increase false positives
  - No information on how much it increases false positives for their tests
- Should a complete shutdown be possible?
- How to connect cells (practically)?
  - They assumed instantaneous communication

Attacking Containment

- False positives
  - Spoofing outside addresses
    - This only prevents legitimate outside address from making connections to the enterprise
- False negatives
  - Use a non-scanning technique
  - Scan under detection threshold
  - Use a whitelisted port to test for liveness before scanning

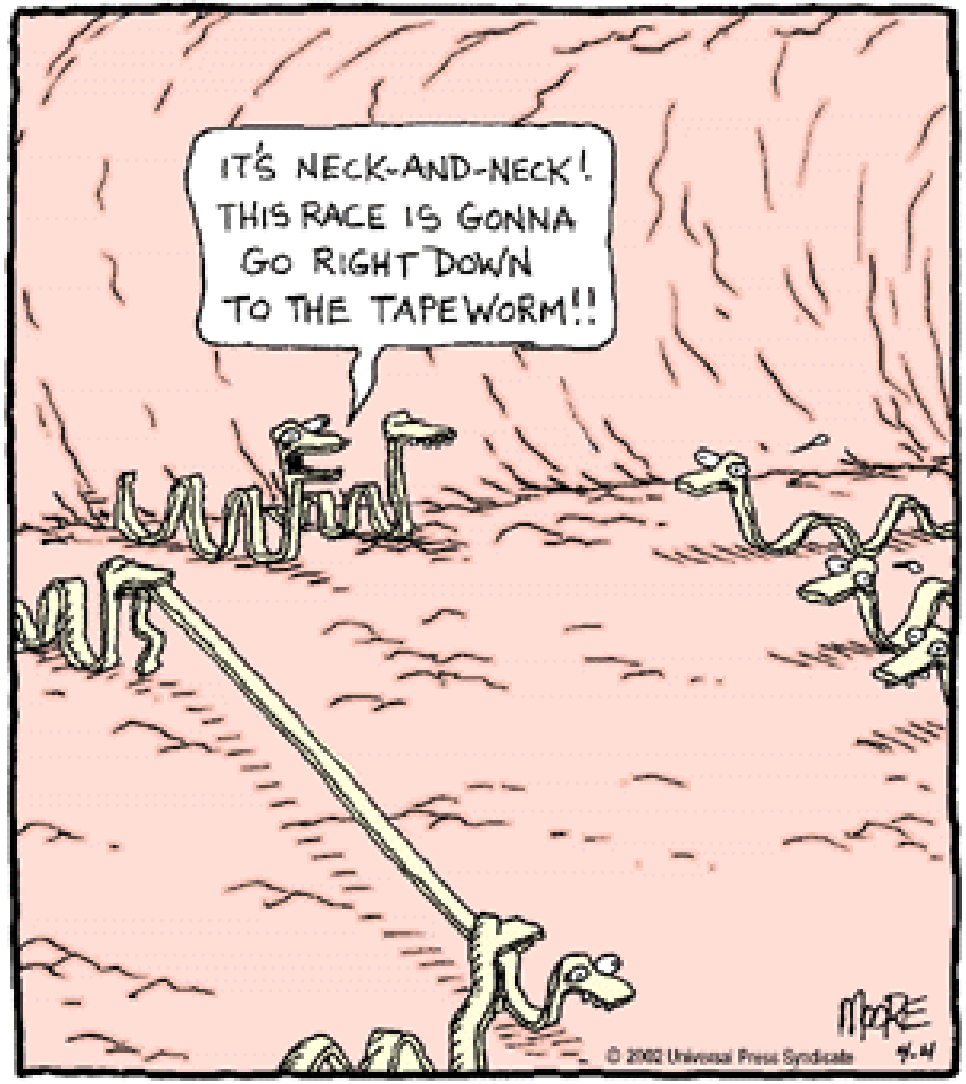
Attacking Containment

- Detecting containment
  - Try to contact already infected hosts
  - Go stealthy if containment is detected
- Circumventing containment
  - Two-sided evasion:
    - Inside and outside host initiate normal connections to counter penalty of scanning

Attacking Cooperation

- Flood cooperation channels
  - Attempt to outrace containment if threshold is permissive
- Cooperative collapse:
  - False positives cause lowered thresholds
  - Lowered thresholds cause more false positives
  - Feedback causes collapse of network

Conclusion



Intestinal Parasite Olympics.