Very Fast Containment of Scanning Worms

Nicholas Weaver, Stuart Staniford, Vern Paxson

ICSI, Nevis Networks, ICSI & LBNL

Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Scanning Worms

- Worms
  - Malicious, self-propagation programs
  - Represent threat to large network
- What's scanning worm?
  - Pick "random" addresses, attempt to infect them
    - Blaster – linear scanning
    - Code Red – fully random
    - Code Red II & Nimda – bias toward local address
- Common properties of scanning worms
  - Most scanning attempts result in failure
  - Infected machines institute many connection attempts

Spread of Scanning Worms

- How to mitigate the spread of worms?
  - Prevent
    - Reduce size of vulnerable population
    - Single victim can infect millions of vulnerable hosts
    - Complete infection of local network from single original source
  - Treat
    - Once a host is infected, clean it up immediately
    - Using antivirus software, patches
    - Limitation: long time to develop cleanup code
  - Containment

Containment

- Purpose
  - Protect individual networks, isolate infected hosts
  - e.g. firewalls, content filter, automated blacklists
- Most promising solution
  - Completely automated
  - No need participation of each host on the Internet
- Reaction time
  - Detection of malicious activity
  - Propagation of containment information
  - Activate any containment strategy

Containment Strategy

- Address blacklisting
  - Maintain a list of IP addresses infected
  - Drop all packets from the addresses in the list
  - Pros: implement easily with existing filtering
  - Cons: update continuously to reflect newly infected
- Content filtering
  - Need a database of content signatures
  - Pros: a single update is sufficient
  - Cons: hard to automatically create signature

Worm Containment

- Defense against scanning worms
  - Detect worms, block infected hosts
  - Based on worm behavior rather than signature (content)
  - Capable to stop new worms
- Break the network into many cells
  - Must have very low false positive rate
    - Cause a DoS if false positive rate is high
  - Need to complete deployment within an enterprise
    - Integrate into network's outer switches

Epidemic Threshold

- Epidemic Threshold
  - Worm-suppression device must necessarily allow some scanning before it triggers a response
  - Worm may find a victim during that time
  - Epidemic occurs if each infection results in a single child
  - Exponential epidemic occurs if results in more than one child

- The epidemic threshold depends on
  - Sensitivity of containment response devices
  - Density of vulnerable host on the network
  - Degree to target, i.e. capability of worms

Sustained Scanning Threshold

- Sustained Scanning Threshold
  - If a worm scans slower than this rate, the detector will not trigger
  - Vital to achieve as low a sustained scanning threshold as possible
  - Target a threshold of 1 scan per minute

- For example

Outline

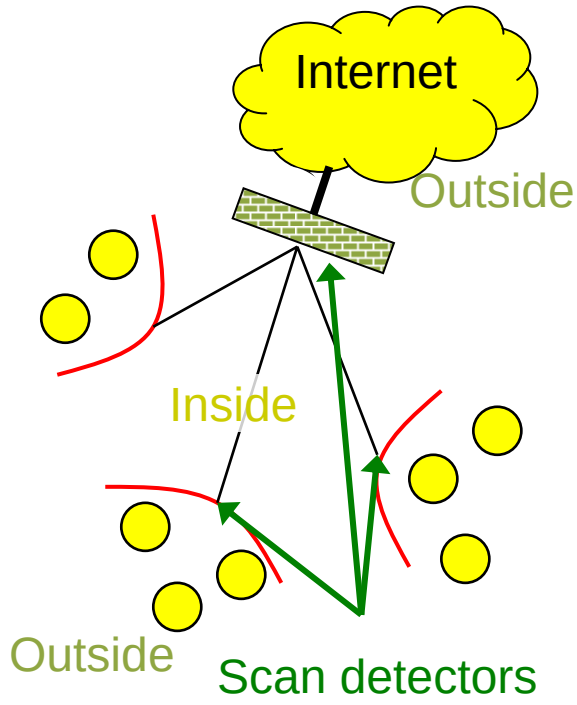
- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Scan Suppression

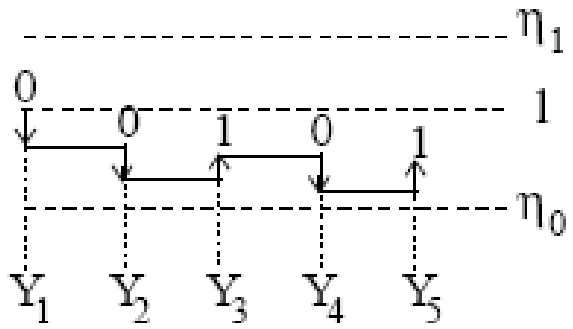
- Scan suppression
  - Responding to detected portscans by blocking future scanning attempts
- Portscans
  - Probe attempts to determine if a service is operating at a target IP address
- Portscans have two basic types
  - Horizontal scans
    - search for identical service on large number of machines
  - Vertical scans
    - examine an individual machine to discover all running services

Scan Suppression

- Protect the enterprise, forget the Internet
  - Prevent scans from Internet is too hard
  - If inside node is inflected, filter sees all traffic
  - Cell (LAN) is outside, enterprise network is inside
  - Can also treat entire enterprise as cell, Internet as outside



Scan Suppression



- Scan detection algorithm derived fror… …eshold Random Walk (TRW)
  - Sequential Hypothesis Testing (SHT) to determine if a connection will fail or succeed
  - Assume that benign traffic has a higher probability of success than attack traffic
  - TRW can make decision by likelihood
- Implementation easier than TRW
  - Suitable for both hardware and software implementation
  - Simplified algorithm caused increased false negative rate
  - No changes in the false positive rate

Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Hardware Implementation

- Constraints
  - Memory access speed
    - Must be very fast to keep up with high packet rates
    - On full duplex Gb Ethernet, access DRAM 4 times per packet
  - Memory size
    - Attempt to keep footprint under 16 MB
  - The number of distinct memory banks
  - Algorithm complexity
- Design goal
  - Less than 16MB of total memory
  - 2 uncached memory accesses per packet
  - Include within conventional NIDS, such as Bro or Snort

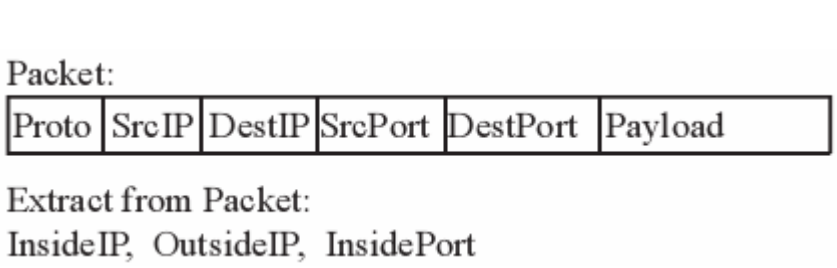
Mechanism

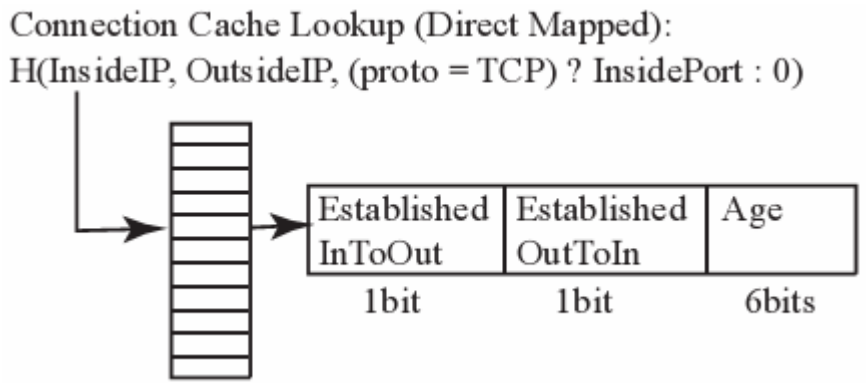
- Approximate caches
  - Collisions cause imperfections (Bloom filter)
  - Fixed memory available
  - Allow collisions to cause aliasing
  - Err on the side of false negative
- Efficient small (32-bit) block ciphers
  - Prevent attackers from controlling collisions
  - Permute the N-bit value
  - Separate the resulting N-bit value into an index and a tag

Approximate Scan Suppression

- Strategies
  - Track connections and addresses using approximation caches
  - Replace old addresses and ports if corresponding entry has timed out
  - Block when counts (misses – hits) > threshold
  - Track addresses indefinitely as long as no evict their states from our caches
  - Implement a “hygiene filter” to thwart some stealthy scanning techniques without causing undue restrictions on normal machines

Connection Cache

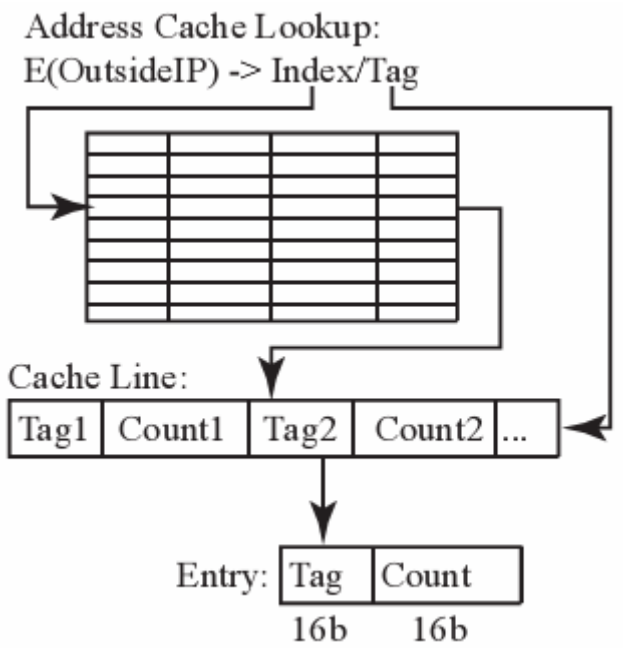




- Record if we have seen a packet in each direction
- Aliasing turns failed connection attempt into success (biases to false negative)
- Age is reset on each forwarded packet
- Every minute, back ground process purges entries older than D_{conn} (10 mins)

Address Cache

- Trace “outside” addresses
- Encrypt them to create index and tag
- Counter tracks differences between misses and hits
- Counts are decayed every D_{miss} seconds (60 sec)
- Hard limit on negative counts (-20)



Algorithm

- Look up the connection table and address table

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
```

Parameter and 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

Evaluation

- 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
    - on DNS and SMTP servers due to fan-out
    - need to be white-listed
  - About 20% false negative rate
  - Detects scanning after 10-30 attempts

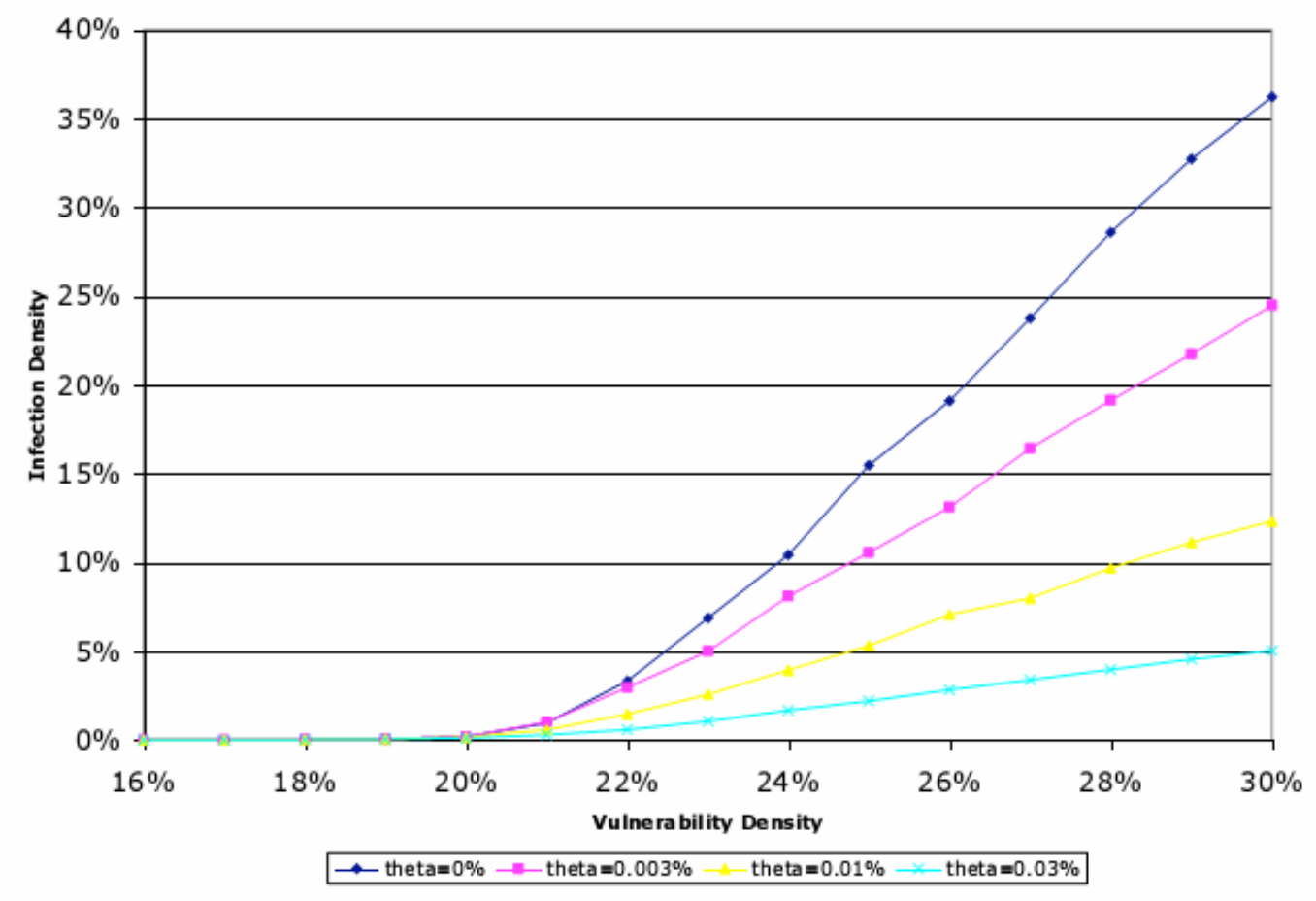
Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Cooperation

- Divide enterprise into small cells
  - Connect all cells via low-latency channel efficiently
  - Cooperate to reduce thresholds during an attack
  - A cell's detector notifies others when it blocks an address ("kill message")
- Blocking threshold dynamically adapts to number of blocks (X) in enterprise:
  - $T' = T(1 - \theta)^X$, for very small θ
  - Control how to reduce threshold as a worm spread
- Avoid cooperative collapse
  - a cascade in sensitivity increase

Cooperation – effect of θ



Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Attacking Worm Containment

- False positives
  - Forge packet, spoofing outside address
  - False positive create a DoS target
- False negatives
  - Use a non-scanning technique (topological, meta-server, passive, and hit-list)
  - Scan under detection threshold (1 scan/min)
  - Use a white-listed port to test for liveness before scanning
  - Offset misses by making valid connections

Attacking Worm Containment

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

Attacking Worm Containment

- Detecting containment
  - Try to contact already infected hosts
  - Go stealthy if containment is detected
- Circumventing containment
  - Embed scan in storm of spoofed packets
  - Two-sided evasion
    - Inside and outside host initiate normal connections to counter penalty of scanning
    - Can modify algorithm to prevent, but lose vertical scan detection

Outline

- Worm Containment
- Scan Suppression
- Hardware Implementation
- Cooperation
- Attack
- Conclusion

Conclusion

- Develop containment algorithms suitable for deployment in high-speed, low-cost network hardware
- Devise the mechanisms for cooperation that enable multiple containment devices to more effectively detect and respond to an emerging infection