Defending against Hitlist Worms using Network Address Space Randomization

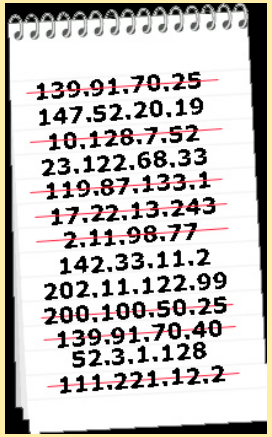
Kostas Anagnostakis

Internet Security Lab, I^{2}R, Singapore

Joint work with: S. Antonatos, P. Akritidis, E.P. Markatos @ FORTH, Greece

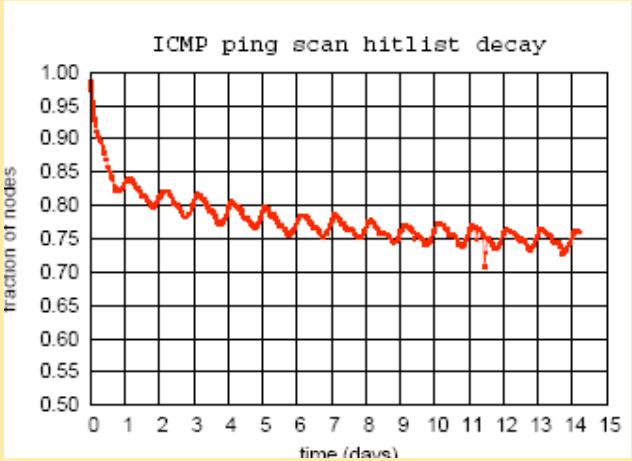
The problem

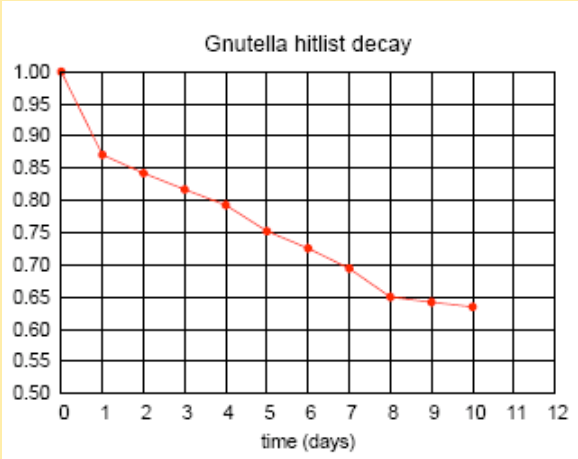
- Hitlist worms use lists of vulnerable hosts, collected in stealth before launching the attack
  - High success rate (compared to scanning) allows such worms to infect millions of hosts in seconds
  - Propagation speed likely to be too fast for any reactive defense proposal
  - Behavior different from scanning worms, making them harder to detect

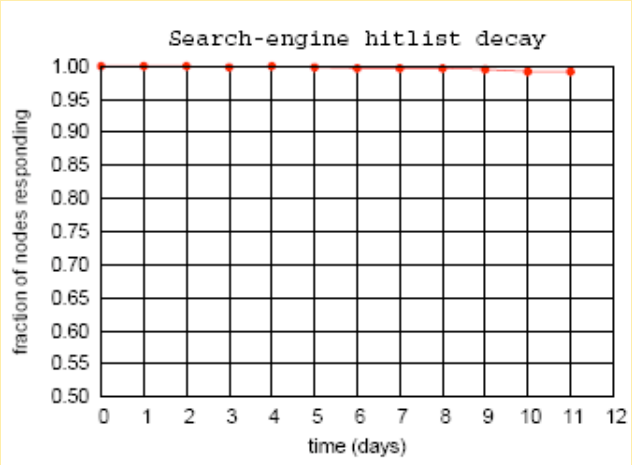


- Various methods for building hitlists
  - Web search e.g., find web servers
  - Gnutella crawling: find active normal hosts
  - Random scanning: ping/nmap random addresses

An early experiment with hitlist generation







- Depending on hitlist generation strategy, hitlist entries tend to become stale at different rates
- **Natural to ask whether we can artificially increase the decay rate of hitlists**

Network Address Space Randomization

- Basic idea:
  - Force hosts to change addresses frequently enough so that hitlist entries have become stale by the time the worm starts propagating
- Questions:
  - Implementation & deployment cost
  - Effectiveness in throttling worm & limiting infection
  - Collateral damage

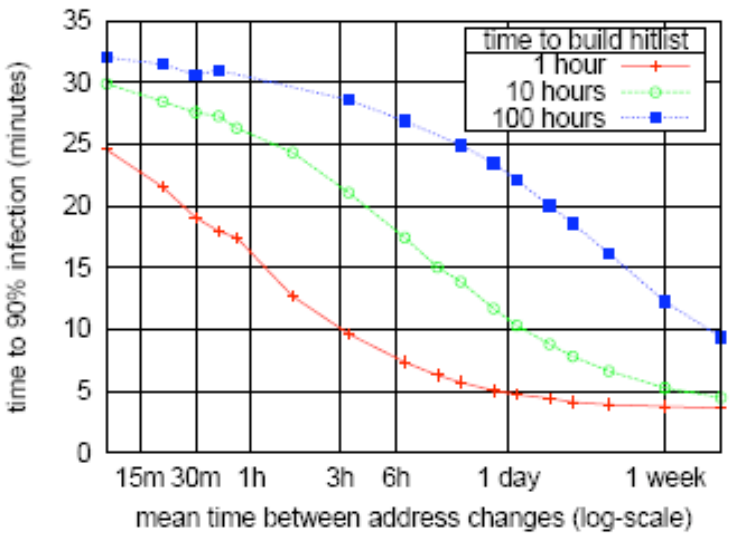
NASR implementation

- Modified DHCP server
  - Address change enforcement by changing lease time+semantics
  - Interface with libpcap app to minimize impact on applications
- In addition to DHCP lease timer, two internal timers
  - Soft timer triggers change only when host has no active connections
  - Hard timer triggers change regardless of host activity
  - DHCP lease timer controls how frequently clients should poll server
- Alternative implementation on gateway
  - Static internal address, randomized external address
  - To avoid breaking connections, old <IP,port> pairs are kept alive until connection terminates
  - More expensive, but protects against failures

Experiments: overview

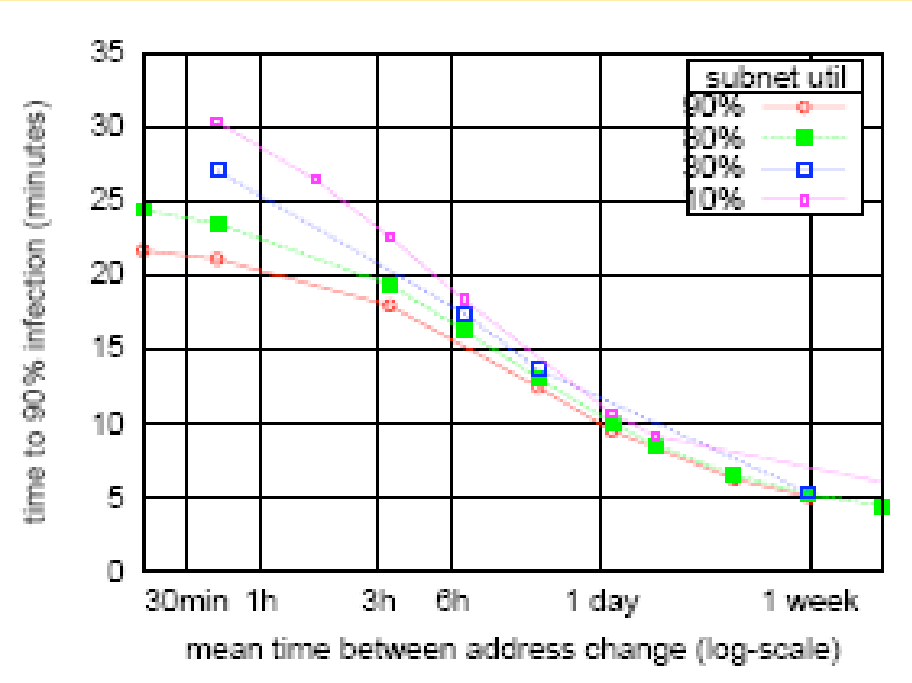
- Model-based simulations to evaluate effectiveness
  - 1M hosts, flat network topology
  - Ignore congestion, traffic effects
  - Pure hitlist & hybrid hitlist+scanning worm
- Trace-driven simulation to evaluate cost
  - Several traces from diverse environments

Experiments: effectiveness of NASR



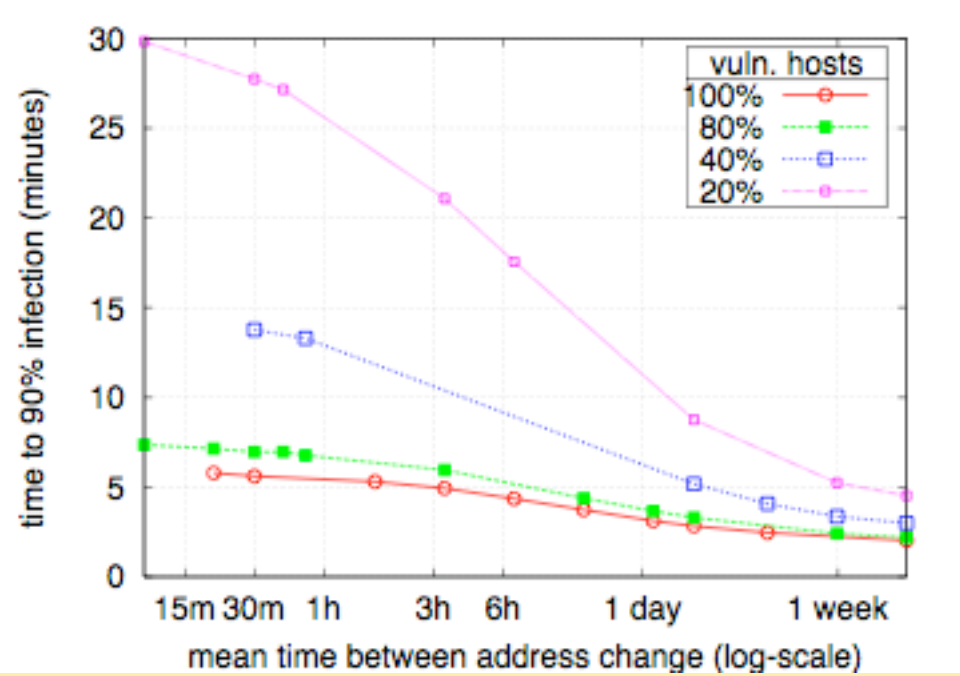
- Assumptions:
  - 20% of hosts are vulnerable
  - Scan success probability: 0.3
  - Hybrid hitlist+scanning worm
- Results:
  - No NASR: worm needs 5 minutes to spread
  - With NASR: 24 to 32 minutes when we have frequent changes
  - **4 to 5 times slow down**

Effectiveness vs. address space utilization



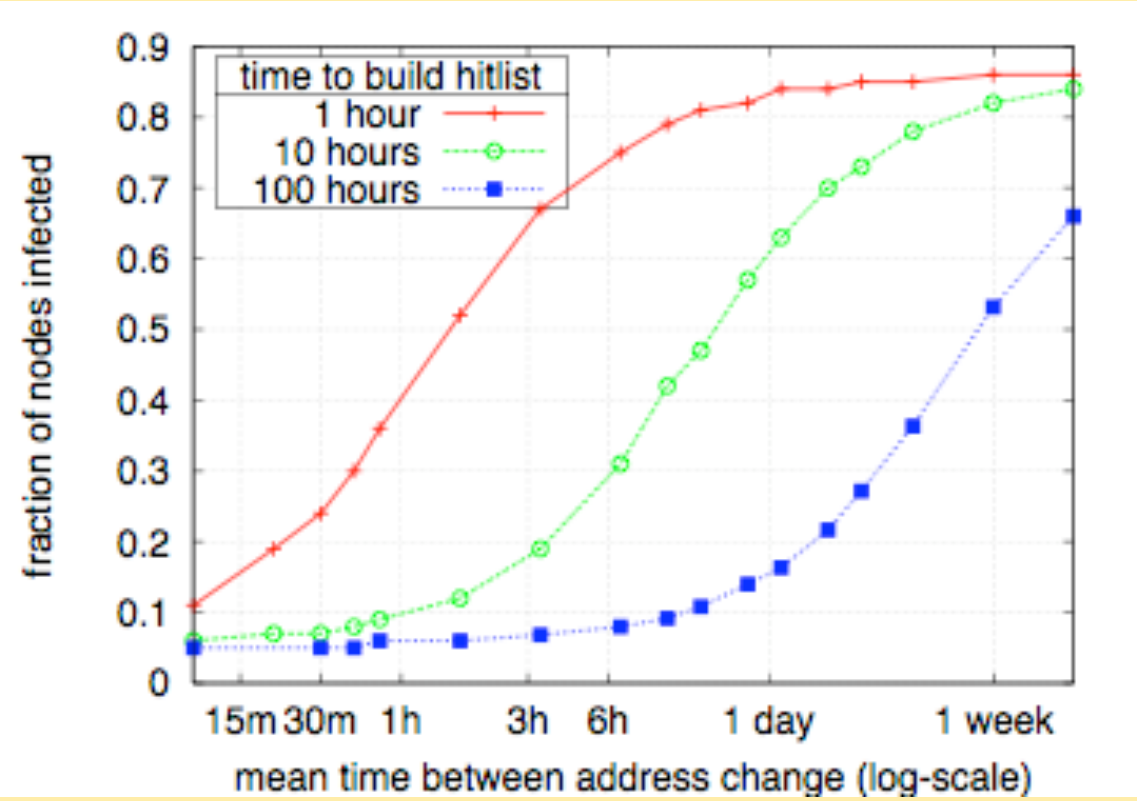
- Higher subnet utilization=> higher success rate for worm when hitting stale entries
- NASR effective even for 90% utilization

Effectiveness vs. vulnerable population density



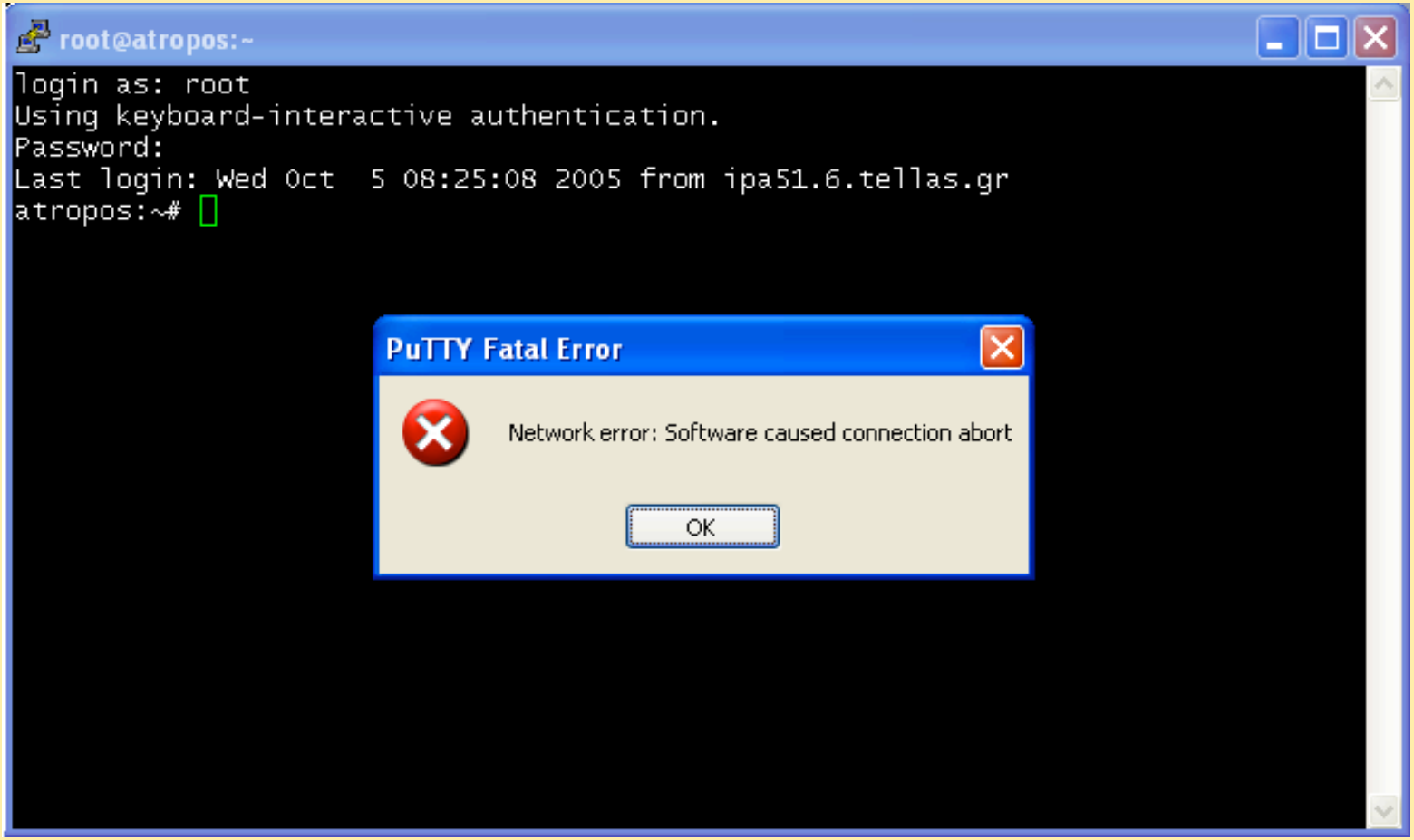
- Impact of NASR is greater for smaller vulnerable populations
- Reason is higher failure rate for stale entries

Interaction with scan-blocking



- When deployed together with scan-blockers, NASR can help contain hitlist worms

Cost of NASR

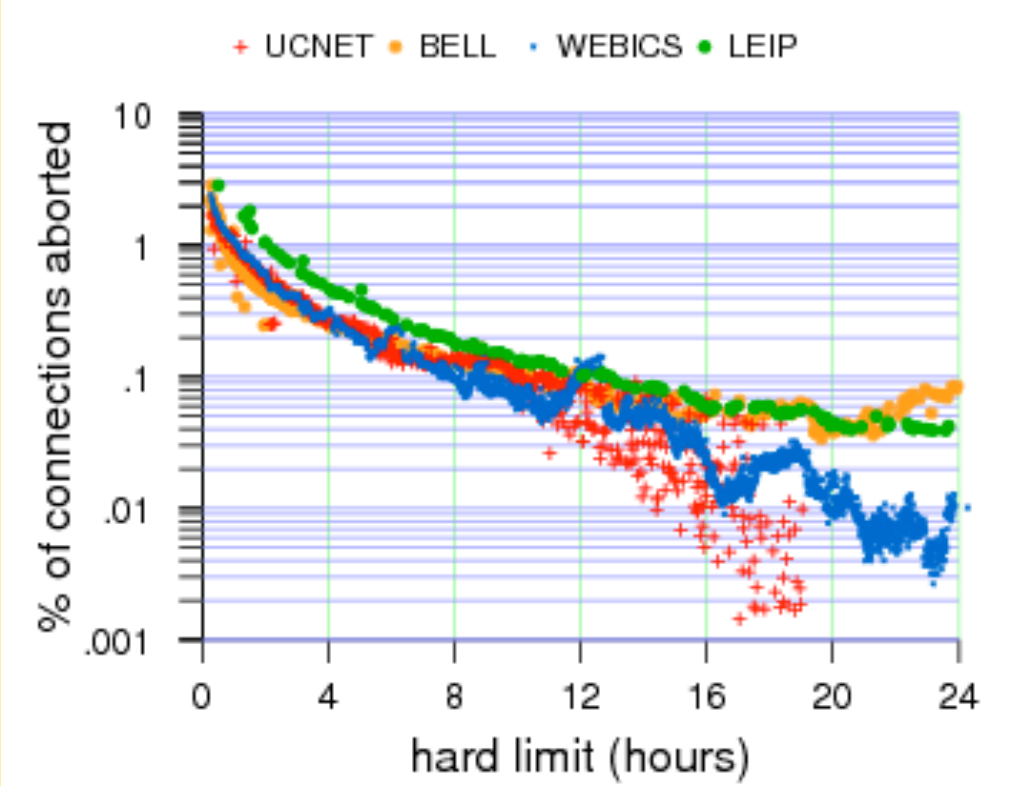


(although many modern applications are designed to be robust to connection failures)

Experiments: cost of NASR

- Used flow traces from four different environments
  - UCNET: local campus network, trace duration 10 hours
  - WEBICS: FORTH web server, 18 days
  - LEIP, University of Leipzig, one week
  - BELL, Bell Labs, one week
- Measured fraction of connections aborted vs. soft and hard limits
  - Offers only an approximation of what really matters e.g., user dissatisfaction, bytes wasted in retransmissions, …

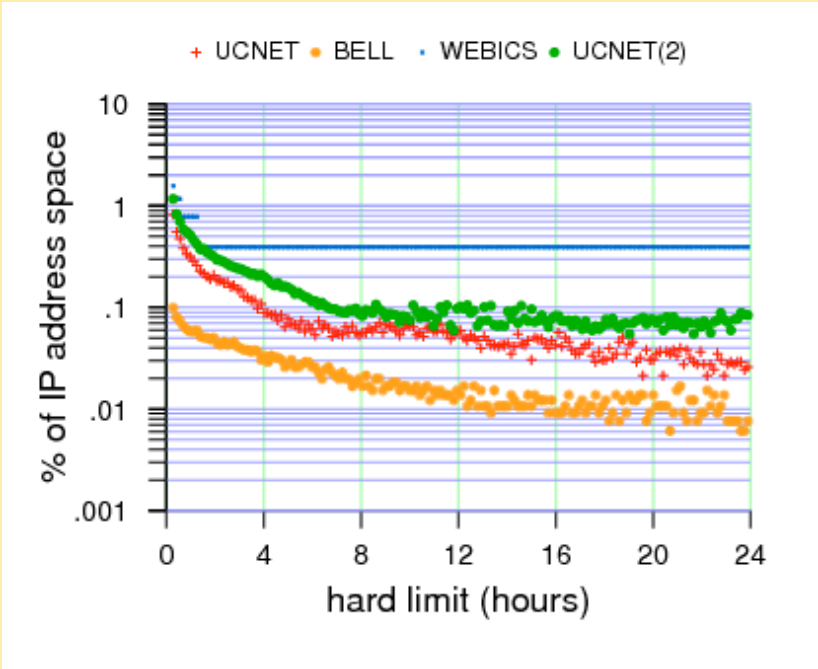
Experiments: cost of NASR

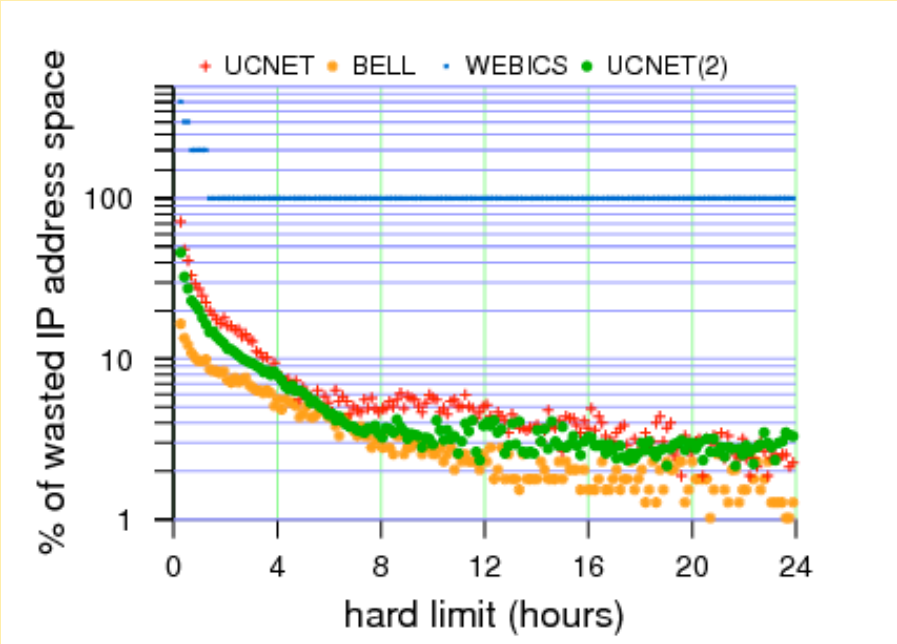


- Cost seems reasonable for hard limit > 2 hours
- Cost is comparable to typical WAN connection failure rates and attack detection FP rates

Experiments: cost of gateway NASR

- We measure address space utilization + fraction of wasted address space





- Cost seems reasonable: ~ 10% of address space wasted even for higher randomization rates

Open Issues

- DNS hitlists
- Hitlist generation times
- Interaction with other defenses
- Likely responses of worm creators

Related Work

- Randomization techniques in OS
  - Similar in principle
- BBN APOD & Sandia DYNAT
  - "IP/port hopping" similar to NASR
  - Key difference is that APOD/DYNAT focus on passive (snooping) adversary, targeted attacks
  - As a result, requires modification to the client

Summary

- We have explored the effectiveness of Network Address Space Randomization (NASR) for defending against hitlist worms
- Mixed results:
  - Advantages: relatively easy to implement & deploy, results in 3x-5x worm slowdown, cost is moderate, may expose worm to detection heuristics
  - Disadvantages: assumes hitlist is generated slowly in stealth, scope limited to IP hitlists