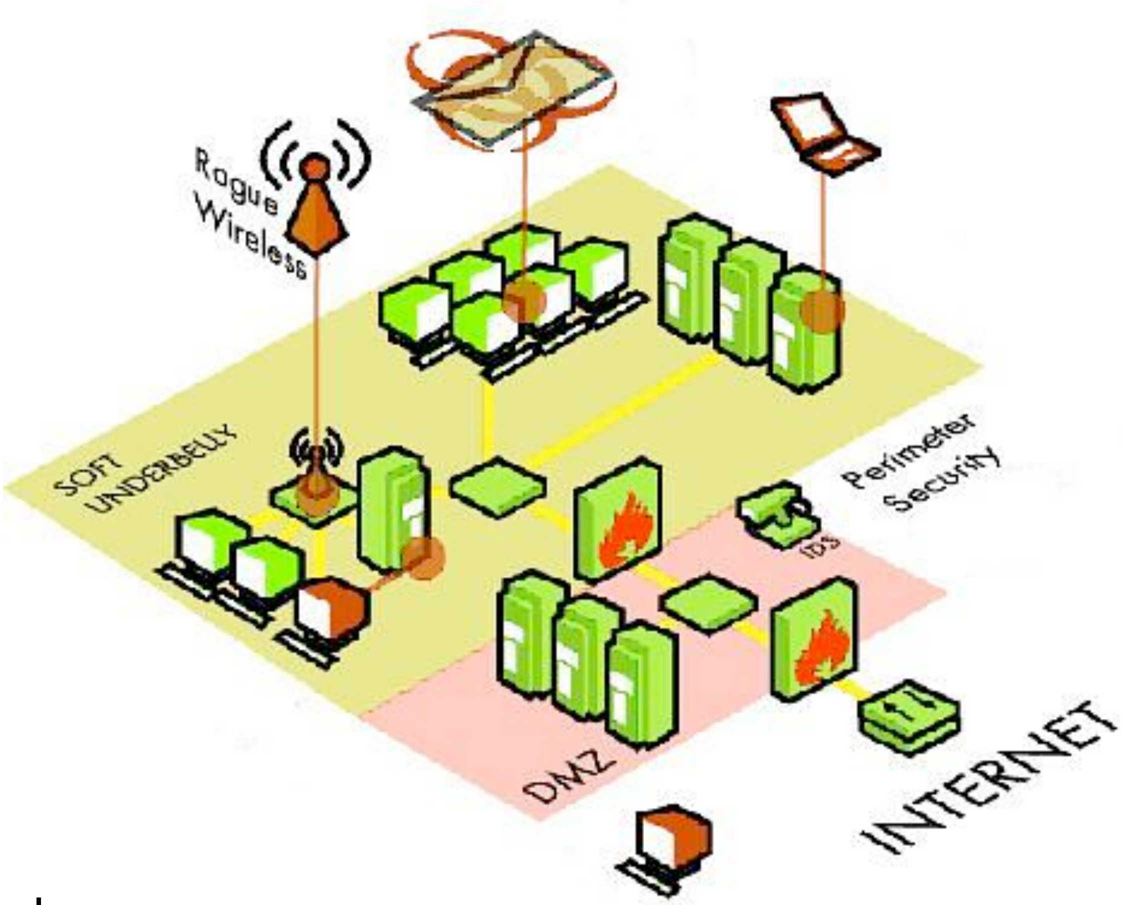
The Evolving Threat of Internet Worms

Jose Nazario, Arbor Networks <jose@arbor.net>

Why Worm Based Intrusions

- Relative ease
  - "Write once, run everywhere" promise can come true
- Penetration
  - Right past firewalls via laptops, find the weakest link
- Persistence
  - Worms keep working so you don't have to
- Coverage
  - Attack everything eventually

Why Worms are Successful



- Missing patches
- Rogue access
- Missing access control

Evolution of Worm Threats

- Previously, worms were simple clones
- Worms have become more complicated systems
  - Multi-vector
    - Grow your potential target base
  - DDoS tool propagation
    - Utilize the army of machines
  - Dynamic
    - Thwart static detection mechanisms
  - Counterworms
    - Fight back with the same strategy

Multi-vector Worms

Goal is to thwart simple defenses and infect more machines

- Code Red vs. Nimda (2001)
  - Code Red: one attack vector (IIS)
  - Nimda: multiple attack vectors (IIS, mail, IE, open shares)
- Sircam (2001)
  - Mass mailer, also spread via open shares
- Blaster (2003)
  - MS-RPC or WebDAV attacks

DDoS Tool Propagation

Use the worm to attack an adversary

- Code Red (2001)
  - SYN flood against a static IP
- Blaster (2003)
  - SYN flood against a static domain
  - Variants carried a DDoS toolkit
- Sapphire, Welchia (2003)
  - The worm's spread is a DDoS

Dynamic Worm Appearances

Try and develop a worm with longevity by evading defenses

- Hybris (2000)
  - Used alt.comp.virus to spread code updates
- Lirva (2003)
  - Attempted to download new packages from website
- Sobig (2003)
  - Contacted website for next set of instructions

Counterworms

Fight the worm with a fast, scalable attack

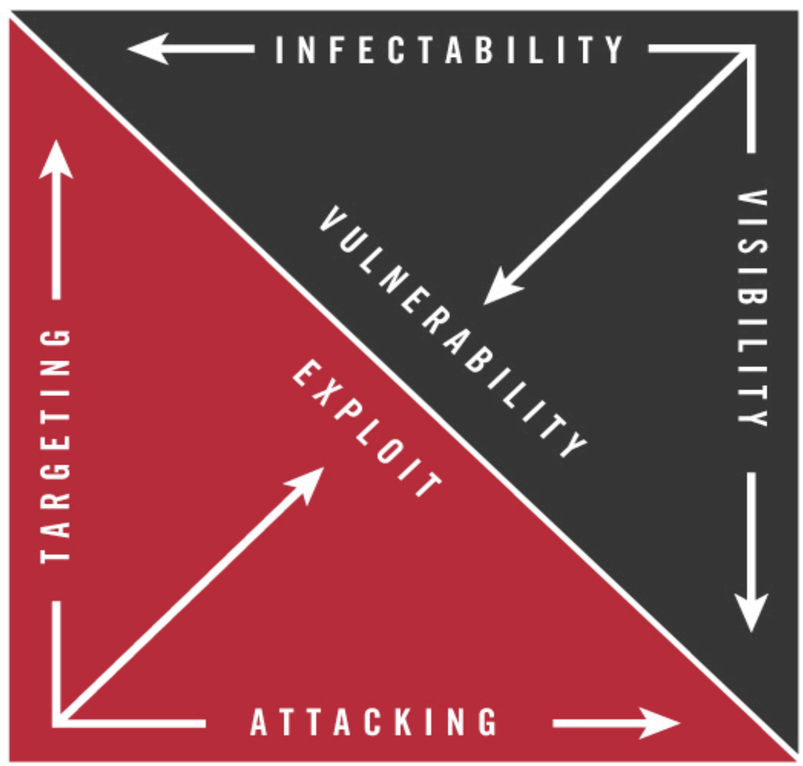
- Code Green (2001)
  - Anti-Code Red worm
- Cheese (2001)
  - Anti-L1on worm
- Welchia (2003)
  - Anti-Blaster worm

Cause more traffic and problems than they attempt to solve

Worm Authors Are Learning

- It's growing easier to build worms
  - Recycle an exploit, automation code, build, launch
- Use flexible targeting for DoS attacks
- No need to target multiple platforms
  - One platform works well enough
- Multiple infection vectors lead to longevity
  - Nimda still present two years later
- Local bias effective at enterprise penetration
  - Worms will be carried into the enterprise
  - Laptops, VPN connections

Vectors of Control



Current Visibility Control

- Classic firewall strategy for the Internet
  - — Minimally protect the DMZ
  - — Maximally protect the internal network
- DMZ for exposed services
  - — Control data flow between trusted, untrusted networks
- Hardened wall against internal, external networks

Classic Vulnerability Control

- Minimized setups on system rollouts
  - Construct an image with minimal software
- Patch maintenance
  - Worms typically attack known holes
- Aggressive known vulnerability inventorying
  - Regular system inventories, comparisons against vulnerability databases (e.g. CVE)

Controlling Infectability

- Hardened systems
  - OS level changes
    - Non-executable stack
    - Permissions for any subsystem
- Hardened applications
  - Application configurations
- Strengthened configurations
  - Services and privileges for any system

Going Beyond the Firewall

- Traditional firewall configuration methods
  - Decide policy, install filters
  - Adjust by reading logs, tweak as needed
  - Broken applications or upset users
- Informed firewall configurations
  - Measure traffic, infer usage
  - Determine policy, install policy
- Assisted by Peakflow X

Intelligent Risk Assessment

- Traditional vulnerability scanners
  - Scan for a service, list machines offering that service
  - Banner grab, report service type, report potential vulnerabilities
- Usage, policy-aware vulnerability scanners
  - Scan for services, compare against usage and policy, report differences
  - Performed by Peakflow X

Combating Worms

- Minimize visibility
  - Tune access filters to a minimal set
  - Externally reachable
  - Internally used
- Minimize vulnerability
  - Track used services
  - Identify, remove unused services
  - Couple to strong patch management

Detecting Worms

- Challenge
  - — In the face of dynamic behaviors, reliably detect the presence of a worm
- Solution
  - — Every worm attempts to spread from host to host
  - — Specific forms of traffic will increase
  - — Not every host will have sent this traffic before
    - Example: web server becoming a web client
- Therefore
  - — Detect the cascading change in host behaviors

Data Gathering for Worm Detection

- Blackhole networks
  - No background traffic
  - Collect attempts from worm trying random hosts
- Live enterprise networks
  - Traffic and relationship modeling
- Live backbone networks
  - Interface and topology statistics
  - Traffic modeling and analysis

Principles of Correlation Analysis

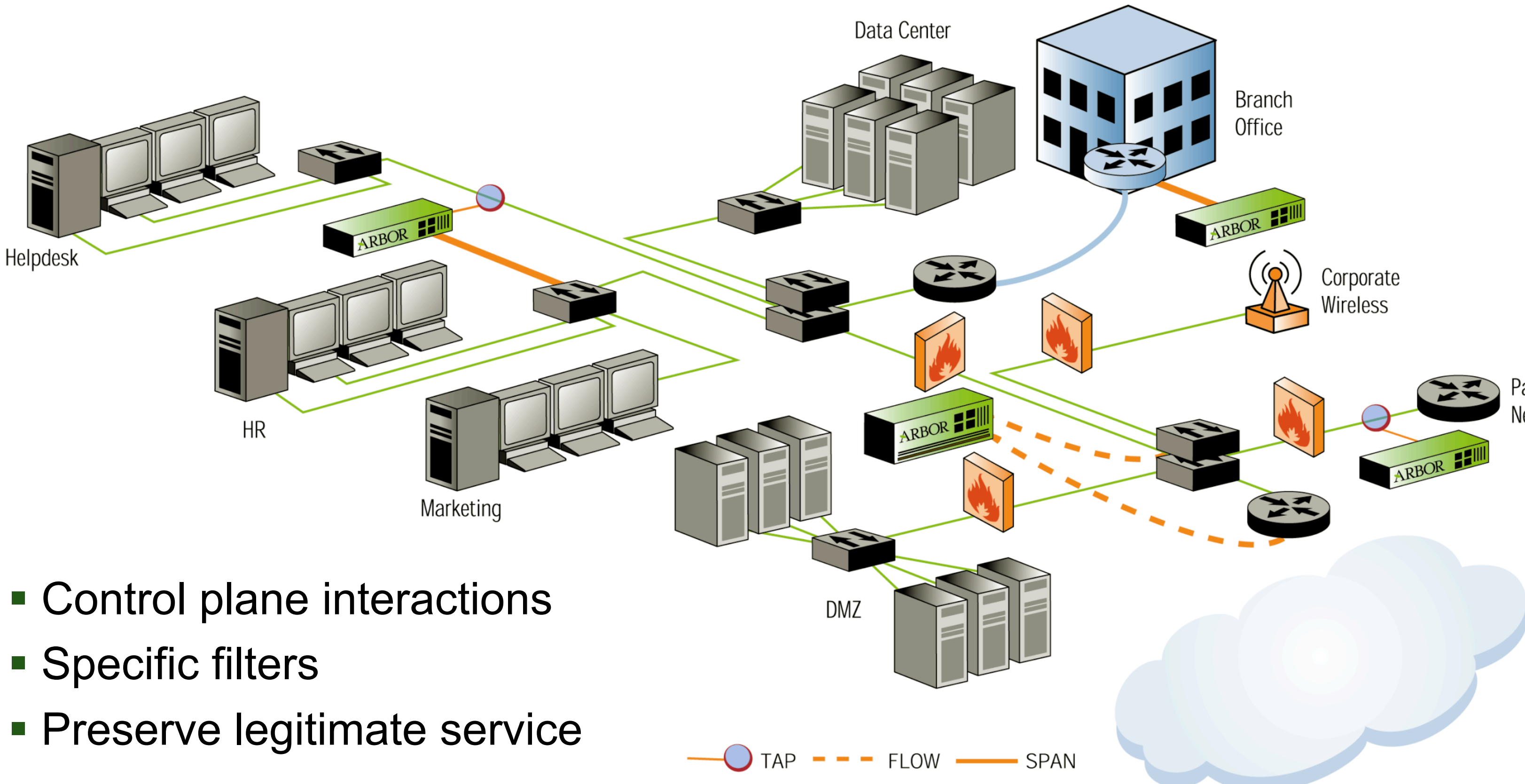
- Two types of correlations to qualify events
  - Auto-correlation
    - Frequency and sources for any single type of anomaly
    - Example: scan frequencies
  - Cross-correlation
    - Frequency and sources of related anomalies
    - Example: scans followed by traffic increases

During worm outbreaks, these frequencies will increase from a growing number of hosts

Worm Detection by Peakflow X

- Uses correlation analysis
  - Partially based on an expert system
  - Extendable by the user via a filter language
- Produces a detailed report
  - Pattern of the worm's behavior
  - Hosts matching this pattern
    - Dynamically grows
  - Amount of traffic caused by the worm
- Couple to flow log for additional forensics

Safe Quarantine Interactions



- Control plane interactions
- Specific filters
- Preserve legitimate service

ARBOR NETWORKS

Conclusions

- Worm authors are getting smarter
  - Worms are getting easier to write, more effective
- Worm detection mechanisms are getting more sophisticated and robust
- IDS and firewall mechanisms are advancing to develop worm defense techniques