Internet Outbreaks: Epidemiology and Defenses

Stefan Savage

Collaborative Center for Internet Epidemiology and Defenses
Department of Computer Science & Engineering
University of California at San Diego

In collaboration with Jay Chen, Cristian Estan, Ranjit Jhala, Erin Kenneally, Justin Ma, David Moore, Vern Paxson (ICSI), Colleen Shannon, Sumeet Singh, Alex Snoeren, Stuart Staniford (Nevis), Amin Vahdat, Erik Vandekeift, George Varghese, Geoff Voelker, Michael Vrable, Nick Weaver (ICSI)

Who am I?

- Assistant Professor, UCSD
  - B.S., Applied History, CMU
  - Ph.D., Computer Science, University of Washington
  - Research at the intersection of networking, security and OS
- Co-founder of Collaborative Center for Internet Epidemiology and Defenses (CCIED)
  - One of four NSF Cybertrust Centers, joint UCSD/ICSI effort
  - Focused on large-scale Internet attacks (worms, viruses, botnets, etc)
- Co-founded a number of commercial security startups
  - Asta Networks (failed anti-DDoS startup)

  - Netsift Inc, (successful anti-worm/virus startup)

A Chicken Little view of the Internet…

Why Chicken Little is a naïve optimist

- Imagine the following species:
  - Poor genetic diversity; heavily inbred
  - Lives in “hot zone”; thriving ecosystem of infectious pathogens
  - Instantaneous transmission of disease
  - Immune response 10-1M times slower
  - Poor hygiene practices
- **What would its long-term prognosis be?**

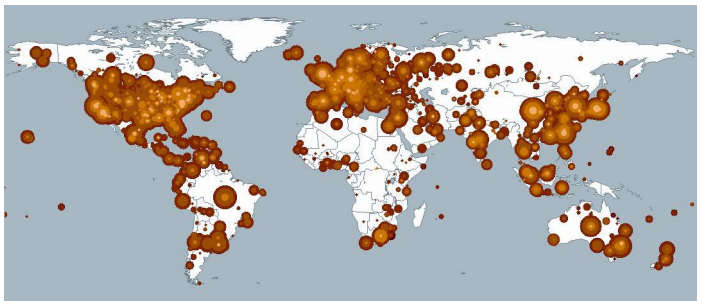
Why Chicken Little is a naïve optimist

- Imagine the following species:
  - Poor genetic diversity; heavily inbred
  - Lives in “hot zone”; thriving ecosystem of infectious pathogens
  - Instantaneous transmission of disease
  - Immune response 10-1M times slower
  - Poor hygiene practices
- **What would its long-term prognosis be?**
- What if diseases were designed…
  - Trivial to create a *new* disease
  - Highly profitable to do so

Threat transformation

- **Traditional threats**
  - Attacker manually targets high-value system/resource
  - Defender increases cost to compromise high-value systems
  - Biggest threat: insider attacker

- **Modern threats**
  - Attacker uses automation to target **all** systems at once (can filter later)
  - Defender must defend **all** systems at once
  - Biggest threats: software vulnerabilities & naïve users



Large-scale technical enablers

- ***Unrestricted connectivity***
  - Large-scale adoption of IP model for networks & apps
- ***Software homogeneity & user naiveté***
  - Single bug = mass vulnerability in millions of hosts
  - Trusting users (“ok”) = mass vulnerability in millions of hosts
- **Few meaningful defenses**
- **Effective anonymity (minimal risk)**

Driving economic forces

- No longer just for fun, but for profit
  - SPAM forwarding (MyDoom.A backdoor, SoBig), Credit Card theft (Korgo), DDoS extortion, etc…
  - Symbiotic relationship: worms, bots, SPAM, DDoS, etc
  - Fluid third-party exchange market (**millions** of hosts for sale)
    - Going rate for SPAM proxying 3 -10 cents/host/week
      - Seems small, but 25k botnet gets you $40k-130k/yr
    - Raw bots, 1$+/host, Special orders ($50+)
- "Virtuous" economic cycle
- Bottom line:

  Large numbers of compromised hosts = ***platform***
  DDoS, SPAM, piracy, identity theft = **applications**

What service-oriented computing really means…

Subject: **I offer the DDoS attack service !**
From: ddos@safe-mail.net <DDOS Service>
Date: 3/3/05 10:54
Newsgroups: alt.2600.cardz

HI,

I offer the DDOS attack service, I offer estimate of expense on hour base. Free demonstration (10 minutes).
The price is based on the difficulty to pull down the target website, for the free demonstration or information please contact :

DDOS Service at : ddos@safe-mail.ne

Today's focus: Outbreaks

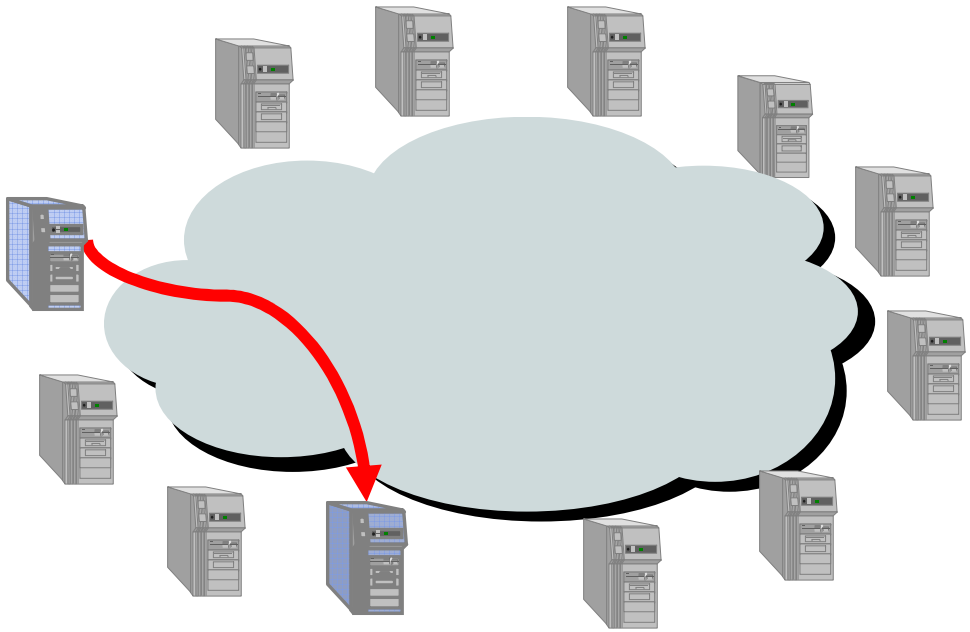
- Outbreaks?
  - Acute epidemics of infectious malcode designed to actively *spread* from host to host over the network
  - E.g. Worms, viruses, etc (I don't care about pedantic distinctions, so I'll use the term worm from now on)
- Why epidemics?
  - Epidemic spreading is the fastest method for large-scale network compromise
- Why fast?
  - Slow infections allow much more time for detection, analysis, etc (traditional methods may cope)

Today

- Network worm review
- Network epidemiology
- Threat monitors & automated defenses

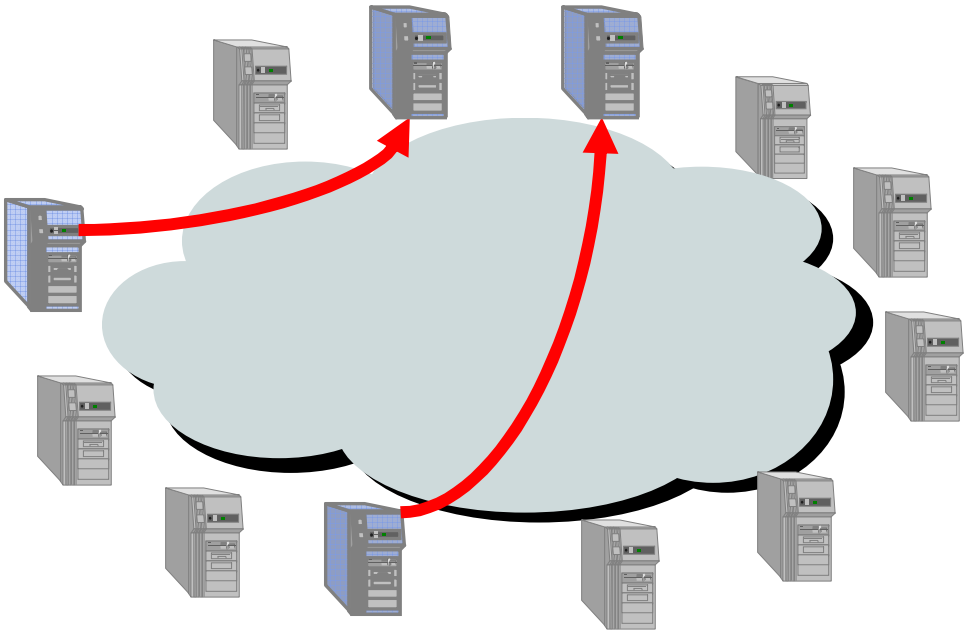
What is a network worm?

- Self-propagating self-replicating network program
  - Exploits some vulnerability to infect remote machines
  - Infected machines continue propagating infection



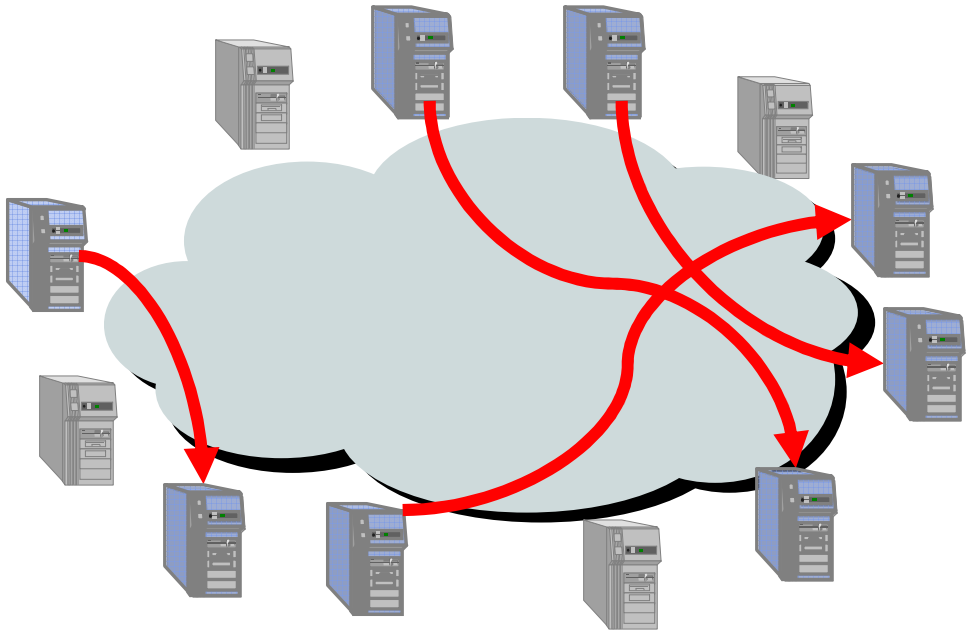
What is a network worm?

- Self-propagating self-replicating network program
  - Exploits some vulnerability to infect remote machines
  - Infected machines continue propagating infection



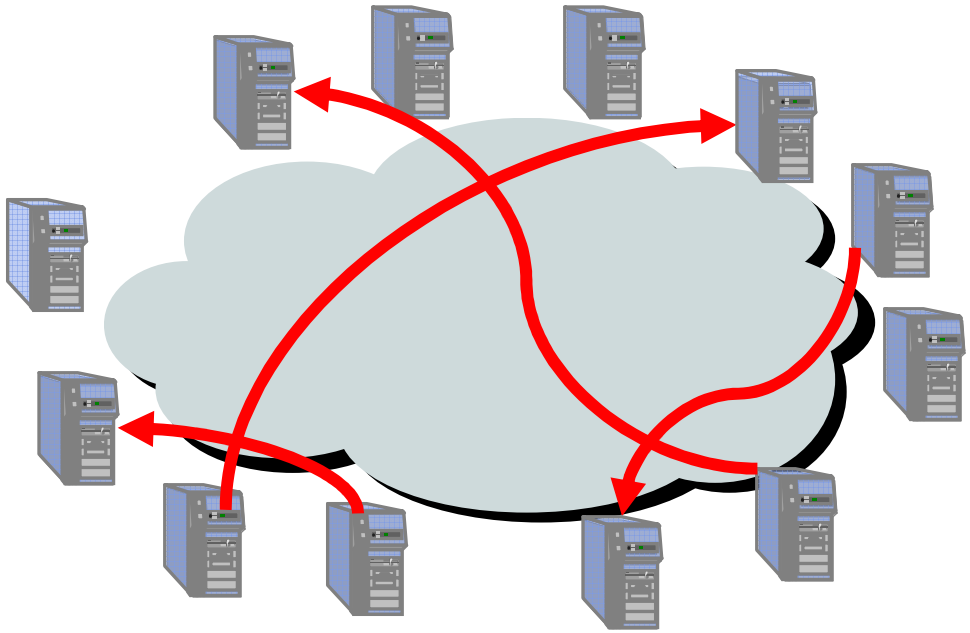
What is a network worm?

- Self-propagating self-replicating network program
  - Exploits some vulnerability to infect remote machines
  - Infected machines continue propagating infection

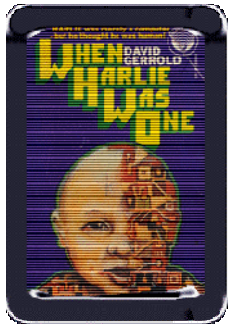


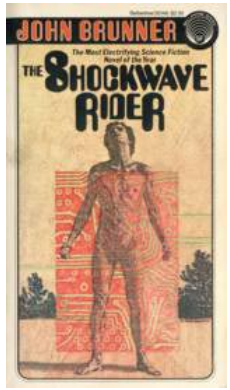
What is a network worm?

- Self-propagating self-replicating network program
  - Exploits some vulnerability to infect remote machines
  - Infected machines continue propagating infection



A brief history of worms…





- As always, Sci-Fi authors get it first
  - Gerold's "*When H.A.R.L.I.E. was One*" (1972) – "Virus"
  - Brunner's "Shockwave Rider" (1975) – "tapeworm program"
- Shoch&Hupp co-opt idea; coin term "worm" (1982)
  - Key idea: programs that self-propagate through network to accomplish some task; benign
- Fred Cohen demonstrates power and threat of self-replicating viruses (1984)
- Morris worm exploits buffer overflow vulnerabilities & infects a few thousand hosts (1988)

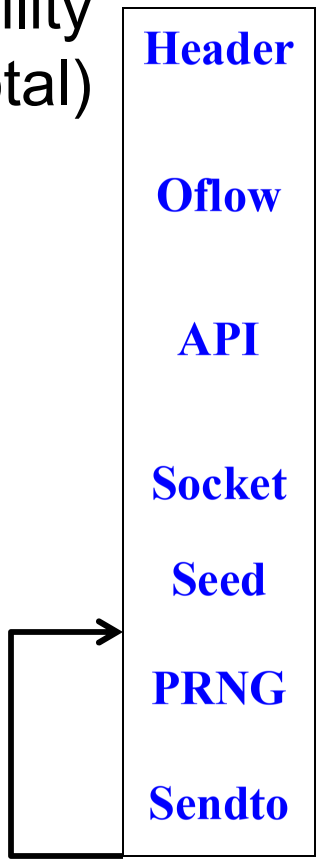
Hiatus for over a decade…

The Modern Worm era

- Email based worms in late 90's (Melissa & ILoveYou)
  - Infected >1M hosts, but requires user participation
- **CodeRed** worm released in Summer 2001
  - Exploited buffer overflow in IIS; no user interaction
  - Uniform random target selection (after fixed bug in CRv1)
  - Infects 360,000 hosts in 10 hours (CRv2)
  - Attempted to mount simultaneous DDoS attack on whitehouse.gov
  - Like the energizer bunny… still going
- Energizes **renaissance** in worm construction (1000's)
  - Exploit-based: CRII, Nimda, **Slammer**, Blaster, Witty, etc…
  - Human-assisted: SoBig, NetSky, MyDoom, etc…
  - *6200 malcode variants* in 2004; 6x increase from 2003 [Symantec]

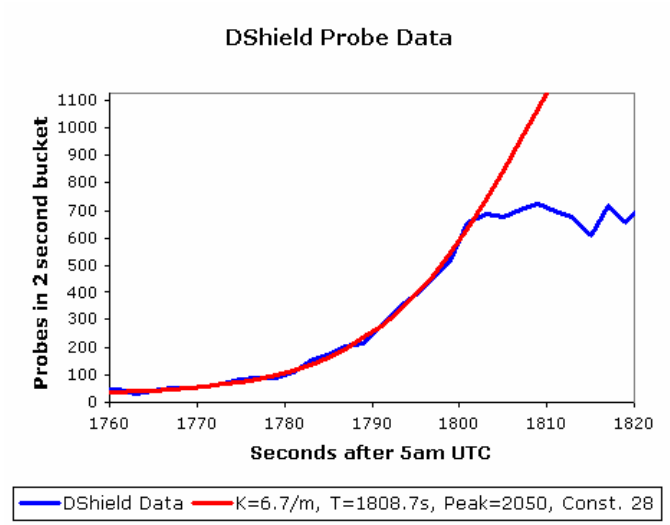
Anatomy of a worm: Slammer

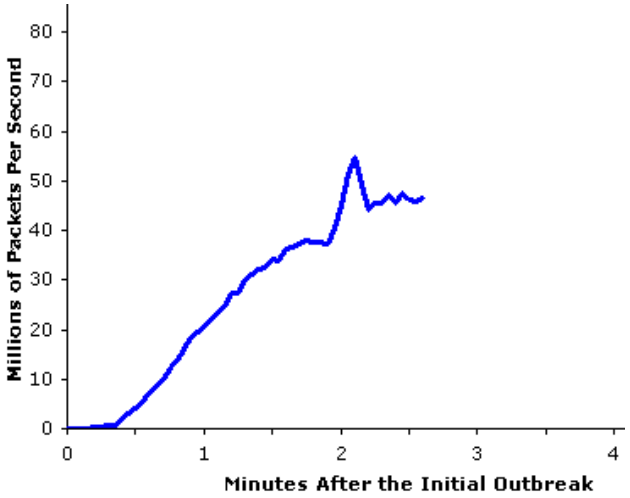
- Exploited SQL server buffer overflow vulnerability
- Worm fit in a single UDP packet (404 bytes total)
- Code structure
  - Cleanup from buffer overflow
  - Get API pointers
    - Code borrowed from published exploit
  - Create socket & packet
  - Seed PRNG with `getTickCount()`
  - While (TRUE)
    - Increment Pseudo-RNG
      - Mildly buggy
    - Send packet to pseudo-random address
- Main advancement: **doesn't listen** (decouples scanning from target behavior)



A pretty fast outbreak: Slammer (2003)

- First ~1min behaves like classic random scanning worm
  - Doubling time of ~8.5 seconds
  - CodeRed doubled every 40mins
- >1min worm starts to saturate access bandwidth
  - Some hosts issue >20,000 scans per second
  - Self-interfering (no congestion control)
- Peaks at ~3min
  - **>55million IP scans/sec**
- **90% of Internet scanned in <10mins**
  - Infected ~100k hosts (conservative)





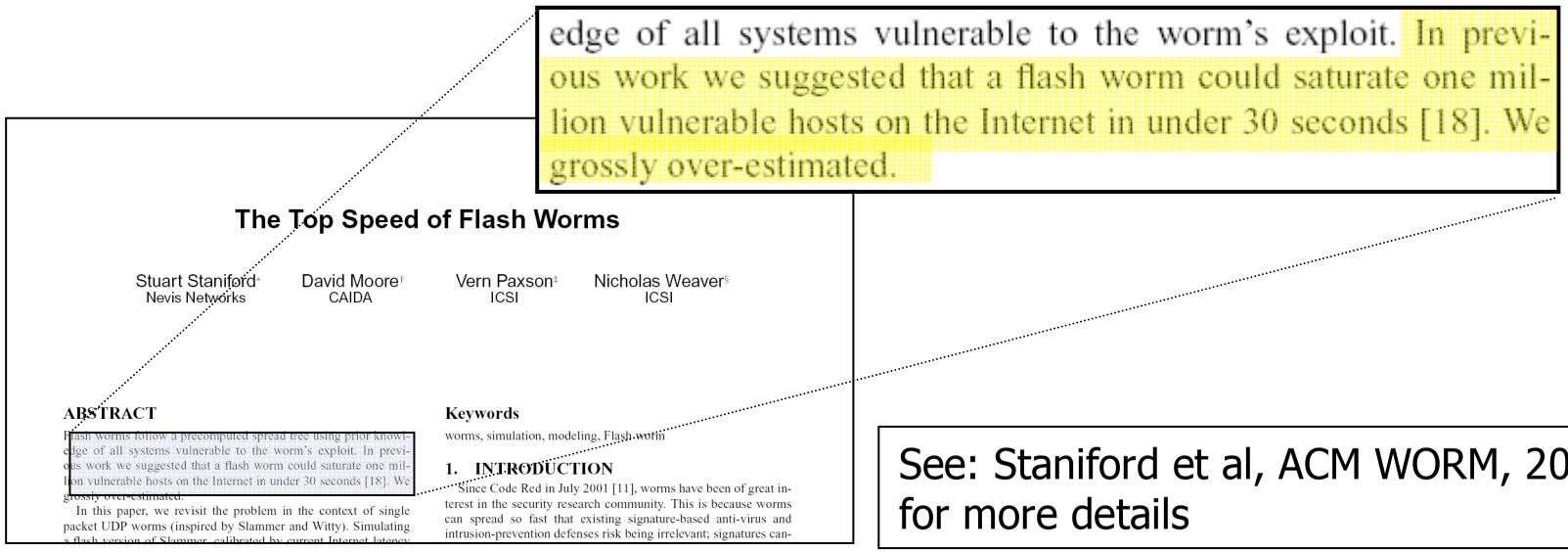
See: Moore et al, IEEE Security & Privacy, 1(4), 2003 for more details

Was Slammer really fast?

- **Yes**, it was orders of magnitude faster than CR
- **No**, it was poorly written and unsophisticated

Was Slammer really fast?

- **Yes**, it was orders of magnitude faster than CR
- **No**, it was poorly written and unsophisticated
- **Who cares?** It is *literally* an academic point
  - The current debate is whether one can get < 500ms
  - **Bottom line**: way faster than people!



edge of all systems vulnerable to the worm's exploit. In previous work we suggested that a flash worm could saturate one million vulnerable hosts on the Internet in under 30 seconds [18]. We grossly over-estimated.

The Top Speed of Flash Worms

Stuart Staniford* Nevis Networks
David Moore† CAIDA
Vern Paxson‡ ICSI
Nicholas Weaver§ ICSI

ABSTRACT

[illegible] worms follow a precomputed spread tree using prior knowledge of all systems vulnerable to the worm's exploit. In previous work we suggested that a flash worm could saturate one million vulnerable hosts on the Internet in under 30 seconds [18]. We grossly over-estimated.

In this paper, we revisit the problem in the context of single packet UDP worms (inspired by Slammer and Witty). Simulating a flash version of Slammer, calibrated by current Internet latency

Keywords

worms, simulation, modeling, Flash worm

1. INTRODUCTION

Since Code Red in July 2001 [11], worms have been of great interest in the security research community. This is because worms can spread so fast that existing signature-based anti-virus and intrusion-prevention defenses risk being irrelevant; signatures can-

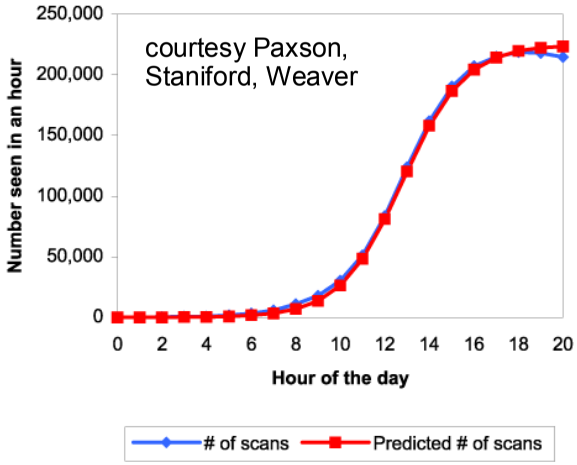
See: Staniford et al, ACM WORM, 2004 for more details

How to think about worms

- Reasonably well described as infectious epidemics
  - Simplest model: Homogeneous random contacts
- Classic SI model
  - N: population size
  - S(t): susceptible hosts at time t
  - I(t): infected hosts at time t
  - ß: contact rate
  - i(t): I(t)/N, s(t): S(t)/N

$$\frac{dI}{dt} = \beta \frac{IS}{N} \qquad \frac{dS}{dt} = -\beta \frac{IS}{N} \quad \rightarrow \quad \frac{di}{dt} = \beta i(1-i)$$

$$i(t) = \frac{e^{\beta(t-T)}}{1+e^{\beta(t-T)}}$$



What's important?

- There are lots of improvements to this model…
  - Chen et al, *Modeling the Spread of Active Worms*, Infocom 2003 (discrete time)
  - Wang et al, *Modeling Timing Parameters for Virus Propagation on the Internet* , ACM WORM '04 (delay)
  - Ganesh et al, *The Effect of Network Topology on the Spread of Epidemics*, Infocom 2005 (topology)
- … but the conclusion is the same. We care about two things:

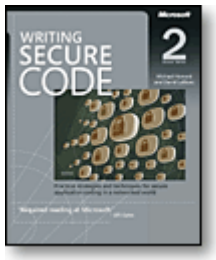
- How **likely** is it that a given infection attempt is successful?
  - Target selection (random, biased, hitlist, topological,…)
  - Vulnerability distribution (e.g. density – S(0)/N)

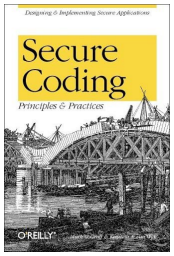
- How **frequently** are infections attempted?
  - ß: Contact rate

What can be done?

- Reduce the number of susceptible hosts
  - **Prevention**, reduce S(t) while I(t) is still small (ideally reduce S(0))
- Reduce the contact rate
  - **Containment**, reduce ß while I(t) is still small
- Reduce the number of infected hosts
  - **Treatment**, reduce I(t) after the fact

Prevention: Software Quality





- **Goal:** eliminate vulnerability

- Static/dynamic testing (e.g. Cowan, Wagner, Engler, etc)
- Software process, code review, etc.

- **Active** research community
- Taken seriously in industry
  - Security code review *alone* for Windows Server 2003 ~ $200M

coverity

FORTIFY
_security->starts here

- Traditional problems: soundness, completeness, usability
- Practical problems: scale and cost

Prevention: Wrappers

- Goal: stop vulnerability from being exploited
- Hardware/software buffer overflow prevention
  - NX, /GS, StackGuard, etc
- Sandboxing (BSD Jail, GreenBorders)
  - Limit capabilities of potentially exploited program

Prevention: Software Heterogeneity

- **Goal:** reduce impact of vulnerability

- Use software diversity to tolerate attack
  - Exploit *existing* heterogeneity
    - Junqueria et al, *Surviving Internet Catastrophes*, USENIX '05
    - Haeberlen et al, *Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures*, NSDI '05
  - Create *artificial* heterogeneity (hot research topic)
    - Forrest et al, *Building Diverse Computer Systems*, HotOS '97
    - Large contemporary literature (address randomization, execution polymorphism)

- Open questions: class of vulnerabilities that can be masked, strength of protection, cost of support

Prevention: Software Updating

- **Goal:** reduce window of vulnerability

- Most worms exploit known vulnerability (1 day -> 3 months)
  - Window shrinking: automated patch->exploit
  - Patch deployment challenges, downtime, **Q/A**, etc
  - Rescorla, *Is finding security holes a good idea?*, WEIS '04

- Network-based filtering: decouple "patch" from code
  - E.g. TCP packet to port 1434 and > 60 bytes
  - Wang et al, *Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits*, SIGCOMM '04
  - Symantec: Generic Exploit Blocking

Prevention: Known Exploit Blocking

- Get early samples of new exploit
  - Network sensors/honeypots
  - “Zoo” samples
- Anti-virus/IPS company distills “signature”
  - Labor intensive process
- Signature pushed out to all customers

Assumes long reaction window

- Host recognizer checks files/memory before execution
  - Much more than grep… polymorphism/metamorphism

- Example: Symantec
  - Gets early intelligence via managed service side of business and DeepSight sensor system
  - >60TB of signature updates per day

Prevention: Hygiene Enforcement

- **Goal**: keep susceptible hosts off network
- Only let hosts connect to network if they are "well cared for"
  - Recently patched, up-to-date anti-virus, etc…
  - Manual version in place at some organizations (e.g. NSF)
- Cisco Network Admission Control (NAC)

Containment

- Reduce contact rate
- **Slow down**
  - Throttle connection rate to slow spread
    - Twycross & Williamson, *Implementing and Testing a Virus Throttle*, USENIX Sec '03
    - Version used in some HP switches
  - Important capability, but worm still spreads…
- **Quarantine**
  - Detect and block worm
  - Rest of talk…

Treatment

- Reduce I(t) after the outbreak is done
  - Practically speaking this is where much happens because our defenses are so bad

- **Two issues**
  - How to detect infected hosts?
    - They still spew traffic (commonly true, but poor assumption)
      - Ma et al, "*Self-stopping Worms*", WORM '05
    - Look for known signature (malware detector)
  - What to do with infected hosts?
    - Wipe whole machine
    - Custom disinfector (need to be sure you get it all…backdoors)
    - Aside: interaction with SB1386…

Quarantine requirements

- We can define reactive defenses in terms of:
  - ***Reaction time*** – **how long** to detect, propagate information, and activate response
  - ***Containment strategy*** – **how** malicious behavior is identified and stopped
  - ***Deployment scenario*** - **who** participates in the system
- Given these, what are the engineering requirements for **any** effective defense?

Its difficult…

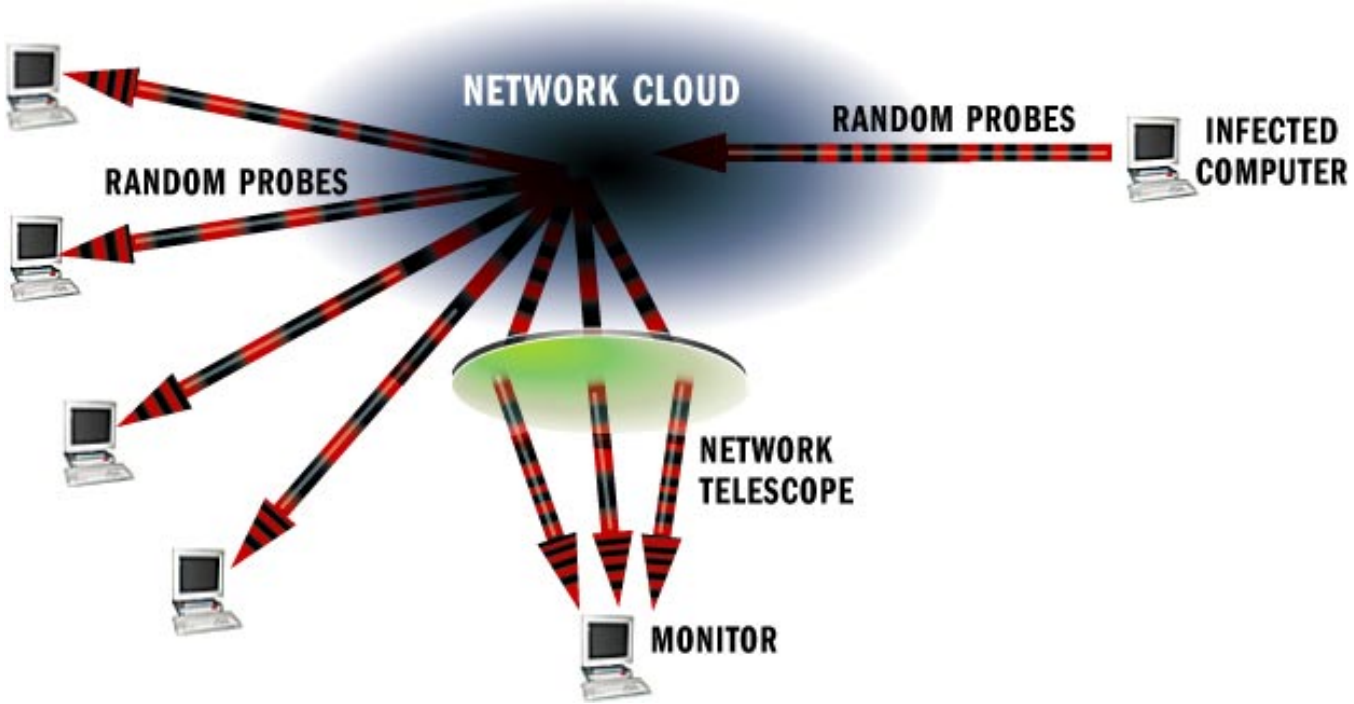
- Even with universal defense deployment, containing a CodeRed-style worm (<10% in 24 hours) is tough
  - Address filtering (blacklists), must respond < 25mins
  - Content filtering (signatures), must respond < 3hrs
- For faster worms (e.g. Slammer), seconds
- For non-universal deployment, life is worse…

See: Moore et al, *Internet Quarantine: Requirements for Containing Self-Propagating Code*, Infocom 2003 for more details

How do we detect new outbreaks?

- Threat monitors
  - Network-based
    - Ease of deployment, significant coverage
    - Inter-host correlation
    - Scalability challenges (performance)
  - Endpoint-based
    - Host offers high-fidelity vantage point (execution vs lexical domain)
    - Scalability challenges (deployment)
- Monitoring environments
  - In-situ: real activity as it happens
    - Network/host IDS
  - Ex-situ: “canary in the coal mine”
    - HoneyNets/Honeypots

Network Telescopes



- Infected host scans for other vulnerable hosts by randomly generating IP addresses
- Network Telescope: monitor large range of unused IP addresses – will receive scans from infected host
- Very scalable. UCSD monitors 17M+ addresses

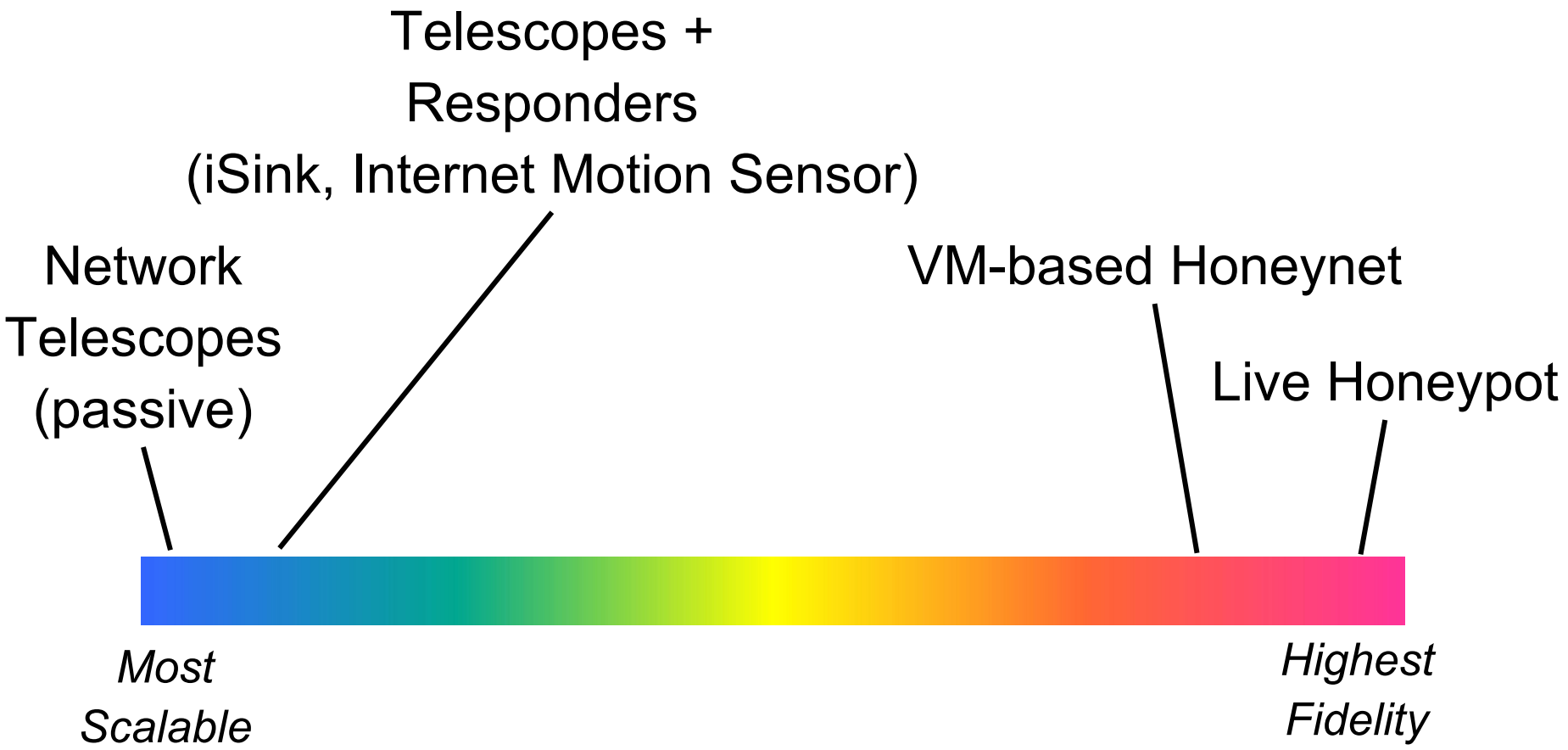
Telescopes + Active Responders

- Problem: Telescopes are passive, can't respond to TCP handshake
  - Is a SYN from a host infected by CodeRed or Welchia? Dunno.
  - What does the worm payload look like? Dunno.
- Solution: proxy responder
  - Stateless: TCP SYN/ACK (Internet Motion Sensor), per-protocol responders (iSink)
  - Stateful: Honeyd
  - Can differentiate and fingerprint payload
    - False positives generally low since no regular traffic

Honeypots

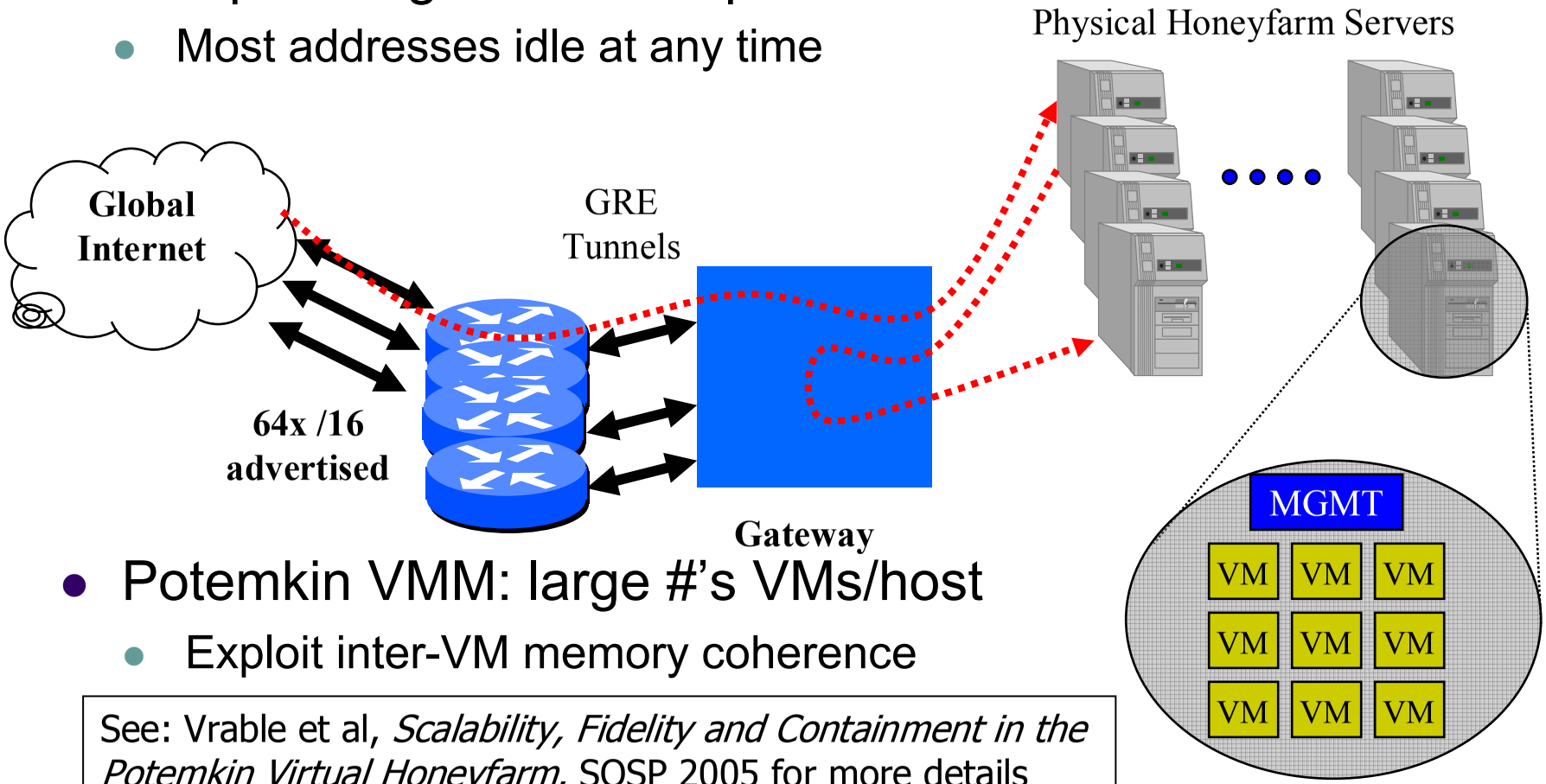
- Problem: don't know what worm/virus would do? No code ever executes after all.
- Solution: deploy real "infectable" hosts (honeypots)
  - Individual hosts or VM-based: Collapsar, HoneyStat, Symantec
  - Generate signatures for new malware… either at network level (honeycomb) or over execution (Vigalante, DACODA, Sting)
  - Low false-positive rate (no one should be here)
- Challenges
  - Scalability ($$$)
  - Liability (grey legal territory)
  - Isolation (warfare between malware)
  - Detection (VMWare detection code in the wild)

The Scalability/Fidelity tradeoff



Potemkin honeyfarm: large scale high-fidelity honeyfarm

- Goal: emulate significant fraction of Internet hosts (1M)
- Multiplex large address space on smaller # of servers
  - Most addresses idle at any time



- Potemkin VMM: large #'s VMs/host
  - Exploit inter-VM memory coherence

See: Vrable et al, *Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm*, SOSP 2005 for more details

Containment

- Key issue: 3rd party liability and contributory damages
  - Honeyfarm = worm accelerator
  - Worse I knowingly allowed my hosts to be infected (premeditated negligence, outside “best practices” safe harbor)
- Export policy tradeoffs between risk and fidelity
  - Block all outbound packets: no TCP connections
  - Only allow outbound packets to host that previously send packet: no outbound DNS, no botnet updates
  - Allow outbound, but “scrub”: is this a best practice?
  - In the end, need fairly flexible policy capabilities
    - Could do whole talk on interaction between technical & legal drivers

Challenges for honeypot systems

- **Depend** on worms trying to infect them
  - What if they don't scan those addresses (smart bias)
  - What if they propagate via e-mail, IM? (doable, but privacy issues)
- Inherent tradeoff between liability exposure and detectability
  - Honeypot detection software exists… perfect virtualization tough
- It doesn't necessary reflect what's happening on **your** network (can't count on it for local protection)

- Hence, there is also a need for approaches that monitor "real" systems (typically via the network)

Scan Detection

- Idea: detect infected hosts via infection attempts

- Indirect scan detection
  - Wong et al, *A Study of Mass-mailing Worms*, WORM '04
  - Whyte et al. *DNS-based Detection of Scanning Worms in an Enterprise Network,* NDSS '05
- Direct scan detection
  - Weaver et al. *Very Fast Containment of Scanning Worms*, USENIX Sec '04
    - Threshold Random Walk – bias source based on connection success rate (Jung et al); Venkataraman et al, *New Streaming Algorithms for Fast Detection of Superspreaders*, NDSS '05

- Can be used inbound (protect self) or outbound (protect others)

Signature Inference

- Monitor network and **learn** signature for new worms in < 1sec
- Signatures can then be used for content filtering

PACKET HEADER

```
SRC: 11.12.13.14.3920 DST: 132.239.13.24.5000 PROT: TCP
```

PACKET PAYLOAD (CONTENT)

```
00F0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 4D 3F E3 77 ............M?.w
0110  90 90 90 90 FF 63 64 90 90 90 90 90 90 90 90 90 .....cd.........
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0130  90 90 90 90 90 90 90 90 EB 10 5A 4A 33 C9 66 B9 ..........ZJ3.f.
0140  66 01 80 34 0A 99 E2 FA EB 05 E8 EB FF FF FF 70 f..4...........p
...
```

Approach

- Monitor network and **learn** signature for new worms in < 1sec
- Signatures can then be used for content filtering

PACKET HEADER

```
SRC: 11.12.13.14.3920 DST: 132.239.13.24.5000 PROT: TCP
```

PACKET PAYLOAD (CONTENT)

```
00F0  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0100  90 90 90 90 90 90 90 90 90 90 90 90 4D 3F E3 77 ............M?.w
0110  90 90 90 90 FF 63 64 90 90 90 90 90 90 90 90 90 .....cd.........
0120  90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ................
0130  90 90 90 90 90 90 90 90 EB 10 5A 4A 33 C9 66 B9 ..........ZJ3.f.
0140  66 01 80 34 0A 99 E2 FA EB 05 E8 EB FF FF FF 70 f..4...........p
...
```

Content sifting

- Assume there exists some (relatively) unique invariant bitstring *W* across all instances of a particular worm

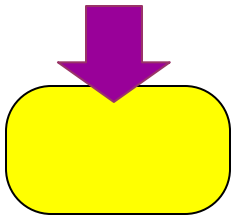
- Two consequences
  - **Content Prevalence**: *W* will be more common in traffic than other bitstrings of the same length
  - **Address Dispersion**: the set of packets containing *W* will address a disproportionate number of distinct sources and destinations

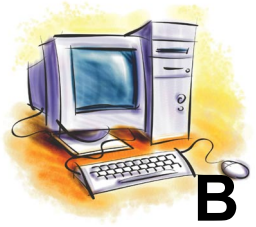
- *Content sifting*: find *W*'s with high content prevalence and high address dispersion and drop that traffic

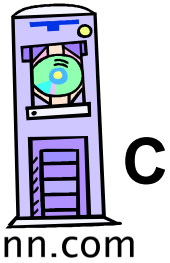
See: Singh et al, *Automated Worm Fingerprinting,* OSDI 2004 for more details

The basic algorithm

Detector in network





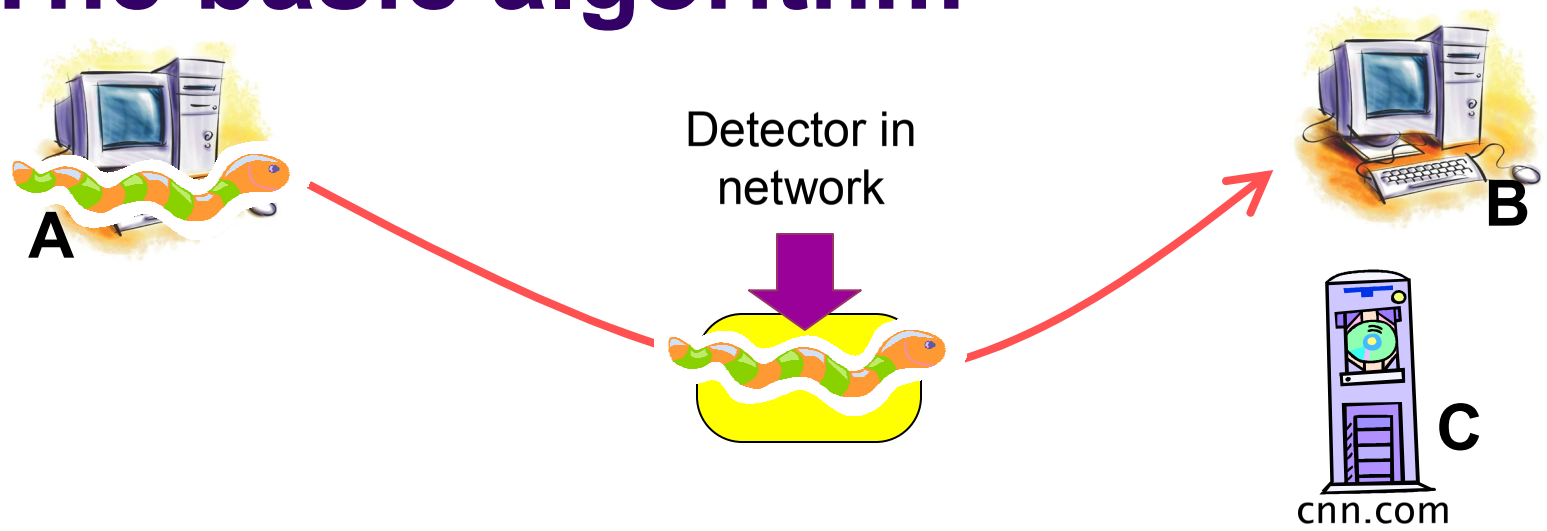


Prevalence Table

Address **Dispersion** Table

Sources	Destinations

The basic algorithm



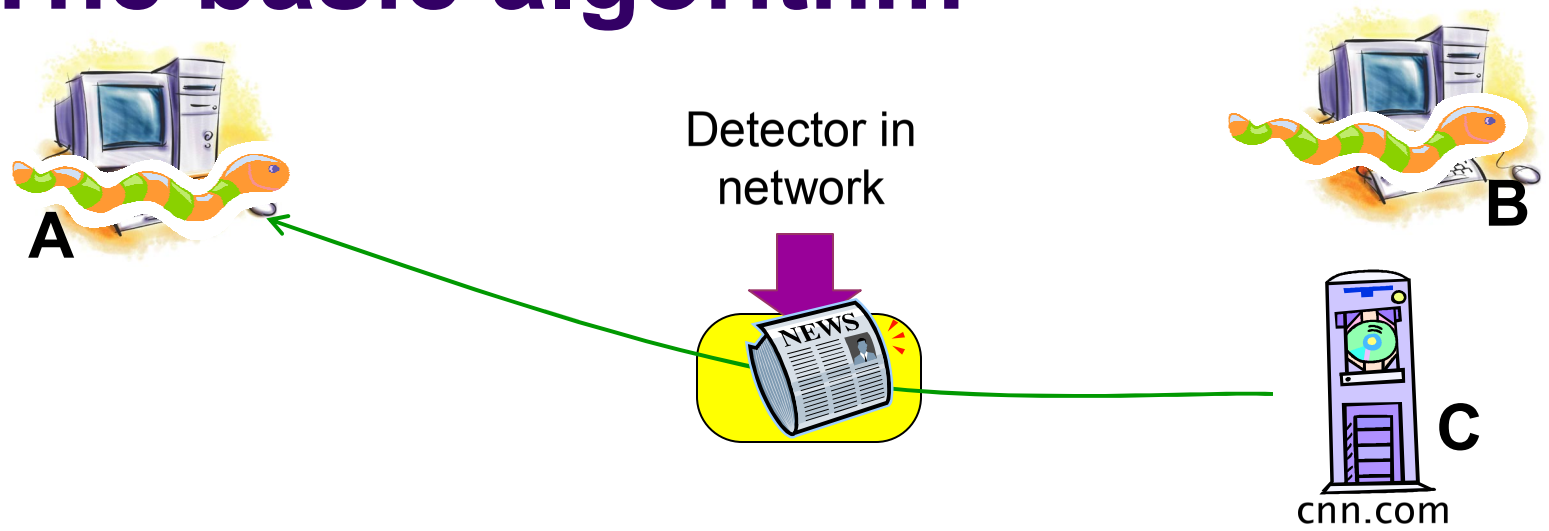
Prevalence Table

	1

Address **Dispersion** Table

Sources	Destinations
1 (A)	1 (B)

The basic algorithm



Prevalence Table

	**1**
	**1**

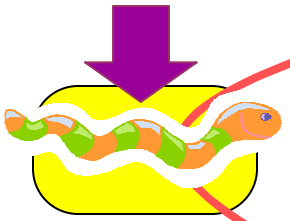
Address **Dispersion** Table

Sources	Destinations
1 (A)	**1 (B)**
1 (C)	**1 (A)**

The basic algorithm

A

Detector in network



B

C

cnn.com

E

D

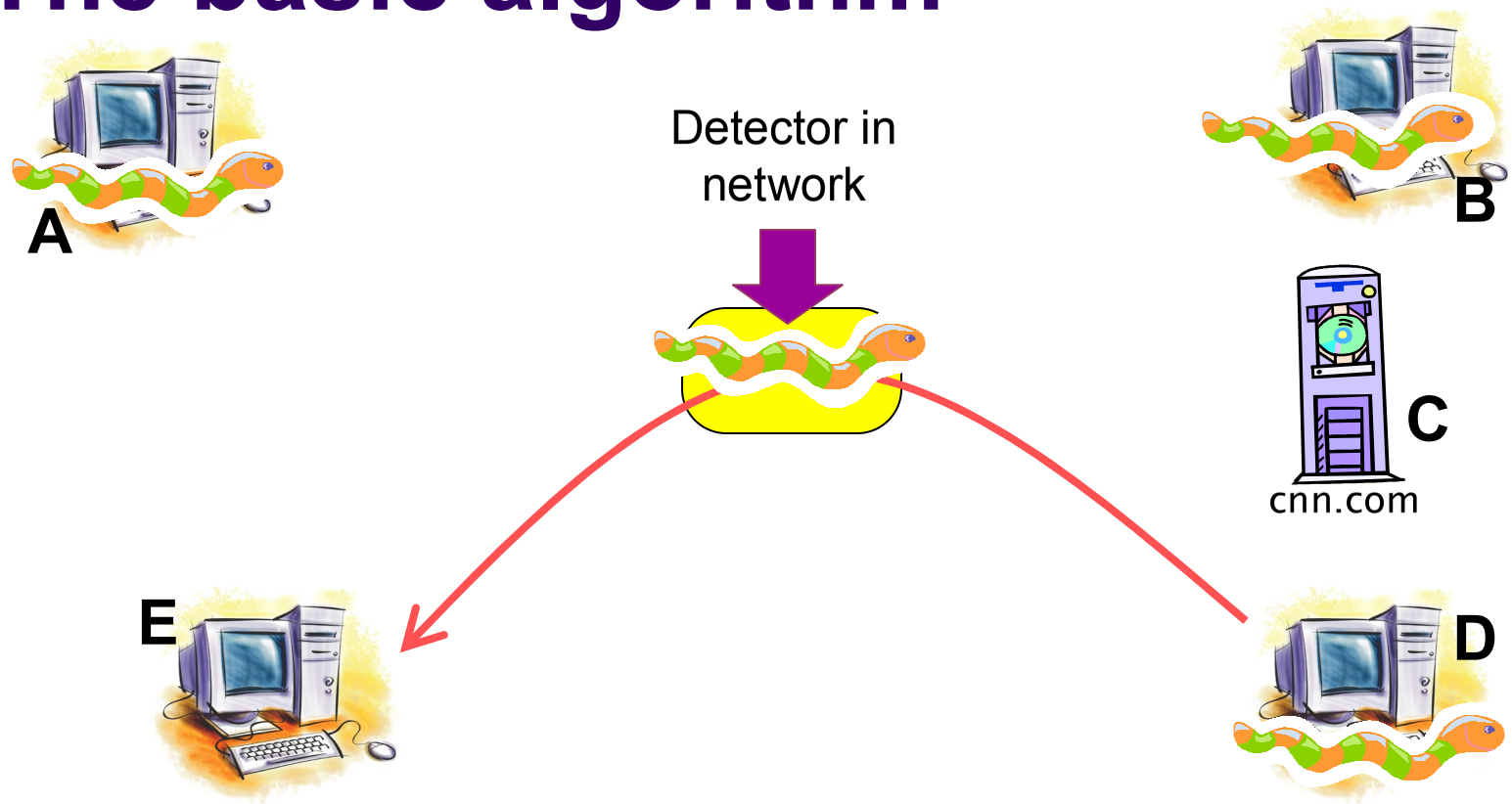
Prevalence Table

	**2**
	**1**

Address **Dispersion** Table

Sources	Destinations
2 (A,B)	**2 (B,D)**
1 (C)	**1 (A)**

The basic algorithm



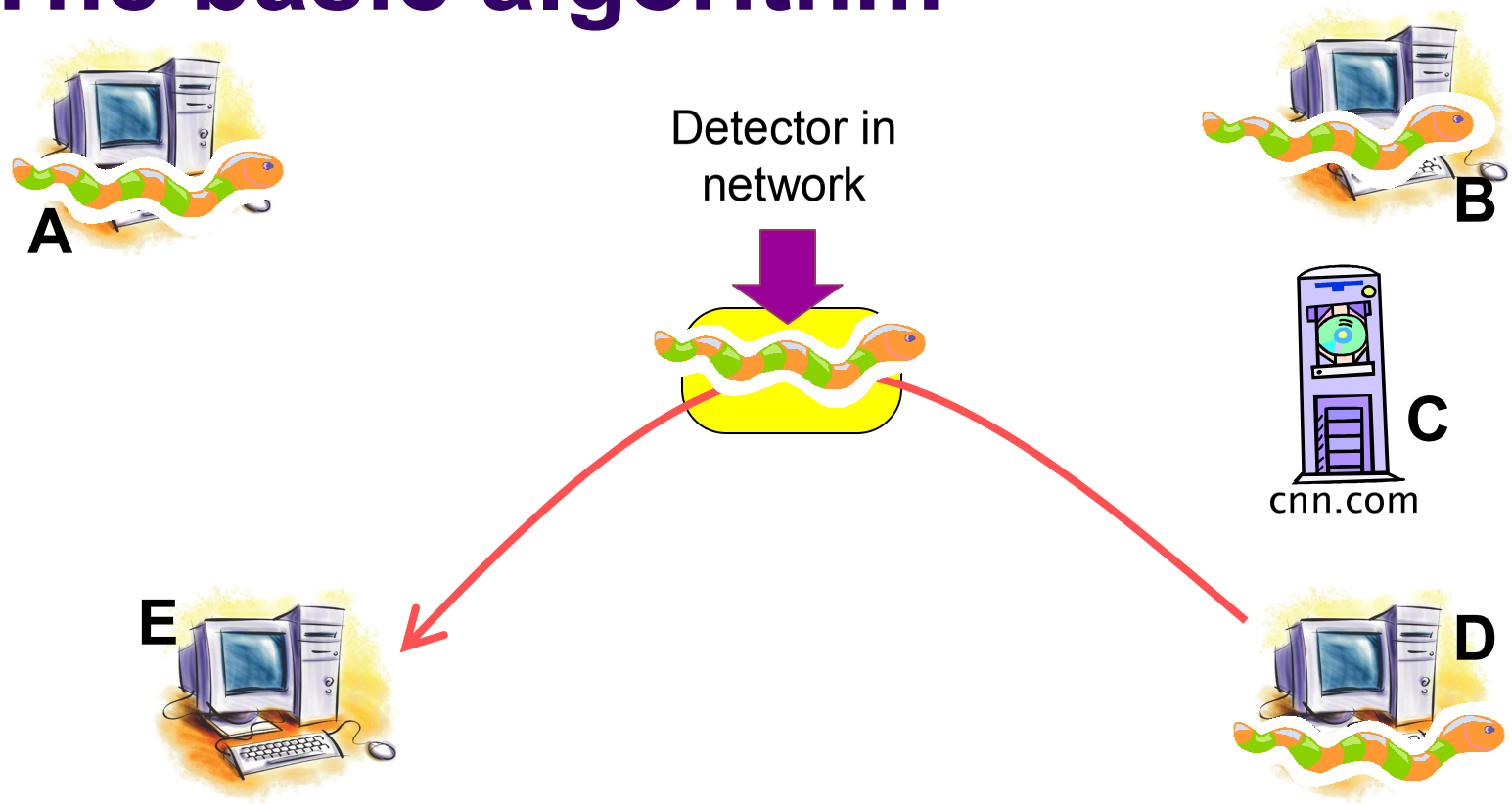
Prevalence Table

	3
	1

Address **Dispersion** Table

Sources	Destinations
3 (A,B,D)	3 (B,D,E)
1 (C)	1 (A)

The basic algorithm



Prevalence Table

	3
	1

Address **Dispersion** Table

Sources	Destinations
3 (A,B,D)	3 (B,D,E)
1 (C)	1 (A)

Challenges

- **Implementation practicality**
  - **Computation**
    - To support a 1Gbps line rate we have 12us to process each packet
      - Dominated by memory references; state expensive
    - Content sifting requires looking at ***every*** byte in a packet
  - **State**
    - On a fully-loaded 1Gbps link a naïve implementation can easily consume 100MB/sec for tables
    - Speed demands may limit to onchip SRAM on ASIC
- Lots of data structure/filtering tricks that make it doable
  - E.g. very few substrings are “popular”, so don’t store the other ones

Experience

- Generally *good.*
  - Detected and automatically generated signatures for **every** known worm outbreak over eight months
  - ***Can*** produce a precise signature for a new worm in a *fraction* of a second
- **Known worms detected**:
  - Code Red, Nimda, WebDav, Slammer, Opaserv, …
- **Unknown worms (with no public signatures) detected**:
  - MsBlaster, Bagle, Sasser, Kibvu, …

Key limitations: Evasion & DoS

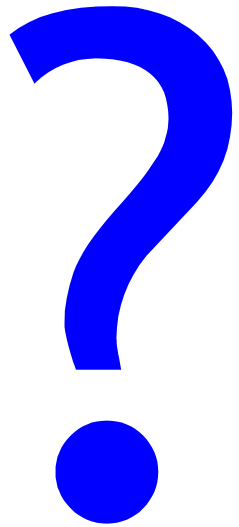
- Polymorphism/metamorphism
  - Newsom et al, *Polygraph: Automatically Generating Signatures for Polymorphic Worms*, Oakland '05
  - Kreugel et al, Polymorphic Worm Detection Using Structural Information of Executables, RAID '05
  - But losing battle, always favors bad guy
- Network evasion
  - Hide in protocol-level ambiguity, hard to normalize traffic at high-speed
  - Dharmapurikar et al, *Robust TCP Stream Reassembly in the Presence of Adversaries, USENIX Sec '05*
- End-to-end encryption
  - Fundamental conflict between organizational desire to impose security policy and employee/customer privacy
- Automated systems can be turned into weapons
  - What if I create some "worm-like" traffic that will produce the signature "Democrats" or "Republicans"?

Some other issues

- Lock down
  - If anomalies detected then reconfigure network into “minimal” mode (e.g. client X should only talk to server Y or server Q)
  - Used by some products
- Distributed alerting
  - You claim X is a signature for a worm, why should I trust you?
  - Vigilante’s Self-Certifying Alerts: elegant solution if your system gathers code
- How do you distribute patch/signature/filter?
  - Need to be faster than worm…
  - One crazy idea: Anti-worms
    - Castaneda et al, *Worm vs WORM: Preliminary Study of an Active counter-Attack Mechanism*, WORM ’04
  - Optimized broadcast tree

Summary

- Internet-connected hosts are highly vulnerable to worm outbreaks
  - Millions of hosts can be “taken” before anyone realizes
  - If only 10,000 hosts are targeted, no one may notice
- Prevention is a critical element, but there will always be outbreaks
- Treatment is a nightmare
- Containment requires fully automated response
- Different detection strategies, monitoring approaches, most at the research stage at best (few meaningful defenses in practice)
- Smart bad guys still have a huge advantage



http://www.ccied.org/