Addressing the Threat of Internet Worms

Vern Paxson

ICSI Center for Internet Research

and Lawrence Berkeley National Laboratory

vern@icir.org

February 22, 2005

What is a Worm?

- Self-replicating/self-propagating code.
- Spreads across a network by exploiting flaws in open services.
  - As opposed to viruses, which require user action to quicken/spread.
- Not new --- Morris Worm, Nov. 1988
  - 6-10% of all Internet hosts infected
- Many more since, but for 13 years none on that scale, until ….

Code Red

- Initial version released July 13, 2001.
- Exploited known bug in Microsoft IIS Web servers.
- Payload: web site defacement
  - **HELLO! Welcome to http://www.worm.com! Hacked By Chinese!**
  - Only done if language setting = English

Code Red of July 13, con’t

- 1st through 20th of each month: spread.
- 20th through end of each month: attack.
  - Flooding attack against 198.137.240.91 …
  - … i.e., *www.whitehouse.gov*
- Spread: via random scanning of 32-bit IP address space.

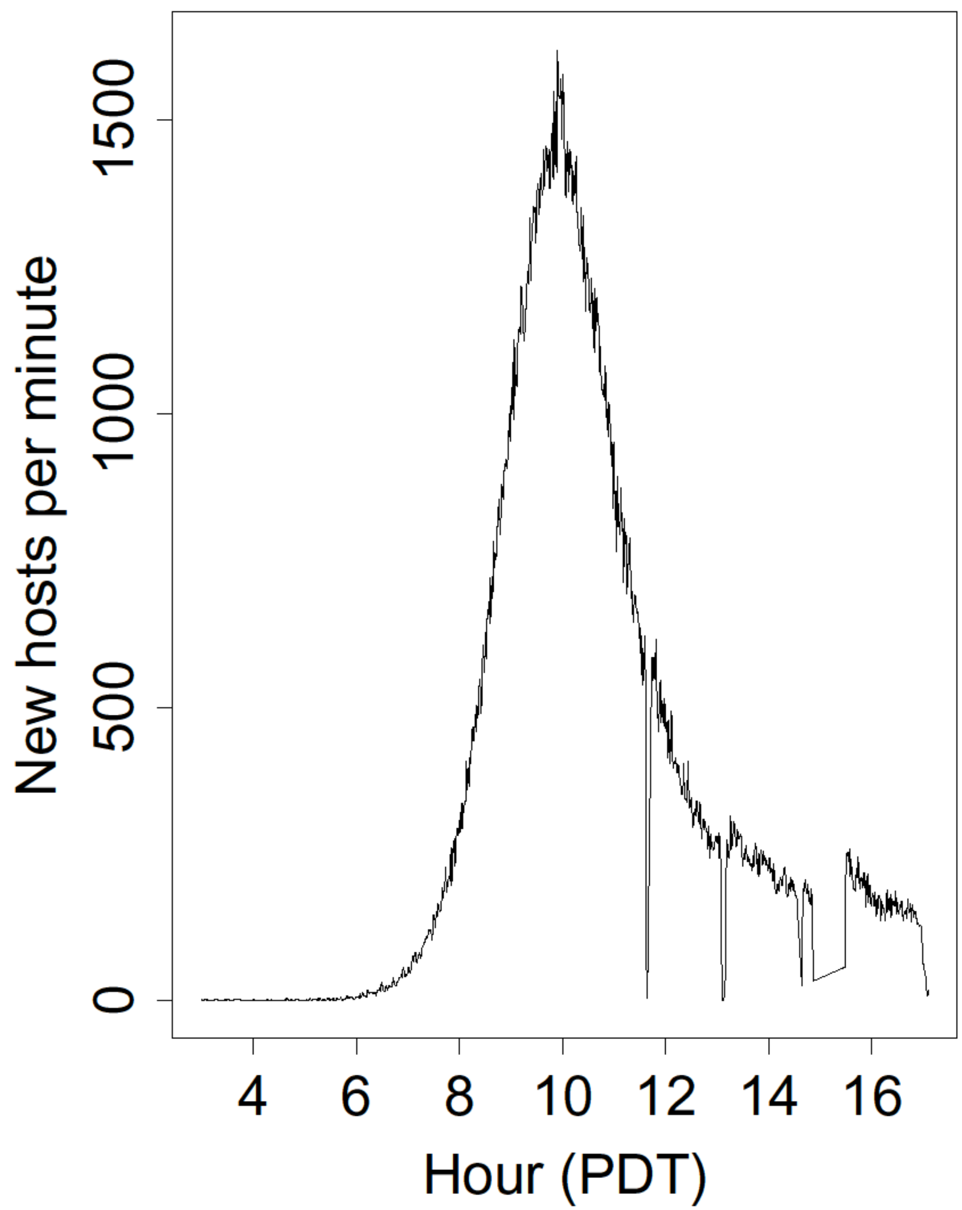
- But: failure to seed random number generator $\Rightarrow$ *linear growth*.

Code Red, con’t

- Revision released July 19, 2001.
- White House responds to threat of flooding attack by <u>changing the address</u> of *www.whitehouse.gov*
- Causes Code Red to <u>die</u> for date ≥ 20^{th} of the month.

- But: this time random number generator correctly seeded. Bingo!

Growth of Code Red Worm



Measuring Internet-Scale Activity: Network Telescopes

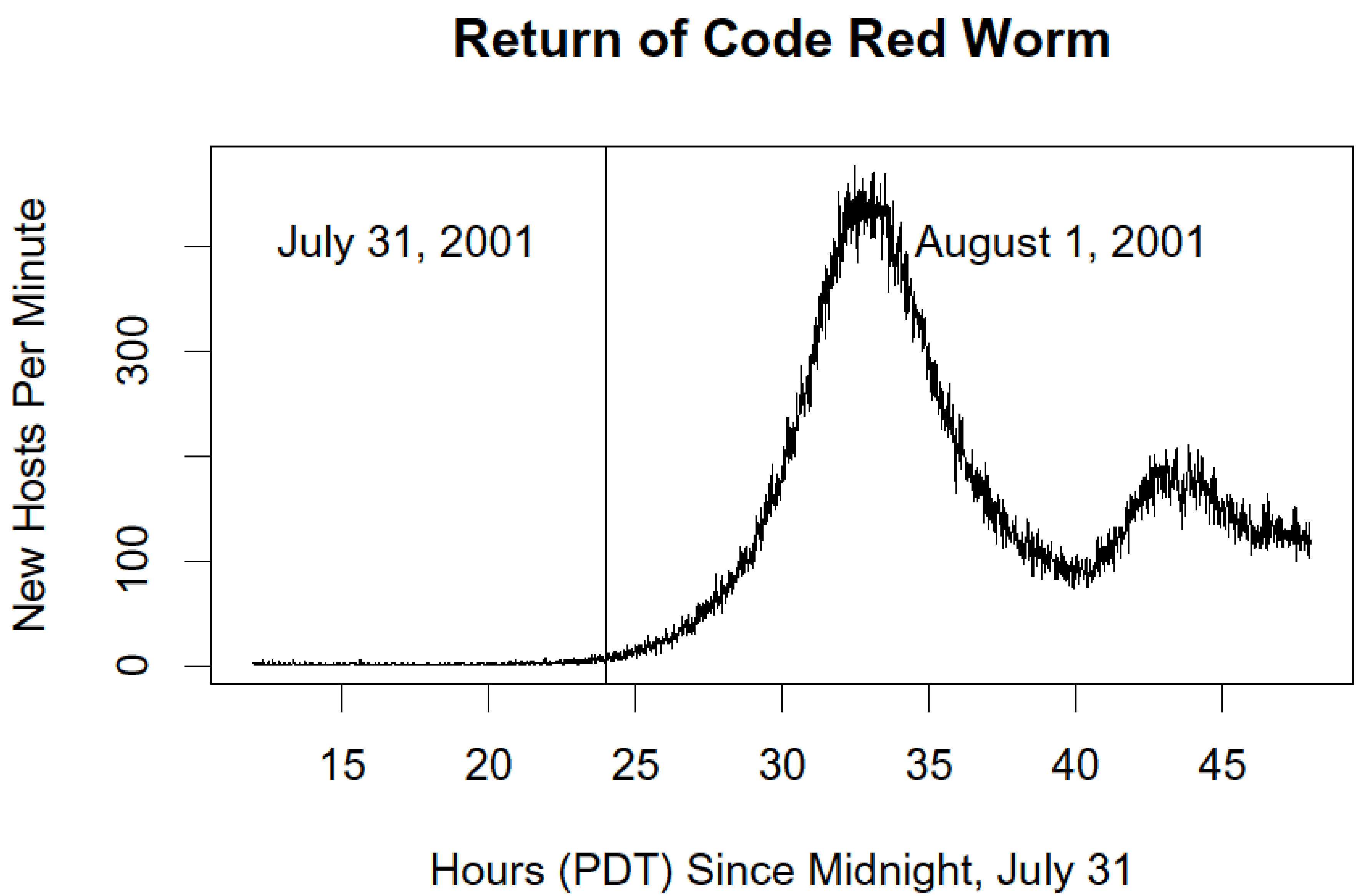
- Idea: monitor a cross-section of Internet address space to measure network traffic involving wide range of addresses
  - "Backscatter" from DOS floods
  - Attackers probing blindly
  - Random scanning from worms
- LBNL's cross-section: 1/32,768 of Internet
  - Small enough for appreciable *telescope lag*
- UCSD, UWisc's cross-section: 1/256.

Spread of Code Red

- Network telescopes estimate of # infected hosts: 360K. (Beware DHCP & NAT)
- Course of infection fits classic *logistic*.
- Note: larger the vulnerable population, *faster* the worm spreads.

- That night ($\Rightarrow$ 20th), worm dies …

  … except for hosts with inaccurate clocks!
- It just takes one of these to restart the worm on August 1st …



Return of Code Red Worm
July 31, 2001
August 1, 2001
New Hosts Per Minute
0
100
300
15
20
25
30
35
40
45
Hours (PDT) Since Midnight, July 31

Striving for Greater Virulence: Code Red 2

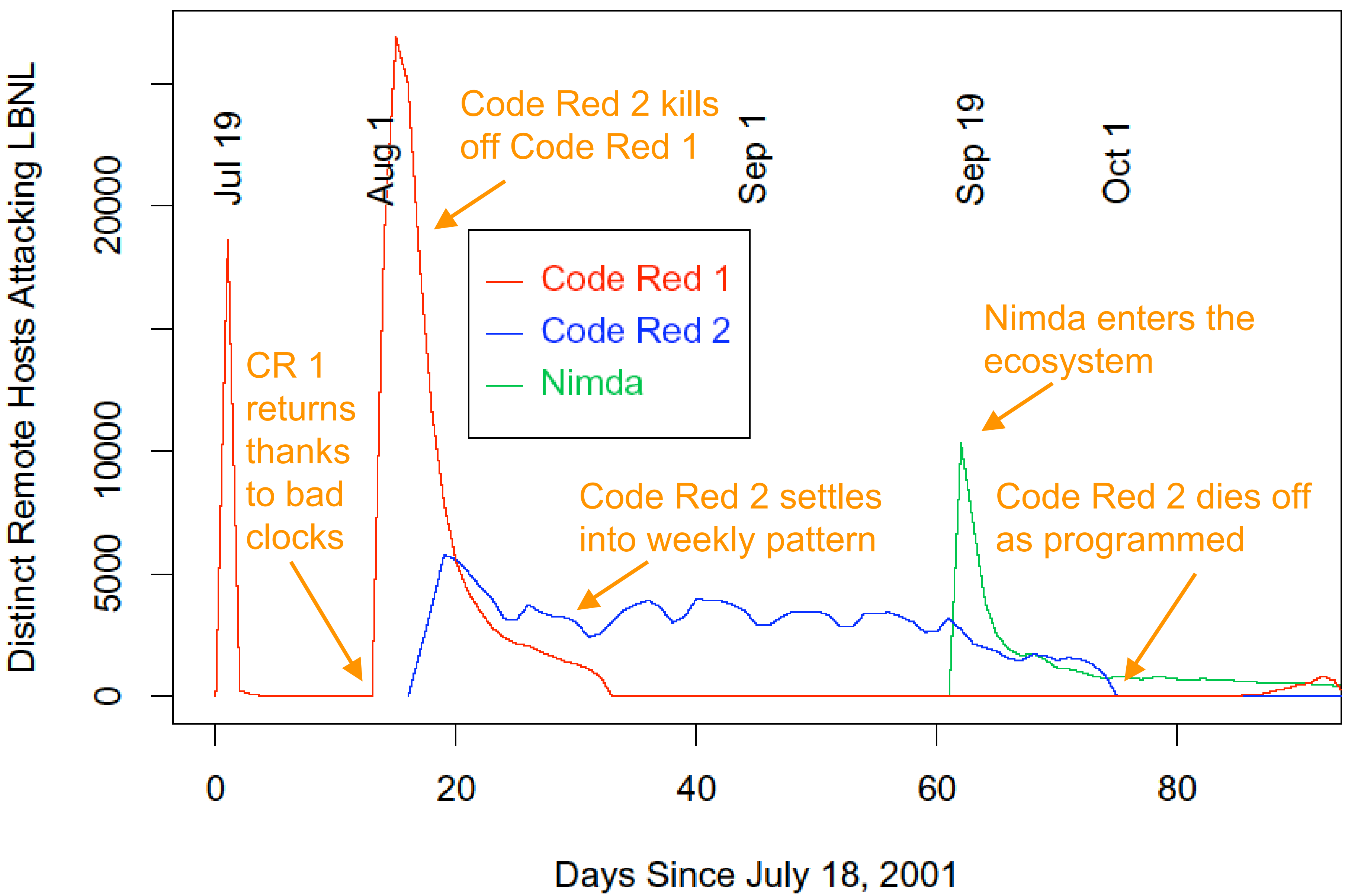
- Released August 4, 2001.
- Comment in code: “Code Red 2.”
  - But in fact completely different code base.
- Payload: a root backdoor, resilient to reboots.
- Bug: crashes NT, only works on Windows 2000.
- *Localized scanning*: prefers nearby addresses.
- <u>Kills Code Red 1</u>.
- Safety valve: programmed to die Oct 1, 2001.

Striving for Greater Virulence: Nimda

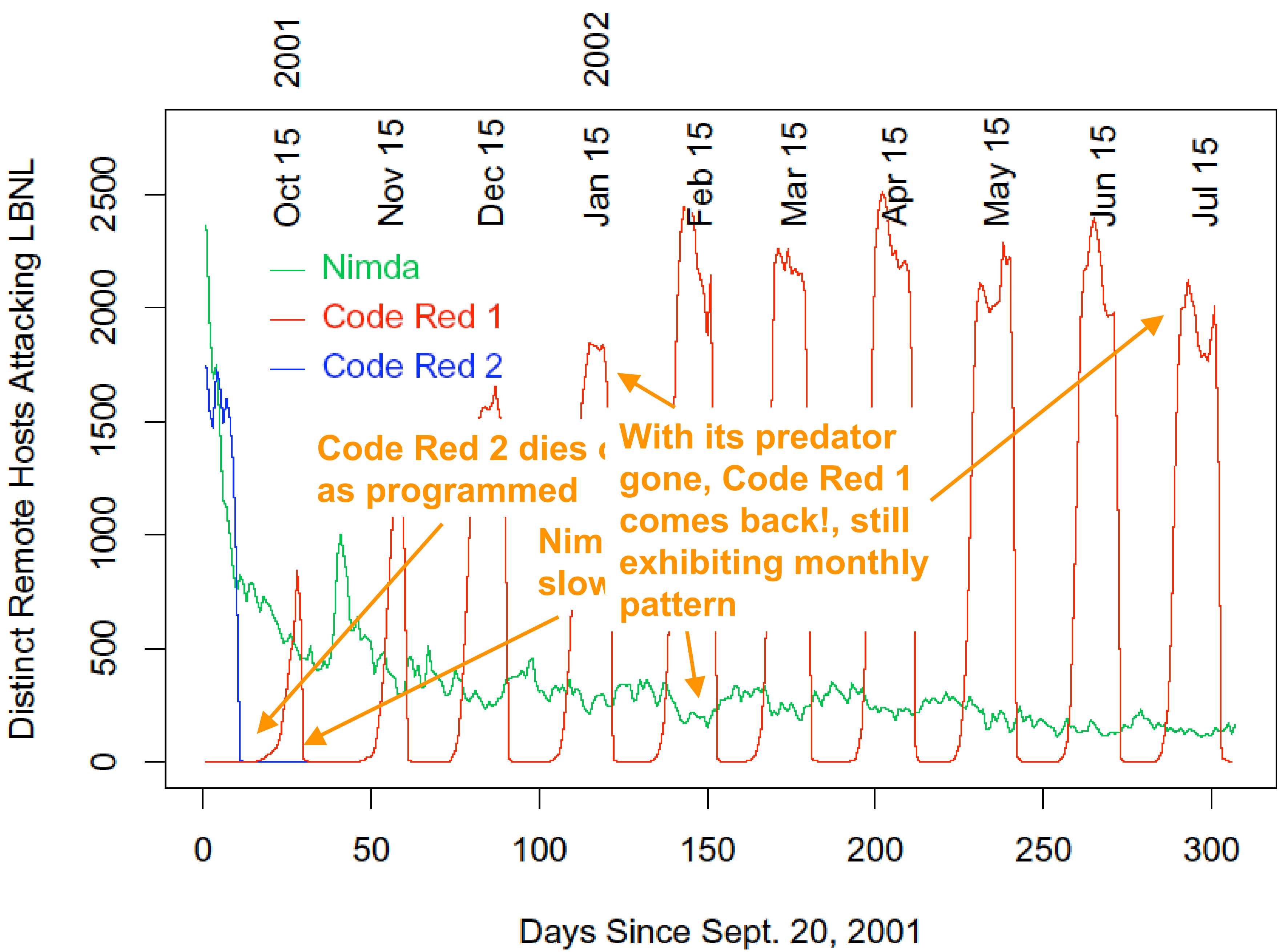
- Released September 18, 2001.
- Multi-mode spreading:
  - attack IIS servers via infected clients
  - email itself to address book as a virus
  - copy itself across open network shares
  - modifying Web pages on infected servers w/ client exploit
  - scanning for Code Red II backdoors (!)

$\Rightarrow$ worms form an *ecosystem*!

- Leaped across firewalls.

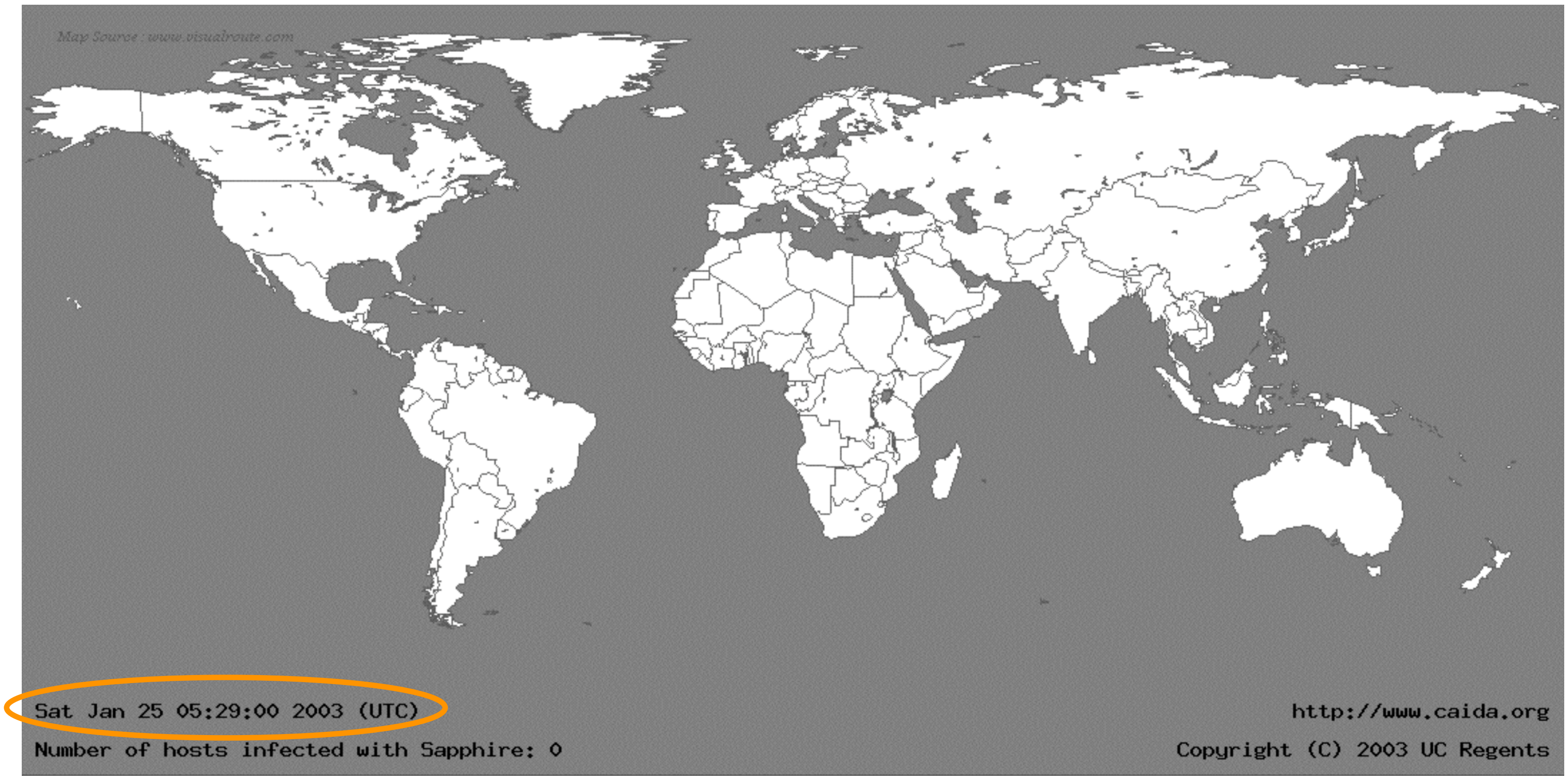


Distinct Remote Hosts Attacking LBNL
Days Since July 18, 2001
0
5000
10000
20000
0
20
40
60
80
Jul 19
Aug 1
Sep 1
Sep 19
Oct 1
Code Red 1
Code Red 2
Nimda
Code Red 2 kills off Code Red 1
CR 1 returns thanks to bad clocks
Nimda enters the ecosystem
Code Red 2 settles into weekly pattern
Code Red 2 dies off as programmed

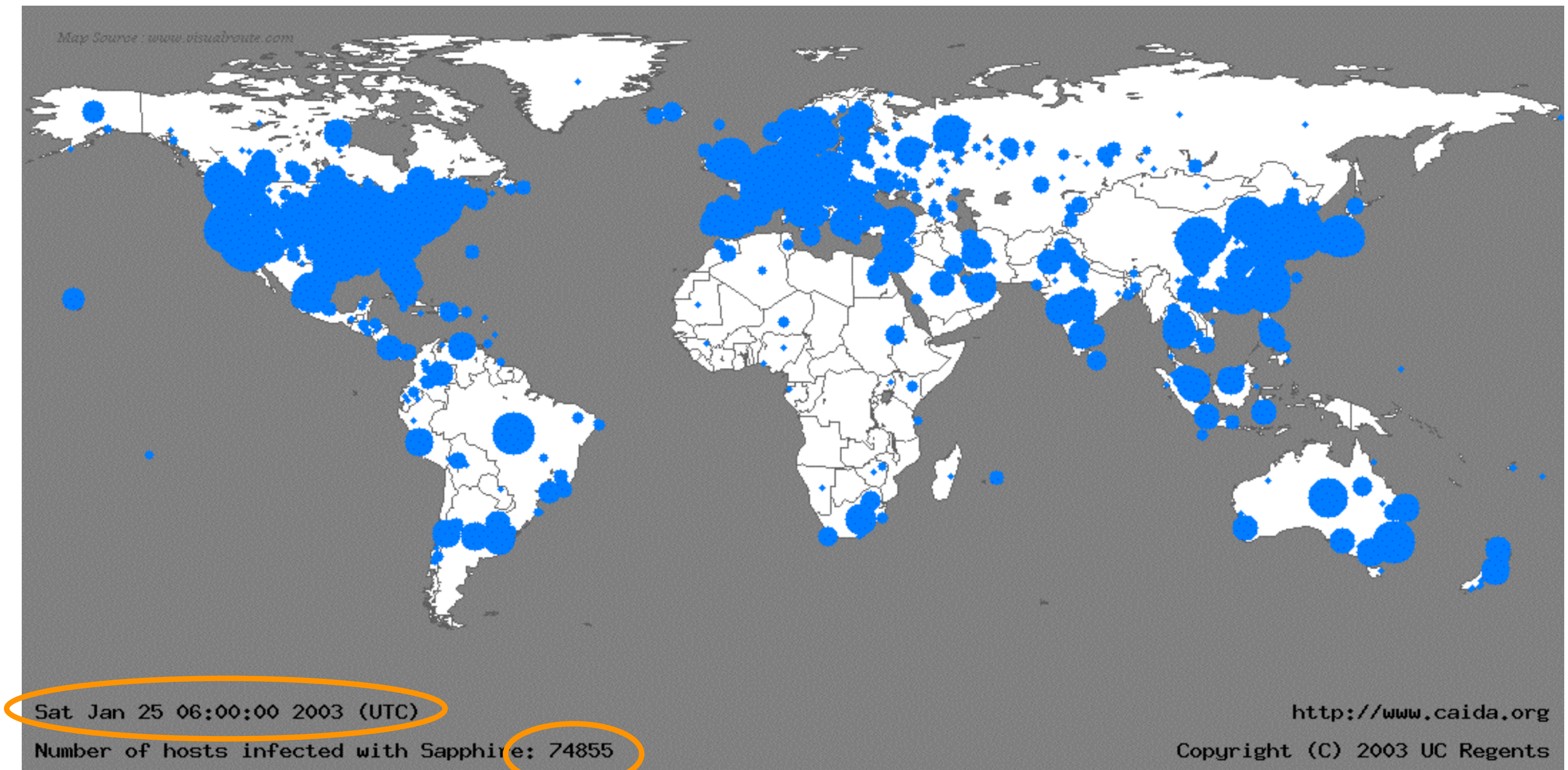


2001
2002
Oct 15
Nov 15
Dec 15
Jan 15
Feb 15
Mar 15
Apr 15
May 15
Jun 15
Jul 15
Nimda
Code Red 1
Code Red 2
Code Red 2 dies (
as programmed
Nim
slow
With its predator
gone, Code Red 1
comes back!, still
exhibiting monthly
pattern
Distinct Remote Hosts Attacking LBNL
0
500
1000
1500
2000
2500
0
50
100
150
200
250
300
Days Since Sept. 20, 2001

Life Just Before Slammer



Life Just After Slammer



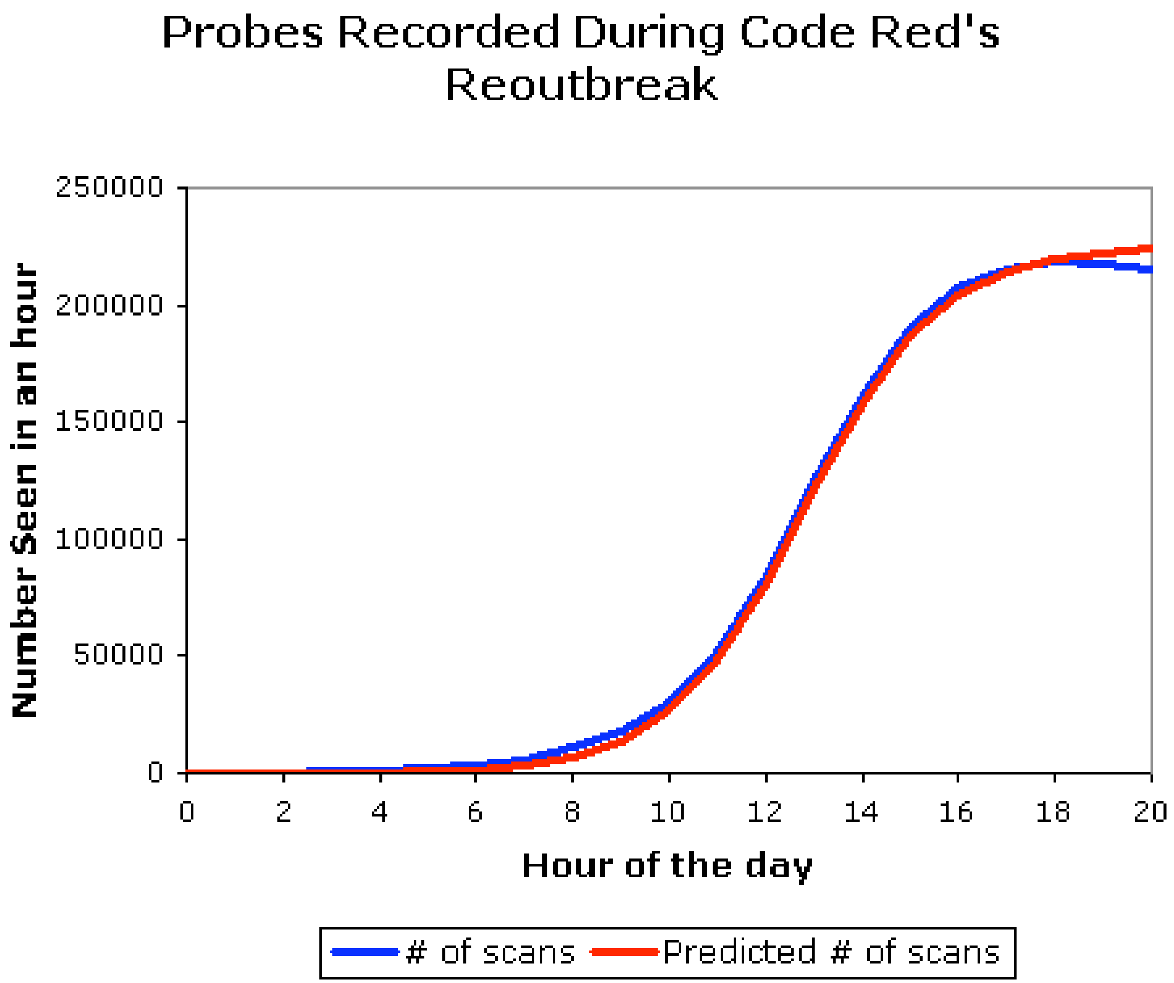
A Lesson in Economy

- Slammer exploited a <u>connectionless</u> UDP service, rather than connection-oriented TCP.
- *Entire worm* fit in a single packet!

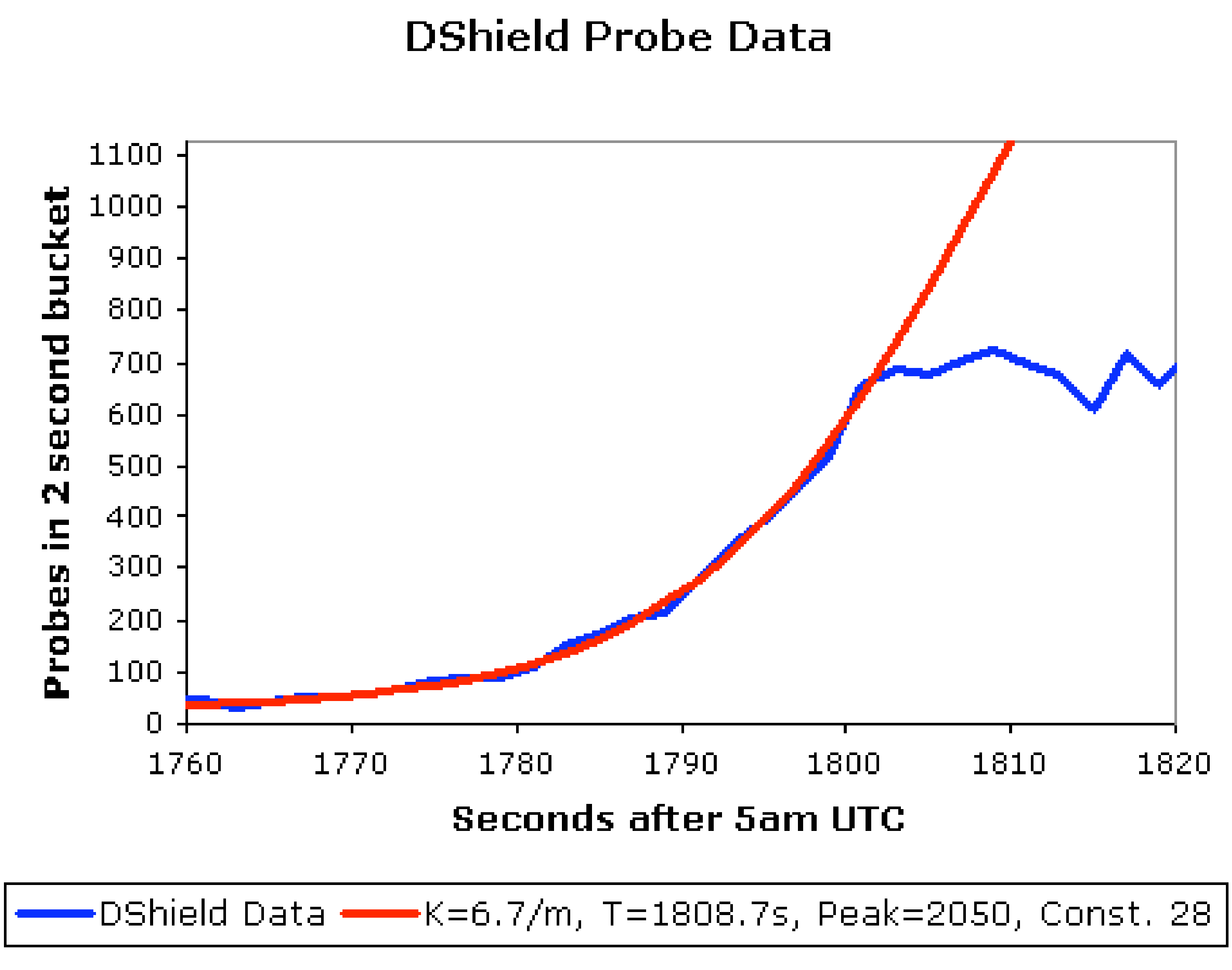
⇒ When scanning, worm could “fire and forget”.

- Worm infected 75,000+ hosts in <u>10 minutes</u> (despite broken random number generator).
  - At its peak, **doubled every <u>8.5 seconds</u>**
- Progress limited by the Internet’s *carrying capacity*!

The Usual Logistic Growth

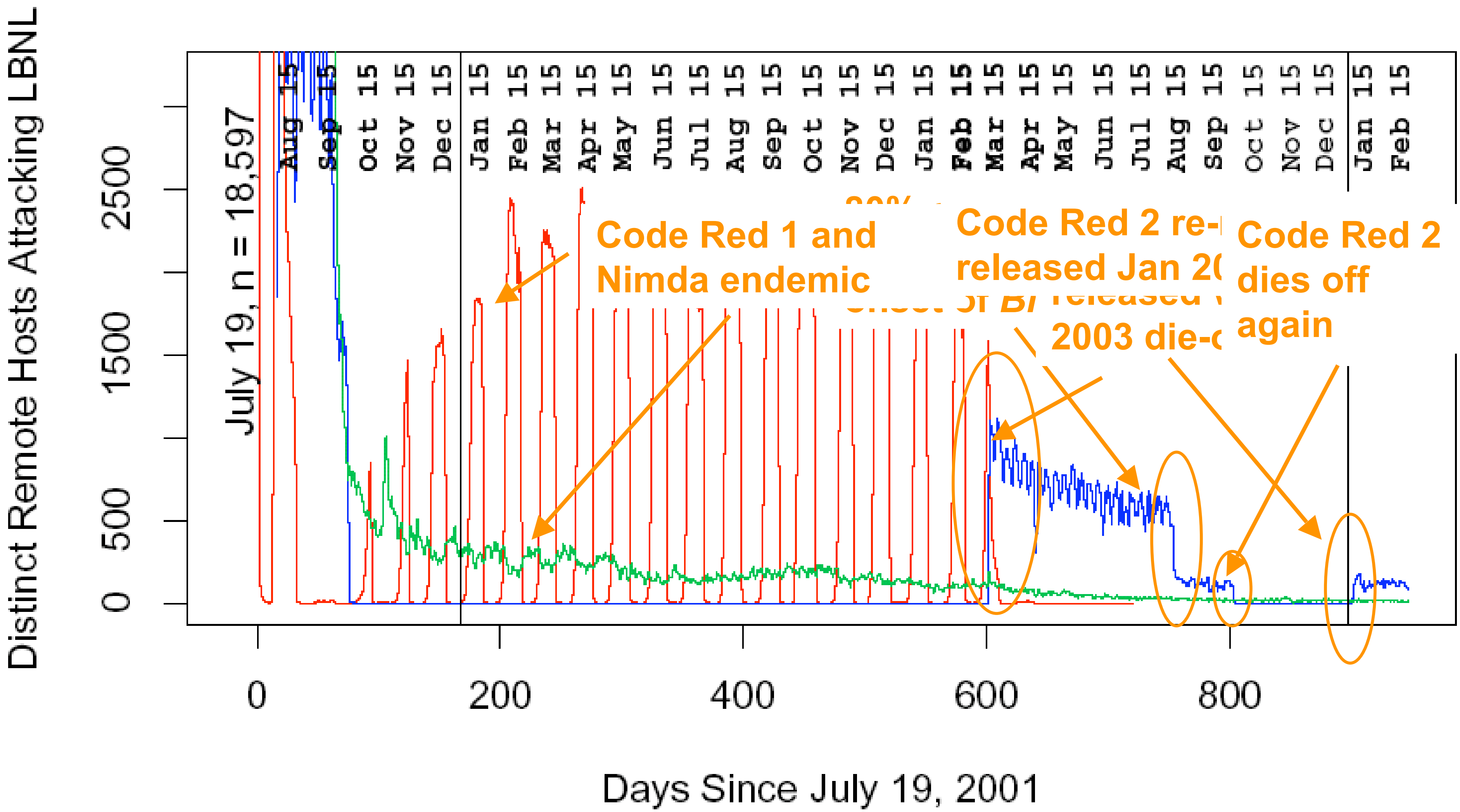


Slammer's *Bandwidth-Limited* Growth



Blaster

- Released August 11, 2003.
- Exploits flaw in RPC service ubiquitous across Windows.
- Payload: attack Microsoft Windows Update.
- Despite flawed scanning and secondary infection strategy, rapidly propagates to (*at least*) 100K's of hosts.
- Actually, bulk of infections are really *Nachia*, a Blaster <u>counter-worm</u>.
- Key paradigm shift: *firewalls don't help*.



Distinct Remote Hosts Attacking LBNL
0
500
1500
2500
July 19, n = 18,597
Aug 15
Sep 15
Oct 15
Nov 15
Dec 15
Jan 15
Feb 15
Mar 15
Apr 15
May 15
Jun 15
Jul 15
Aug 15
Sep 15
Oct 15
Nov 15
Dec 15
Jan 15
Feb 15
Mar 15
Apr 15
May 15
Jun 15
Jul 15
Aug 15
Sep 15
Oct 15
Nov 15
Dec 15
Jan 15
Feb 15
Code Red 1 and
Nimda endemic
Code Red 2
dies off
again
0
200
400
600
800
Days Since July 19, 2001

What if Spreading Were Well-Designed?

- Observation (Weaver): Much of a worm's scanning is <u>redundant</u>.
- Idea: *coordinated* scanning
  - Construct permutation of address space
  - Each new worm starts at a random point
  - Worm instance that "encounters" another instance re-randomizes.

$\Rightarrow$ Greatly accelerates worm in later stages.

What if Spreading Were Well-Designed?, con't

- Observation (Weaver): Accelerate initial phase using a precomputed hit-list of say 1% vulnerable hosts.

$\Rightarrow$ At 100 scans/worm/sec, can infect huge population in a few minutes.

- Observation (Staniford): Compute hit-list of entire vulnerable population, propagate via divide & conquer.

$\Rightarrow$ With careful design, 10^6 hosts in < 2 sec!

Defenses

- <u>Detect</u> via *honeyfarms*: collections of "honeypots" fed by a network telescope.
  - Any outbound connection from honeyfarm = worm.
    (at least, that's the theory)
  - Distill *signature* from inbound/outbound traffic.
  - If telescope covers N addresses, expect detection when worm has infected 1/N of population.
  - Major issues regarding *filtering*
- <u>Thwart</u> via *scan suppressors*: network elements that block traffic from hosts that make failed connection attempts to too many other hosts.

Defenses?

- Observation:
  worms don’t need to randomly scan

- *Meta-server* worm: ask server for hosts to infect (e.g., Google for “**powered by phpbb**”)
- *Topological* worm: fuel the spread with local information from infected hosts (web server logs, email address books, config files, SSH “known hosts”)

$\Rightarrow$ No scanning signature; with rich inter-connection topology, potentially very fast.

Defenses??

- *Contagion* worm: propagate parasitically along with normally initiated communication.

- E.g., using 2 exploits - Web browser & Web server - infect any vulnerable servers visited by browser, then any vulnerable browsers that come to those servers.
- E.g., using 1 BitTorrent exploit, glide along immense peer-to-peer network in days/hours.

$\Rightarrow$ <u>No unusual connection activity</u> at all! **:-(**

Some Cheery Thoughts

(Stefan Savage, UCSD/CCIED)

- 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

Broader View of Defenses

- Prevention -- *make the monoculture hardier*
  - Get the darn code right in the first place …
    - … or figure out what's wrong with it and fix it
  - Lots of active research (static & dynamic methods)
  - Security reviews now taken seriously by industry
    - E.g., ~$200M just to *review* Windows Server 2003
  - But very expensive
  - And very large Installed Base problem

- Prevention -- *diversify the monoculture*
  - Via exploiting existing heterogeneity
  - Via creating artificial heterogeneity

Broader View of Defenses, con't

- Prevention -- *keep vulnerabilities inaccessible*
  - Cisco's *Network Admission Control*
    - Frisk hosts that try to connect, block if vulnerable
  - Microsoft's *Shield* ("*Band-Aid*")
    - Shim-layer blocks network traffic that fits known *vulnerability* (rather than known *exploit*)
- Detection -- *look for unusual repeated content*
  - Can work on non-scanning worms
  - Key off *many-to-many* communication to avoid confusion w/ non-worm sources
  - EarlyBird, Autograph -- distill signature
  - But: what about polymorphic worms?

Once You Have A Live Worm, Then What?

- *Containment*
  - Use distilled signature to prevent further spread
  - Different granularities possible:
    - Infectees (doesn't scale well)
    - Content (or more abstract activity) description
    - Vulnerable population
- Would like to leverage detections by others
  - But how can you *trust* these?
  - What if it's an attacker *lying* to you to provoke a self-damaging response? (Or to hide a later actual attack)

Once You Have A Live Worm, What Then?, con't

- *Proof of infection*
  - Idea: alerts come with a *verifiable audit trail* that demonstrates the exploit, ala' proof-carrying code
- *Auto-patching*
  - Techniques to derive (and test!) patches to fix vulnerabilities in real-time

    (Excerpt from my review: "*Not as crazy as it sounds*")
- *Auto-antiworm*
  - Techniques to automatically derive a new worm from a propagating one, but with disinfectant payload

    (This one, on the other hand, is as crazy as it sounds)

Incidental Damage … Today

- Today's worms have significant real-world impact:
  - Code Red disrupted routing
  - Slammer disrupted elections, ATMs, airline schedules, operations at an off-line nuclear power plant …
  - Blaster possibly contributed to Great Blackout of Aug. 2003 … ?
  - Plus *major* clean-up costs
- But today's worms are <u>*amateurish*</u>
  - <u>Unimaginative</u> payloads

Where are the Nastier Worms??

- Botched propagation the norm
- Doesn't anyone *read the literature*?

  e.g. permutation scanning, flash worms,
  metaserver worms, topological, contagion
- Botched payloads the norm

  e.g. Flooding-attack fizzles

⇒ Current worm authors are in it for kicks …
(… or testing) *No arms race.*

Next-Generation Worm Authors

- Military.
- Crooks:
  - Denial-of-service, spamming for hire
  - *"Access worms"*
  - Very worrisome onset of *blended threats*
    - Worms + viruses + spamming + phishing + DOS-for-hire + botnets + spyware
- Money on the table ⇒ Arms race

  (market price for spam proxies: 3-10¢/host/week)

“Better” Payloads

- Wiping a disk costs $550/$2550*
- “A well-designed version of Blaster could have infected 10M machines.” (8M+ for sure!)
- The same service exploited by Blaster has other vulnerabilities …
- Potentially a lot more $$$: flashing BIOS, corrupting databases, spreadsheets …
- Lower-bound estimate: $50B if well-designed

Attacks on Passive Monitoring

- Exploits for bugs in read-only analyzers!
- Suppose protocol analyzer has an error parsing unusual type of packet
  - E.g., `tcpdump` and malformed options
- Adversary crafts such a packet, overruns buffer, causes analyzer to <u>execute arbitrary code</u>

Witty

- Released March 19, 2004.
- Single UDP packet exploits flaw in the *passive analysis* of Internet Security Systems products.
- “Bandwidth-limited” UDP worm ala’ Slammer.
- Vulnerable pop. (12K) attained in 75 minutes.
- Payload: *slowly corrupt random disk blocks*.

Witty, con’t

- Flaw had been announced the *previous day*.
- Telescope analysis reveals:
  - Initial spread seeded via a *hit-list*.
  - In fact, targeted a U.S. military base.
  - Analysis also reveals “Patient Zero”, a European retail ISP.
- Written by a Pro.

How Will Defenses Evolve?

- Wide-area *automated* coordination/decision-making/trust <u>very hard</u>
- More sophisticated spreading paradigms will require:
  - *Rich application analysis*

    coupled with
  - *Well-developed anomaly detection*

What do we need?

- Hardening of end hosts
- Traces of both worms and esp. *background*
- Topologies reflecting application-interconn.
- Funding that isn't classified

- Good, basic thinking:
  – This area is still *young* and there is a lot of low-hanging fruit / clever insight awaiting …

But At Least Us Researchers are Having Fun …

- Very challenging research problems
  - Immense scale
  - Coordination across disparate parties
  - Application anomaly detection
  - Automated response

- Whole new sub-area
  - What seems hopeless today …
    … can suddenly yield prospects tomorrow.
  - And vice versa: tomorrow can be much more bleak than today!