How to 0wn the Internet in Your Spare Time

Stuart Staniford

Silicon Defense
Eureka, CA
stuart@silicondefense.com

Vern Paxson

ICSI Center for Internet Research (ICIR)
and
Lawrence Berkeley National Laboratory
vern@{icir.org, ee.lbl.gov}

Nicholas Weaver

Computer Science
University of California
Berkeley, CA
nweaver@cs.berkeley.edu

April 1, 2002

What could you do if you 0wn'd a million hosts?

Launch: immensely <u>diffuse</u> DDOS attacks.

- no need to spoof addresses
- can make low-rate requests

⇒ Can make legitimate requests

⇒ <u>Way</u> beyond state-of-the-art to defend against.

- Or: launch 100 concurrent, well-targetted DDOS attacks
  – root name servers, NANOG/Bugtraq, CNN, ...
  – *critical infrastructure?*

What could you do if you 0wn’d a million hosts?, con’t

Access: sensitive information.

Passwords, credit card numbers, address books,
archived email, patterns of user activity, illicit content.

Search: for needles in haystacks.

e.g., search for particular admin’s password
e.g., grep for classified information
e.g., crack crypto keys

Confuse: by corrupting information,

sending out misinformation.

Immense damage: cyberwarfare between nations; terrorism.

How to 0wn a million hosts? — Worms.

Self-replicating/self-propagating code.

Spread by exploiting flaws in open services.

(As opposed to viruses, which require user action to spread.)

Not new — Morris Worm of November 1988:

$\approx$ 10% of Internet hosts infected.

Many since: Ramen, Cheese, sadmind, Goner, Lion, Badtrans, Adore . . .

How bad can it get?

Code Red.

Code Red:

Initial version released July 13, 2001.

Exploited known bug in Microsoft IIS Web servers.

Payload: web site defacement.

Spread by random scanning of 32-bit IP address space.

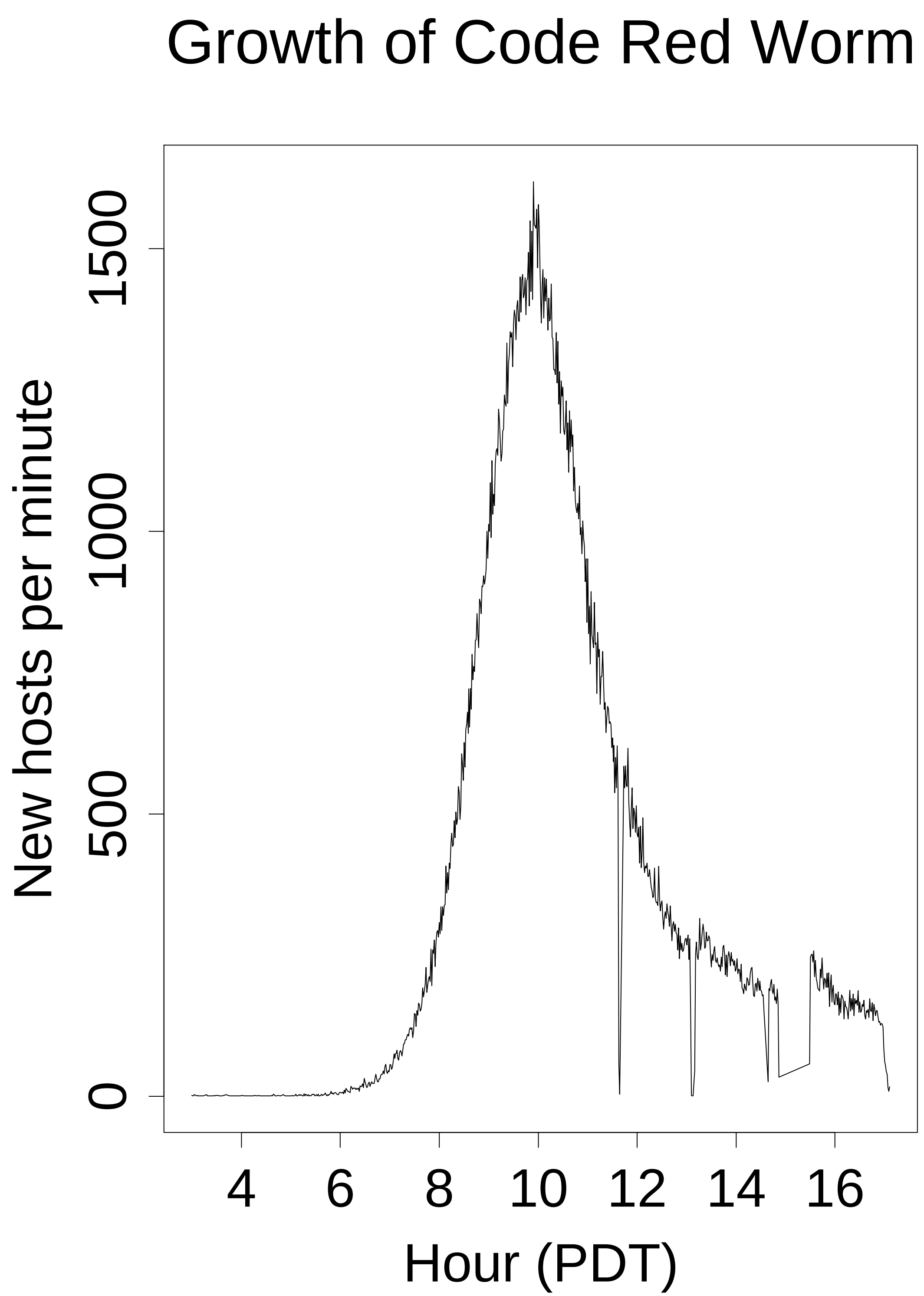
But: failure to seed random number generator $\Rightarrow$ linear growth.

“CRv2” released July 19, 2001.

Payload: flooding attack on `www.whitehouse.gov`.

Bug lead to it <u>dying</u> for date $\geq$ 20th of the month.

But: this time random number generator correctly seeded.


Growth of Code Red Worm
New hosts per minute
0
500
1000
1500
4
6
8
10
12
14
16
Hour (PDT)

Spread of Code Red:

Monitoring two class B's $\Rightarrow$ 300,000 infected hosts.

Analytic model:

$N =$ total number of vulnerable hosts

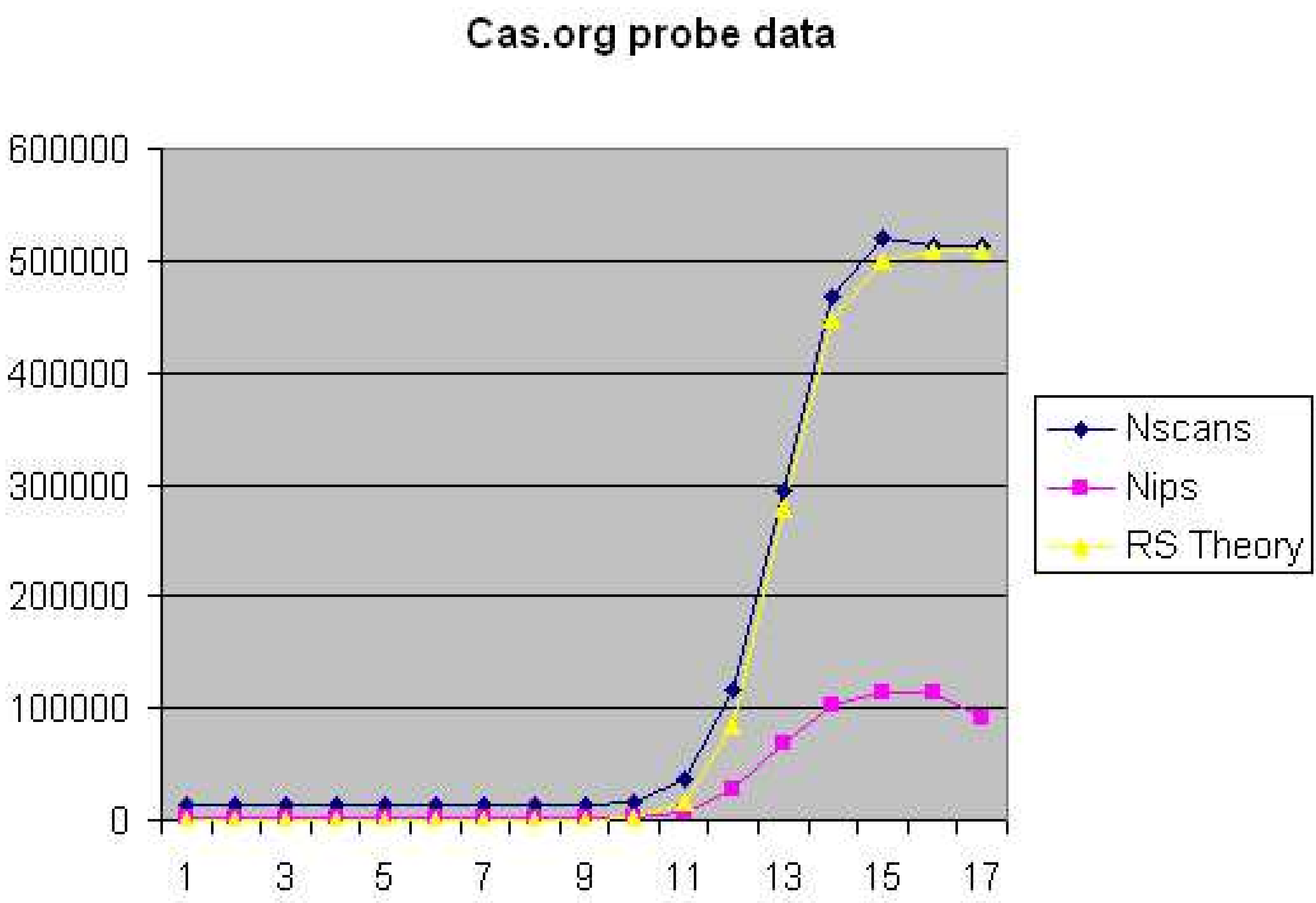
$K =$ compromise rate, new hosts/host/hour

$a(t) =$ proportion of vulnerable machines compromised at time t

Then:

$$a(t) = \frac{e^{K(t-T)}}{1 + e^{K(t-T)}}$$

$\Rightarrow$ *logistic* growth.



Cas.org probe data
600000
500000
400000
300000
200000
100000
0
1
3
5
7
9
11
13
15
17
Nscans
Nips
RS Theory

Spread of Code Red, con't:

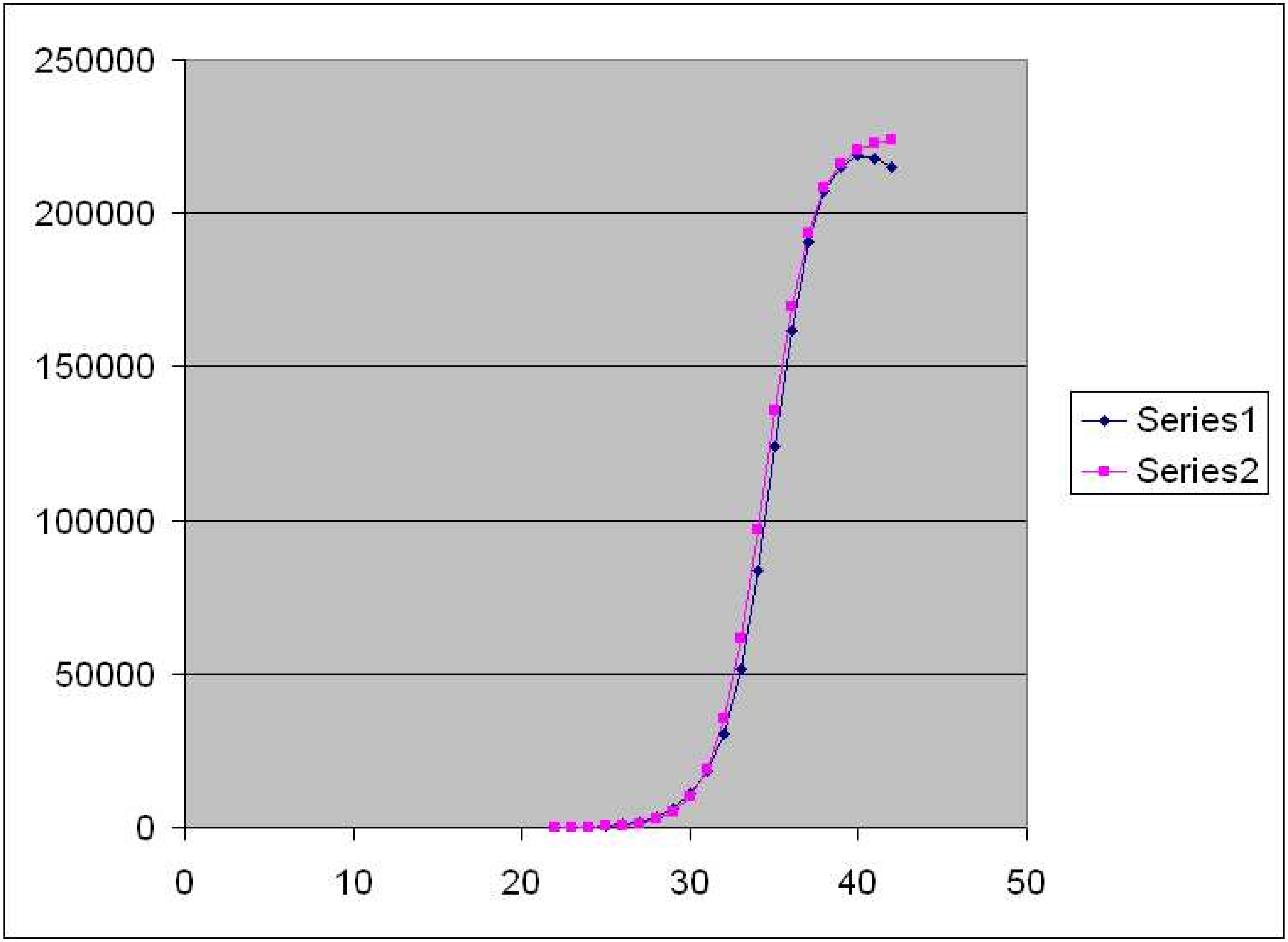
Discrepancies in part due to background scanning rate.

Fit gives $K = 1.8$.

That night, Code Red dies . . .

. . . *except* for hosts with inaccurate clocks!

It just takes *one* of these to restart the worm come the first of the next month!



July 31 / August 1, 2001.

Achieving greater virulence — Code Red II:

Released August 4, 2001.

Comment in code: “Code Red II.”

But in fact completely different code base.

Payload: a root backdoor, resilient to reboots.

Bug: crashes NT, only works right on Windows 2000.

Localized scanning:

- scans its own /16 with probability $\frac{3}{8}$
- scans its own /8 with probability $\frac{1}{2}$
- scans randomly with probability $\frac{1}{8}$

Kills Code Red I.

Achieving greater virulence — Nimda:

Released September 18, 2001.

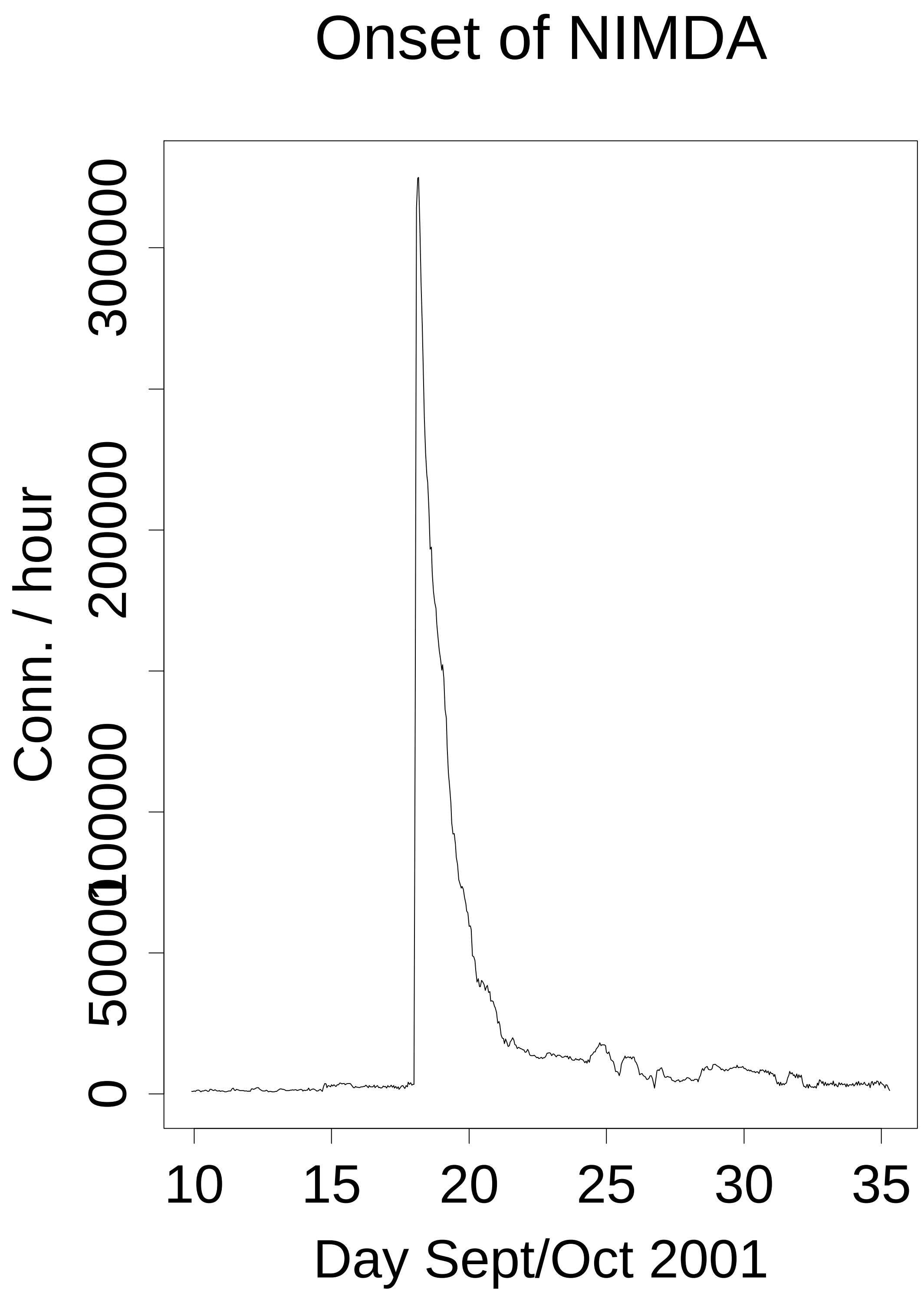
Multi-mode spreading:

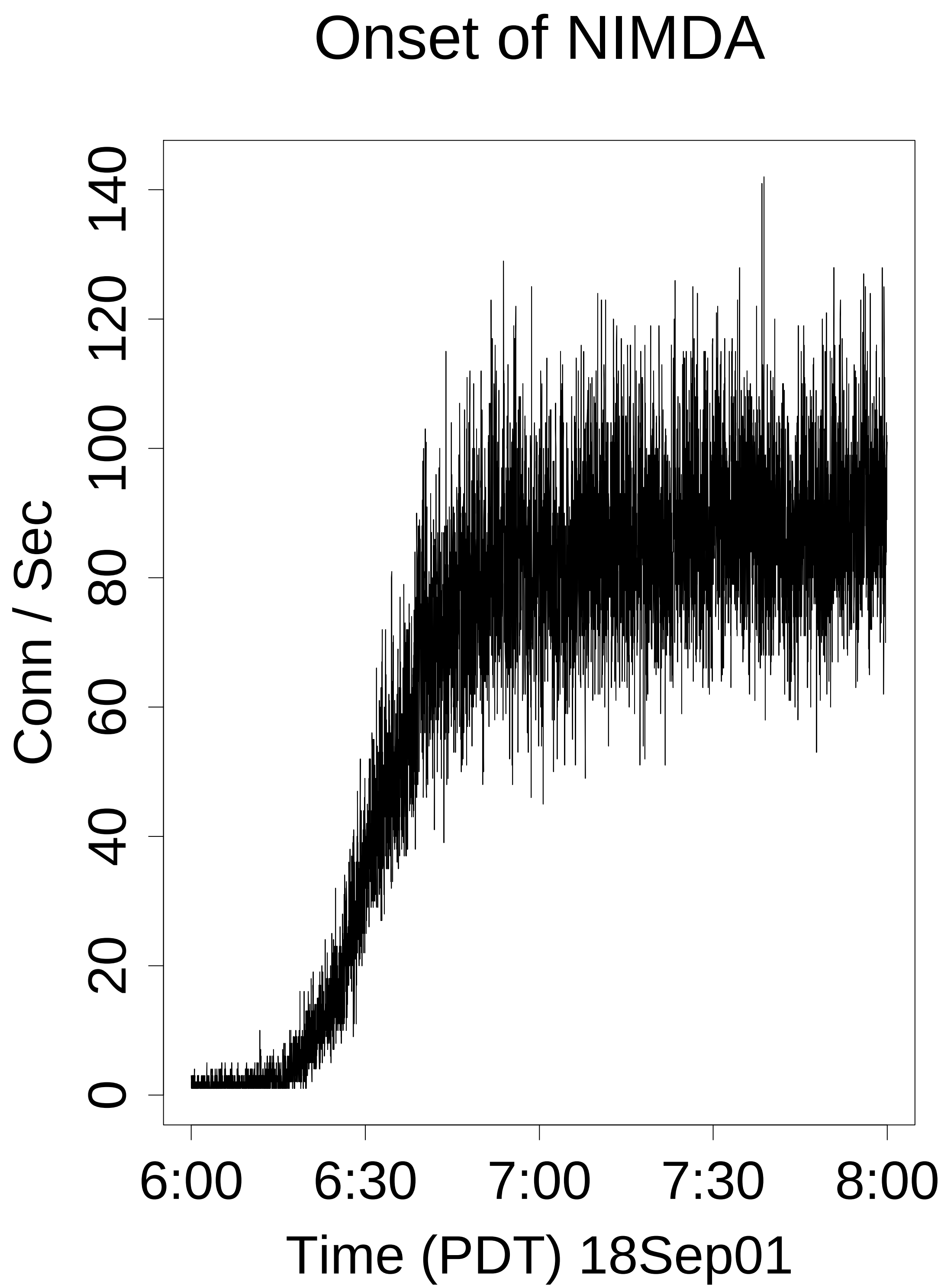
- attack IIS servers via infected clients
- email itself to address book as a virus
- copy itself across open nework shares
- modifying Web pages on infected servers w/ client exploit
- scanning for Code Red II and sadmind backdoors (!)

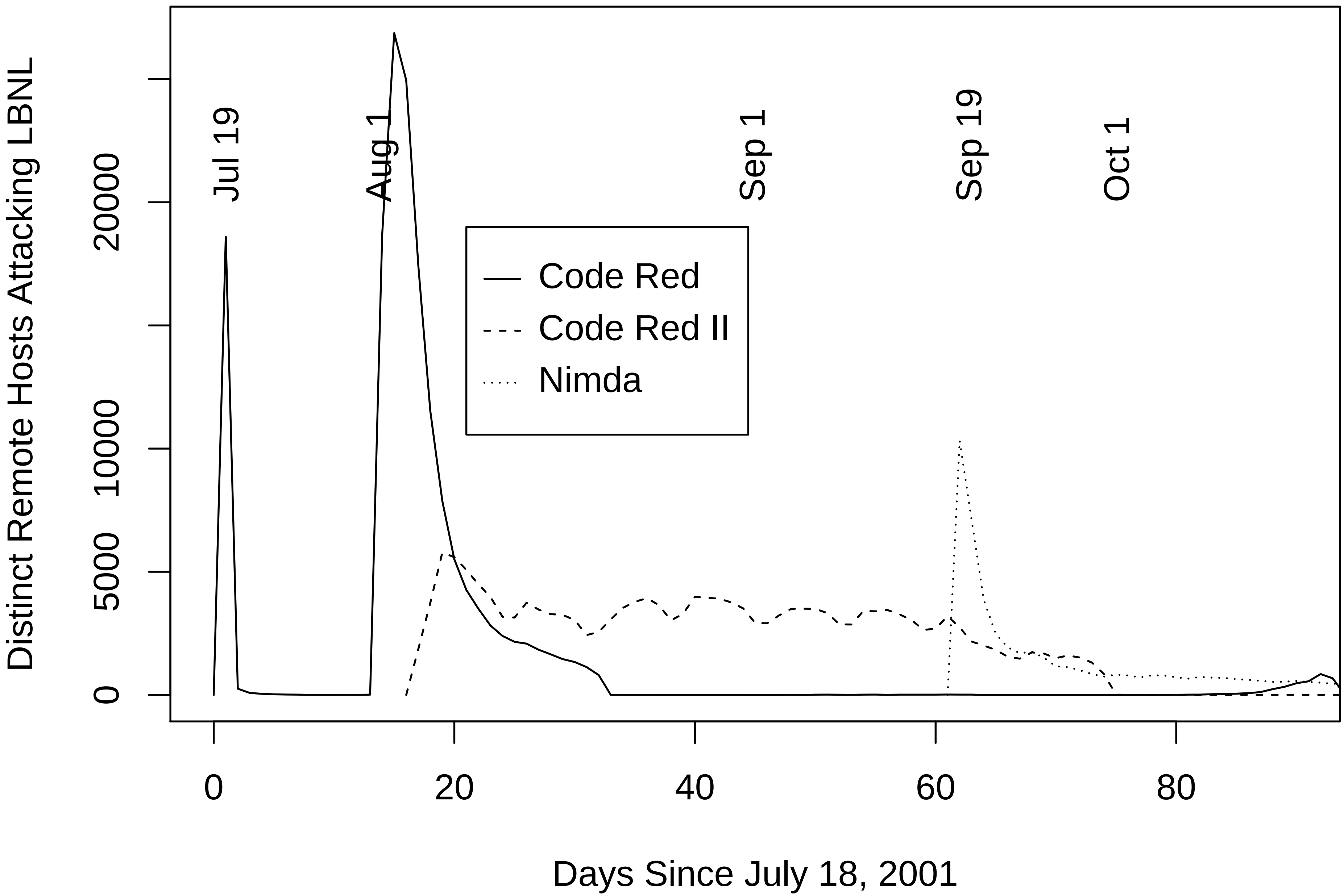
  $\Rightarrow$ worms form an *ecosystem!*

Leaped across firewalls.

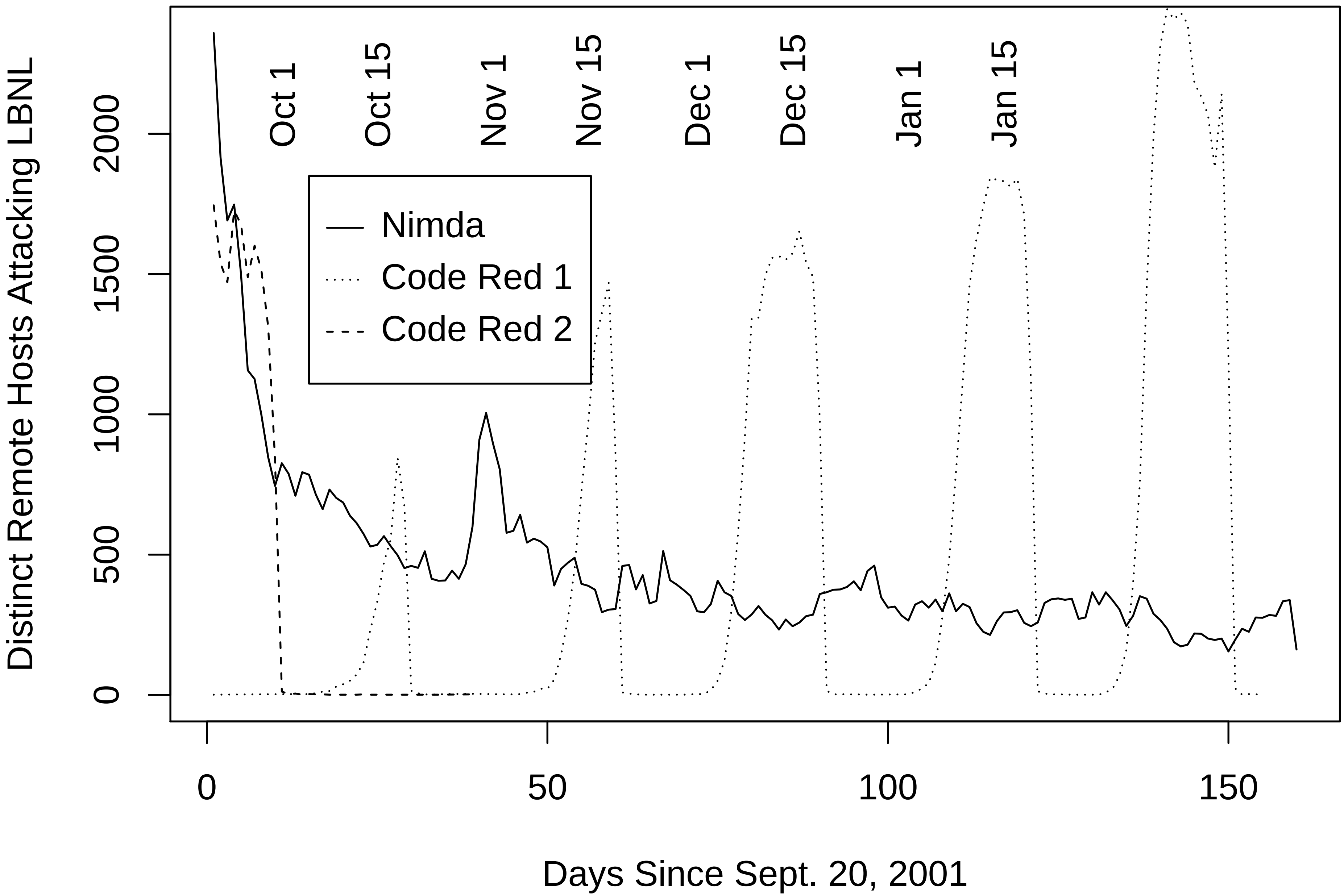
Payload: still unknown.


Onset of NIMDA
0
500000
1000000
2000000
3000000
Conn. / hour
10
15
20
25
30
35
Day Sept/Oct 2001


Onset of NIMDA
Conn / Sec
0
20
40
60
80
100
120
140
6:00
6:30
7:00
7:30
8:00
Time (PDT) 18Sep01



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
Code Red II
Nimda


Distinct Remote Hosts Attacking LBNL
Days Since Sept. 20, 2001
0
500
1000
1500
2000
0
50
100
150
Oct 1
Oct 15
Nov 1
Nov 15
Dec 1
Dec 15
Jan 1
Jan 15
Nimda
Code Red 1
Code Red 2

Spreading faster — distributed coordination (*Warhol* worms):

Idea: *reduce redundant scanning*.

Construct permutation of address space.

Each new worm instance starts at random point.

Worm instance that “encounters” another instance re-randomizes.

Idea: *reduce slow startup phase*.

Construct a “hit-list” of vulnerable servers in advance.

Then: for 1M vulnerable hosts, 10K hit-list, 100 scans/worm/sec,

1 sec to infect $\Rightarrow$ 99% infection in 5 minutes.

Spreading still faster — *Flash* worms:

Idea: use an *Internet-sized hit list*.

Where do you get it?

- *brute-force scanning* — entire addr. space 2hr w/ OC-12 (thanks for the cover, Code Red!)
- *distributed scanning* — use zombies (10 @ LBNL, 2001)
- *stealth scanning* — spread it over several months
- *DNS searches* — e.g., `www.domain.com`
- *spiders* — ask the search engines
- *just listen* — P2P, or exploit existing worms

Flash worms, con't:

Initial copy of the worm has the entire hit list.

Each generation, infects n from the list, gives each $1/n$ of list.

(Or, point them to a well-connected host that serves up portions of the list. Or a hybrid.)

How big is the list?

e.g., 9M addresses, sorted & differenced & gzip'd: 13 MB.

So dominant traffic is N copies of the payload.

Need to engineer for locality, failure & redundancy.

But: $n = 10$ requires ≈ 7 generations to infect 10^7 hosts.

$\Rightarrow$ <u>Tens of seconds.</u>

How can we defend against Internet-scale worms?

Time scales rule out human intervention.

$\Rightarrow$ Need automated detectors, response.

(And perhaps honeyputs to confuse scanning?)

Very hard research question!

And it's only half of the problem . . .

***Contagion* worms:**

Suppose you have two exploits:

E_s (Web server) and E_c (Web client).

You infect a server (or client) with E_s (E_c).

Then you . . . wait. (Perhaps you bait, e.g., host porn.)

When vulnerable client arrives, infect it.

You send over *both* E_s and E_c.

As client happens to visit other vulnerable servers $\Rightarrow$ infects.

Contagion worms, con't:

No change in communication patterns

other than slightly larger-than-usual transfers.

How do you detect this?

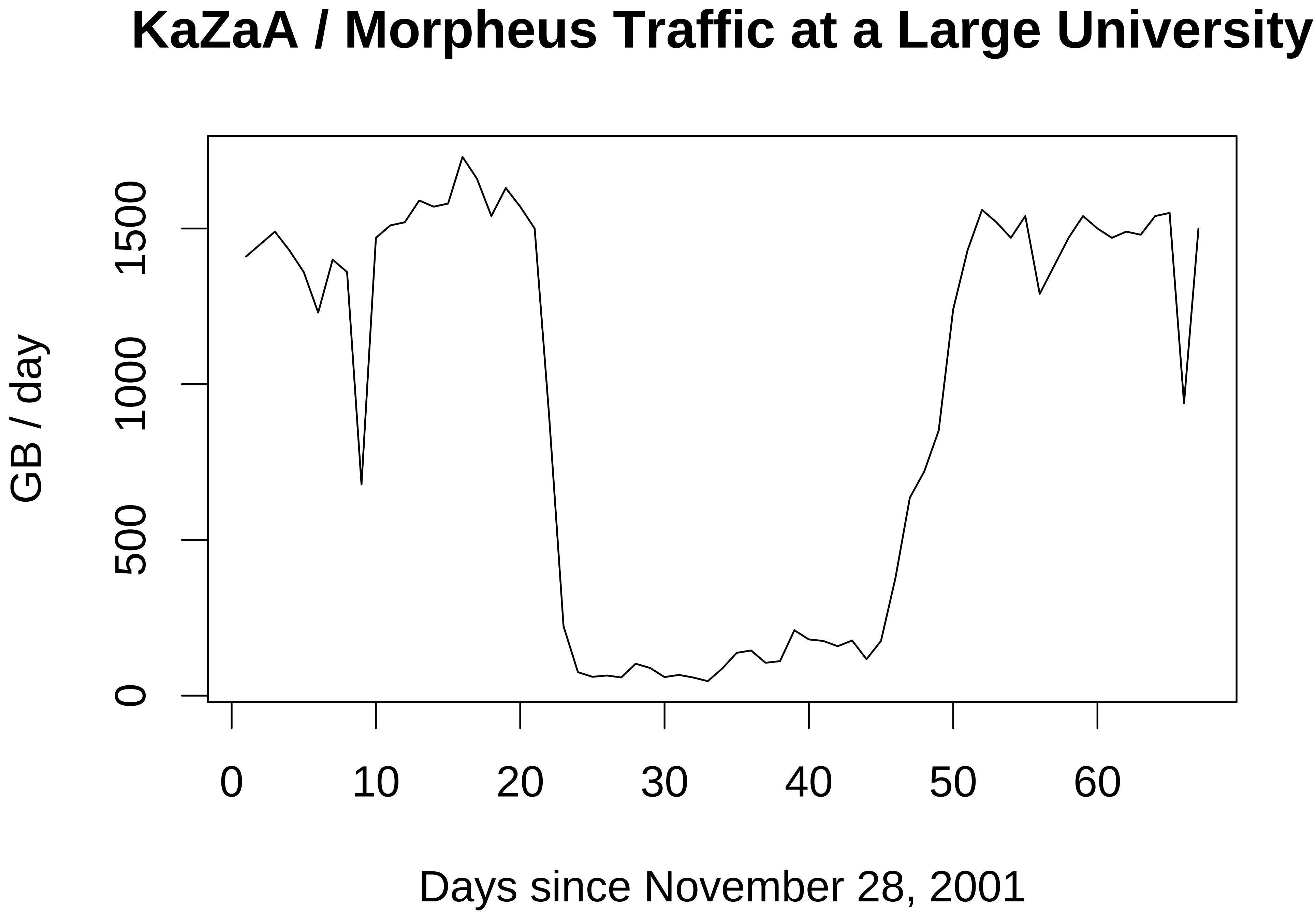
How bad can it be?

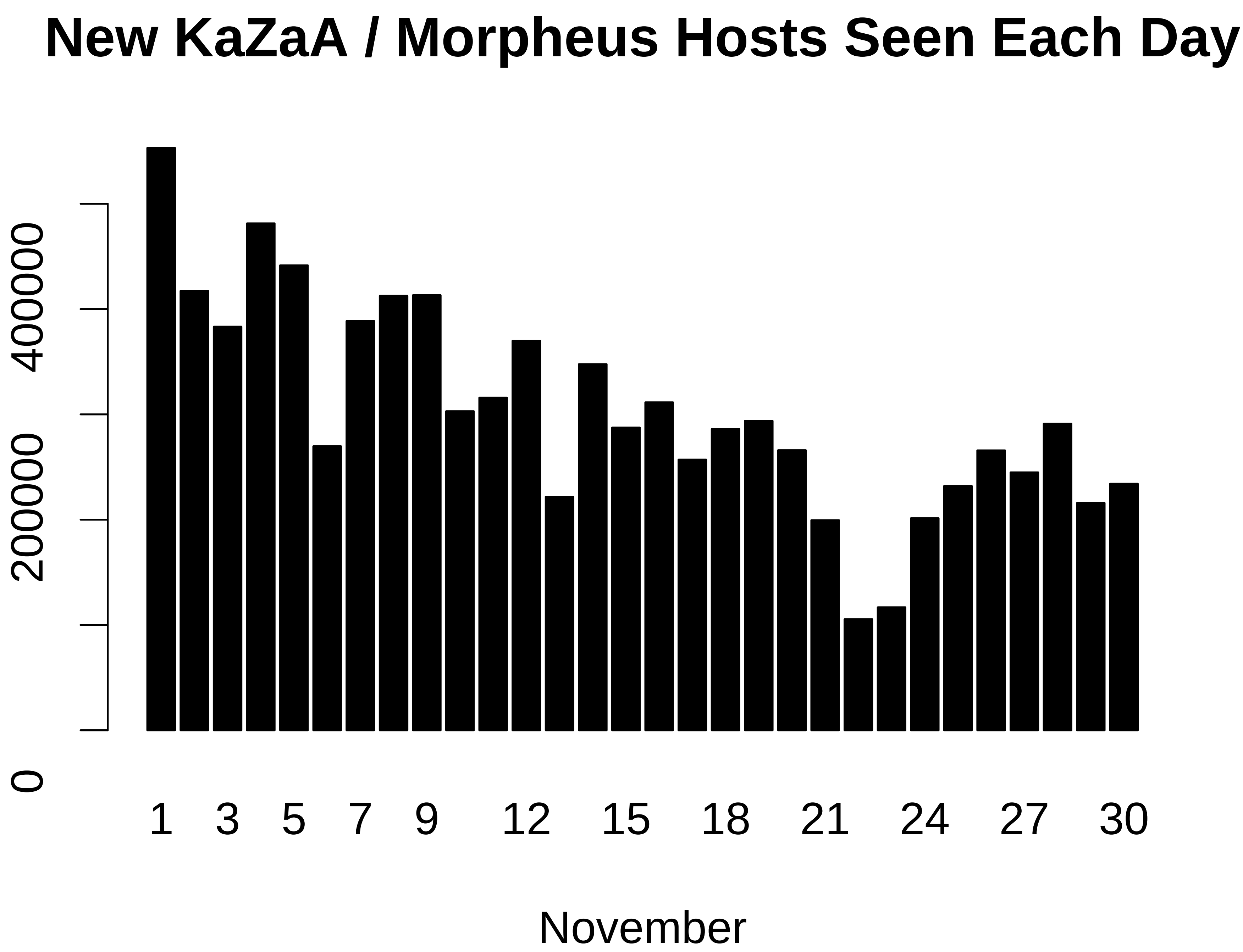
Exploiting Peer-to-Peer networks:

- Likely only need a single exploit, not a pair.
- Often, peers running *identical* software..
- Tend to have rich interconnection patterns to piggyback on.
- Often used to transfer large files.
- Not mainstream — less vulnerability assessment, monitoring.

Exploiting Peer-to-Peer networks, con’t:

- Often give access to user’s desktop rather than server.
- “Grey” content: users less likely to mention unusual activity.
- Come with built-in control / data dissemination plane.
- . . . and can be Very Large . . .


KaZaA / Morpheus Traffic at a Large University
GB / day
0
500
1000
1500
0
10
20
30
40
50
60
Days since November 28, 2001


New KaZaA / Morpheus Hosts Seen Each Day
0
2000000
4000000
1
3
5
7
9
12
15
18
21
24
27
30
November

The threat of contagion worms:

If you 0wn'd a single university, then last November . . .

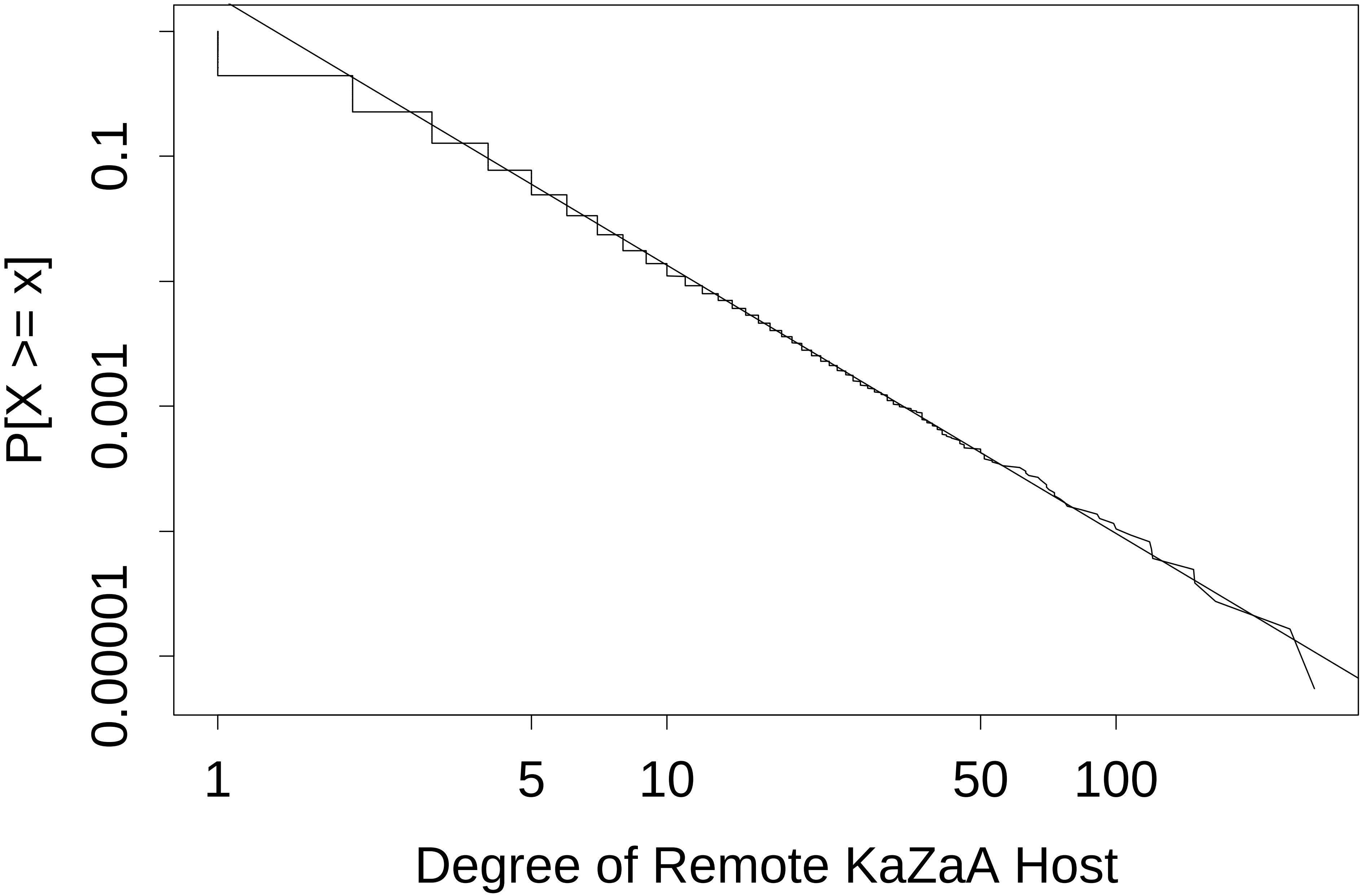
. . . you could have 0wn'd 9,127,468 additional hosts.

How fast?

Certainly, <u>much</u> faster than 1 month.

Degree of remote hosts as seen at Univ.: beautiful power law.

Epidemic Spreading in Scale-Free Networks (Phys. Rev. Letters Apr. 2001) $\Rightarrow$ this could be quite bad!



P[X >= x]
0.1
0.001
0.000001
1
5
10
50
100
Degree of Remote KaZaA Host

Envisioning a *Cyber Center for Disease Control*:

Identify outbreaks

Need decentralized communication mechanisms, multiple communication channels, diverse network of sensors.

Rapid pathogen analysis (how it spreads; what else it does)

Need on-call experts, state-of-the-art analysis tools, libraries of toolkit components, archive of previous worms, lab w/ virtual machines running popular OS's.

Useful even after the fact, esp. in "fog of war."

Envisioning a Cyber-Center for Disease Control, con't:

Fight infections

Mechanisms to propagate signatures out to body of *agents*.

Major issues over control, liability, resilience.

Anticipate new vectors

Track rise of new applications, analyze associated threat.

Envisioning a Cyber-Center for Disease Control, con’t:

Proactively devise and deploy detectors

E.g., develop KaZaA IDS plug-in.

Resist future threats

Vet applications for security soundness, foster research into resilient application design paradigms (that are somehow commercially viable).

The CDC sounds hopelessly hard.

Yet if a nation *(i)* takes the possibility of cyberwarfare seriously, and *(ii)* wants an open Internet . . .

. . . what’s the alternative?