Last time

We continued
Software Security

Getting sick with
Malware

- Types of malware
- How viruses work
- Detecting viruses (and counter-measures)

This time

Continuing with
Software Security

Going digging for
Worms
& virus case studies

- Virus case studies
- How worms work
- Modeling worm spread rates

Virus case studies

Brain

First IBM PC virus (1987)

- Propagation method
  - Copies itself into the boot sector
  - Tells the OS that all of the boot sector is “faulty” (so that it won’t list contents to the user)
    - Thus also one of the first examples of a **stealth** virus
  - Intercepts disk read requests for 5.25” floppy drives
    - Sees if the 5th and 6th bytes of the boot sector are 0x1234
    - If so, then it’s already infected, otherwise, infect it

- Payload:
  - Nothing really; goal was just to spread (to show off?)
  - However, it served as the template for future viruses

PC Tools Deluxe 04.22 — Disk View/Edit Service

Path=A:

Absolute sector 0000000, System BOOT

Displacement	Hex codes																ASCII value
0000K(0000)	FA	E9	4A	01	34	12	00	07	14	00	01	00	00	00	00	20	·ßJ04‡ ♦Ґ @
0016K(0010)	20	20	20	20	20	20	57	65	6C	63	6F	6D	65	20	74	6F	Welcome to
0032K(0020)	20	74	68	65	20	44	75	6E	67	65	6F	6E	20	20	20	20	the Dungeon
0048K(0030)	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
0064K(0040)	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
0080K(0050)	20	28	63	29	20	31	39	38	36	20	42	61	73	69	74	20	(c) 1986 Basit
0096K(0060)	26	20	41	6D	6A	61	64	20	28	70	76	74	29	20	4C	74	& Amjad (pvt) Lt
0112K(0070)	64	2E	20	20	20	20	20	20	20	20	20	20	20	20	20	20	d.
0128K(0080)	20	42	52	41	49	4E	20	43	4F	4D	50	55	54	45	52	20	BRAIN COMPUTER
0144K(0090)	53	45	52	56	49	43	45	53	2E	2E	37	33	30	20	4E	49	SERVICES..730 NI
0160K(00A0)	5A	41	4D	20	42	4C	4F	43	4B	20	41	4C	4C	41	4D	41	ZAM BLOCK ALLAMA
0176K(00B0)	20	49	51	42	41	4C	20	54	4F	57	4E	20	20	20	20	20	IQBAL TOWN
0192K(00C0)	20	20	20	20	20	20	20	20	20	20	20	4C	41	48	4F	52	LAHOR
0208K(00D0)	45	2D	50	41	4B	49	53	54	41	4E	2E	2E	50	48	4F	4E	E-PAKISTAN..PHON
0224K(00E0)	45	20	3A	34	33	30	37	39	31	2C	34	34	33	32	34	38	E :430791,44324
0240K(00F0)	2C	32	38	30	35	33	30	2E	20	20	20	20	20	20	20	20	9,280530.

Home=beg of file/disk End=end of file/disk
ESC=Exit PgDn=forward PgUp=back F2=chg sector num F3=edit F4=get name

Rootkits

- Recall: a rootkit is malicious code that takes steps to go undiscovered
  - By intercepting system calls, patching the kernel, etc.
  - Often effectively done by a man in the middle attack

- Rootkit revealer: analyzes the disk offline and through the online system calls, and compares

- Mark Russinovich ran a rootkit revealer and found a rootkit in 2005…

Rootkits

- Recall: a rootkit is malicious code that takes steps to go undiscovered
  - By intercepting system calls, patching the kernel, etc.
  - Often effectively done by a man in the middle attack

- Rootkit revealer: analyzes the disk offline and through the online system calls, and compares

- Mark Russinovich ran a rootkit revealer and found a rootkit in 2005… **installed by a CD he had bought.**

Sony XCP rootkit

Detected 2005

Sony XCP rootkit

Detected 2005

- Goal: keep users from copying copyrighted material

Sony XCP rootkit

Detected 2005

- Goal: keep users from copying copyrighted material

- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - Intercepted read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)

Sony XCP rootkit

Detected 2005

- Goal: keep users from copying copyrighted material

- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - Intercepted read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)

- How it messed up
  - Morally: violated trust
  - Technically: Hid ***all files*** that started with "sys"
  - Seriously?: The uninstaller did not check the integrity of the code it downloaded, and would not delete it afterwords.

Stuxnet: Propagation

June 2010

- Virus in that it initially spread by infected USB stick
  - Once inside a network, it acted as a worm, spreading quickly

- Exploited **four** zero-day exploits
  - Zero-day: Known to only the attacker until the attack
  - Typically, one zero-day is enough to profit
  - Four was unprecedented
    - Immense cost and sophistication on behalf of the attacker

- Rootkit: installed *signed* device drivers
  - Thereby avoiding user alert when installing
  - Signed with certificates stolen from two Taiwanese CAs

Stuxnet: Payload

Stuxnet: Payload

- Do nothing

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge

Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge
  - .. all the while sending “looks good to me” readings to the user

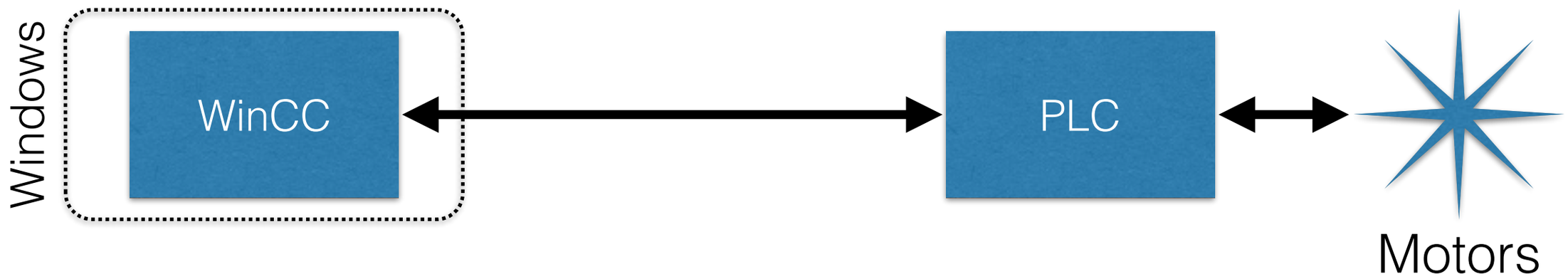
Stuxnet: Payload

- Do nothing
- Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - You know, like those in Iran and Finland
  - .. those ones that are used to operate centrifuges
  - .. for producing enriched uranium for nuclear weapons
- In which case, slowly increase the freq to 1410Hz
  - You know, enough to break the centrifuge
  - .. all the while sending “looks good to me” readings to the user
  - .. then drop back to normal range

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

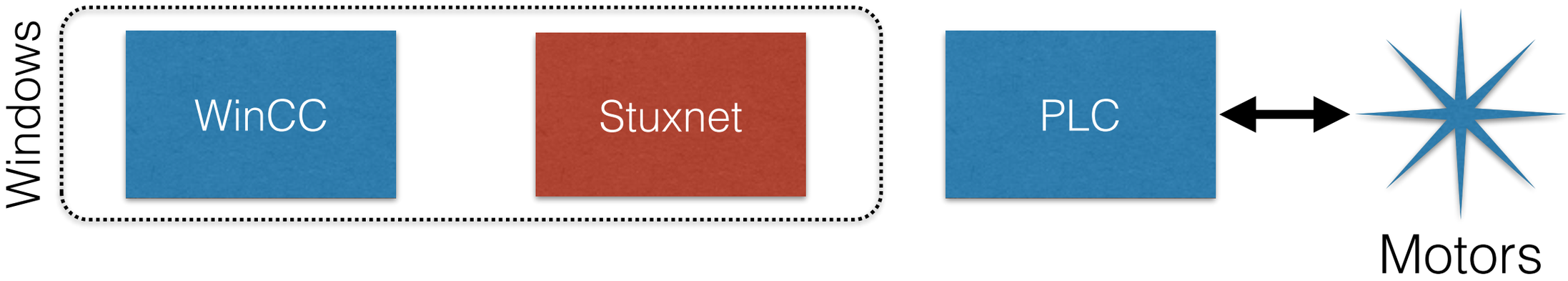
- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



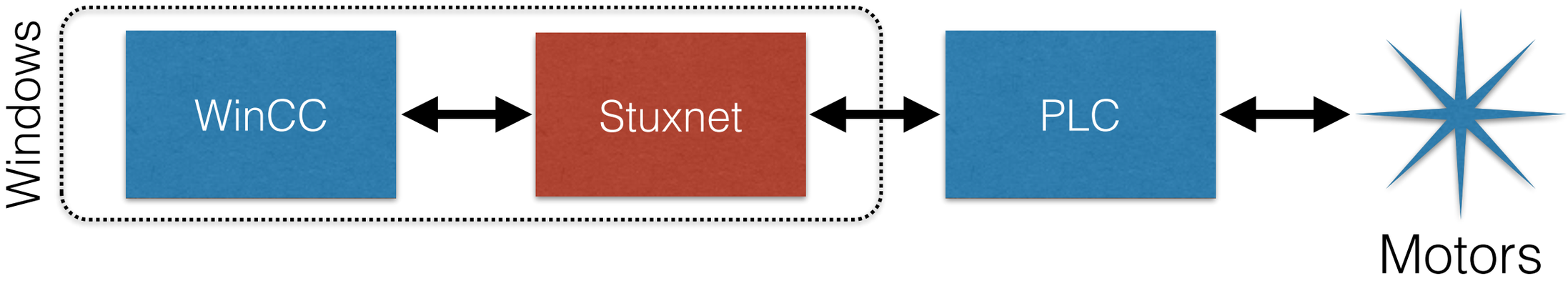
- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

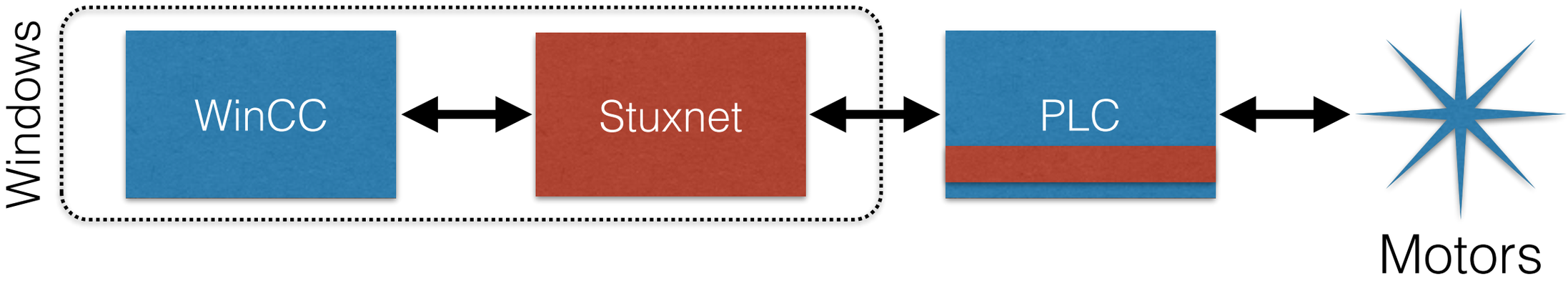


- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards
- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator

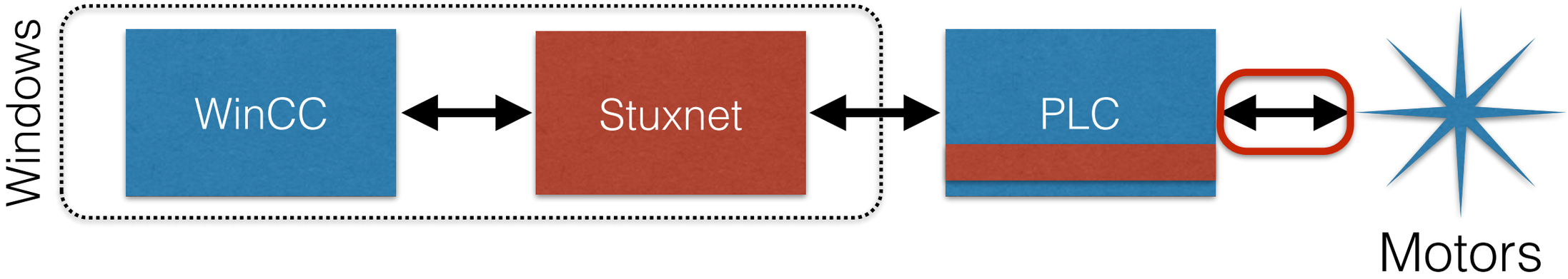


- In reality, it sped up and slowed down the motors
- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



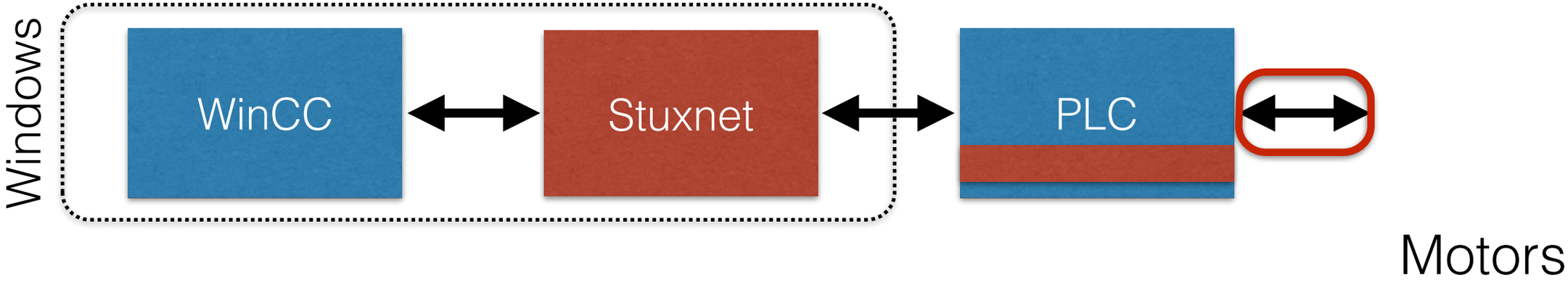
- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Payload

- Targets industrial control systems by overwriting programmable logic boards

- Man-in-the-middle between Windows and Siemens control systems; looked like it was working properly to the operator



- In reality, it sped up and slowed down the motors

- Result: Destroy (or at least decrease the productivity of) nuclear centrifuges

Stuxnet: Fallout

- Iran denied they had been hit by Stuxnet
- Then claimed they were, but had contained it
- Understood now that it took out 1k of Iran's 5k centrifuges
- Security experts believe the U.S. did it (possibly along with Israel) due to its sophistication and cost
- **Legitimized cyber warfare**

Viruses summary

- Technological arms race between those who wish to detect and those who wish to evade detection

- Started off innocuously, capable by only a few very clever people

- But viruses have become commoditized; any scriptkiddy can launch one (creation remains hard)

- No longer purely of academic interest
  - Economic pursuits (zero-day markets)
  - Cyber warfare

Worms

Controlling millions of hosts: ***Why?***

- Distributed Denial of Service (DDoS)
  - Generate network traffic from many sources..
  - .. to a single destination
  - .. with the intention of overloading their network
    - Consume too many resources for legitimate users to also use

- Steal sensitive information from millions of others
  - Even a small fraction of unprotected people ⇒ $

- Confuse and disrupt

Controlling millions of hosts: ***How?***

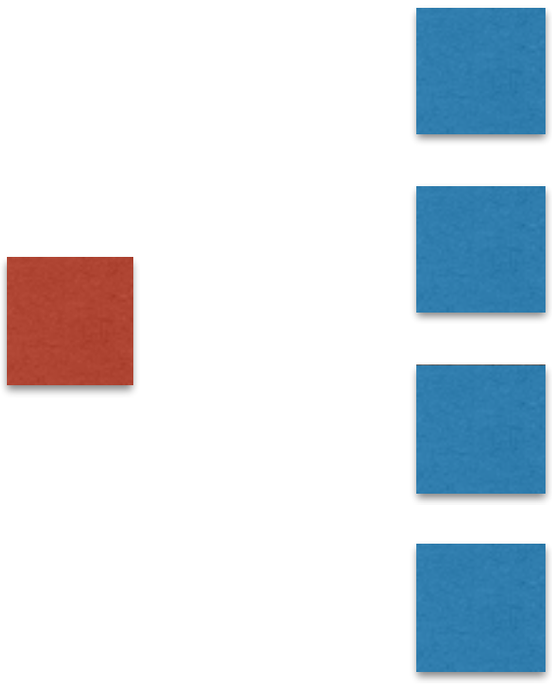
- Worm: *self*-propagates by arranging to have itself *immediately* executed
  - At which point it creates a new, additional instance of itself

- Typically infects by altering *running* code
  - No user intervention required

- Like viruses, propagation and payload are orthogonal

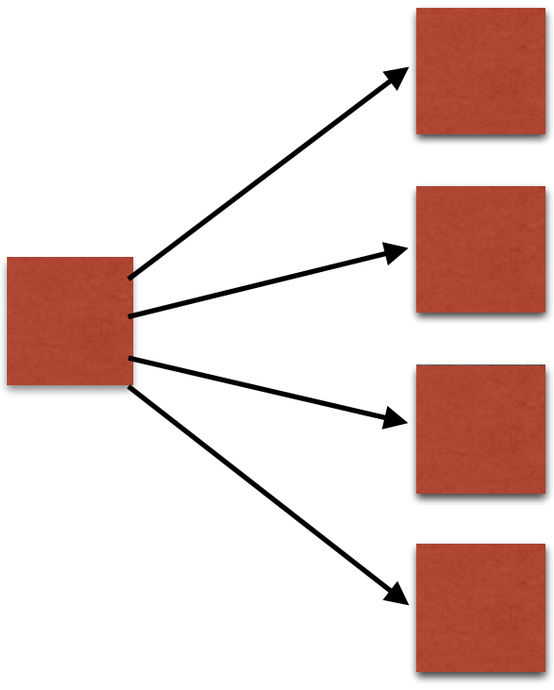
Self propagation

- The goal is to spread as quickly as possible
- The key is ***parallelization***
  - Ditto for viruses, but they require human interaction to trigger each propagation



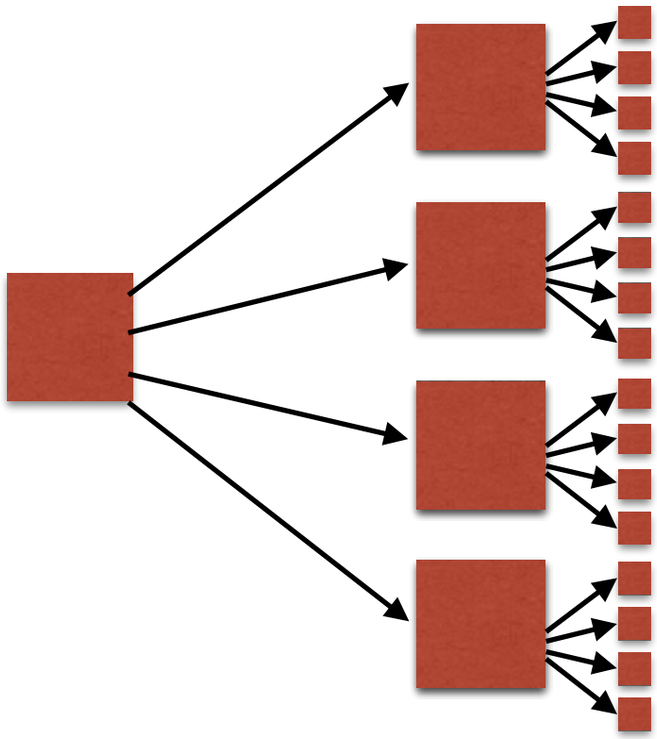
Self propagation

- The goal is to spread as quickly as possible
- The key is ***parallelization***
  - Ditto for viruses, but they require human interaction to trigger each propagation



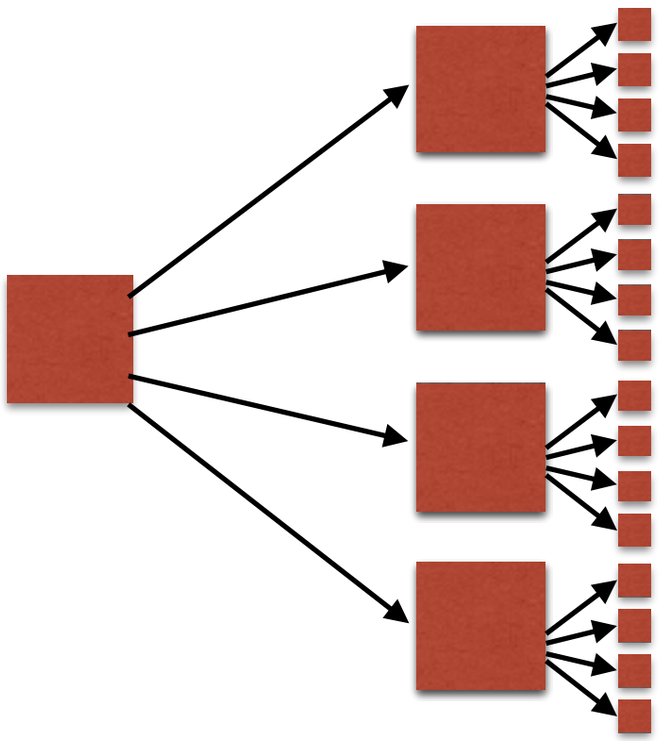
Self propagation

- The goal is to spread as quickly as possible
- The key is ***parallelization***
  - Ditto for viruses, but they require human interaction to trigger each propagation



Self propagation

- The goal is to spread as quickly as possible
- The key is ***parallelization***
  - Ditto for viruses, but they require human interaction to trigger each propagation



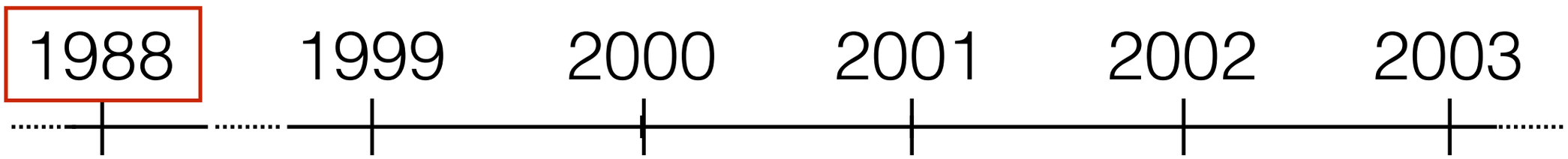
Propagation

(1) Targeting: how does the worm **find new prospective victims**?

(2) Exploit: how does the worm get code to **automatically run**?

A brief history

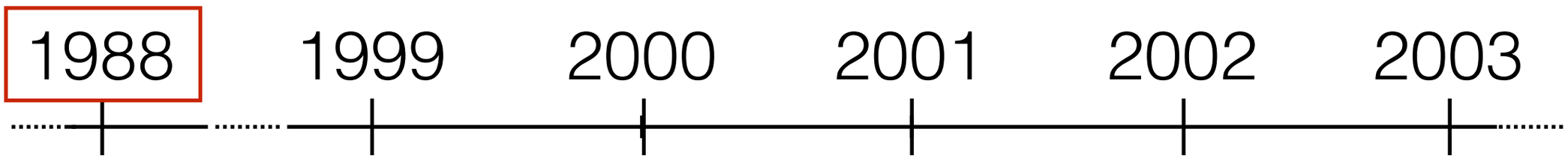
First arrival



- Morris worm
  - Propagated across machines (too aggressively, thanks to a bug)
  - One way it propagated was a **buffer overflow attack** against a vulnerable version of `fingerd` on VAXes
    - Sent a special string to the finger daemon, which caused it to execute code that created a new worm copy
    - Didn't check OS: caused Suns running BSD to crash
  - End result: $10-100M in damages, probation, community service

A brief history

First arrival

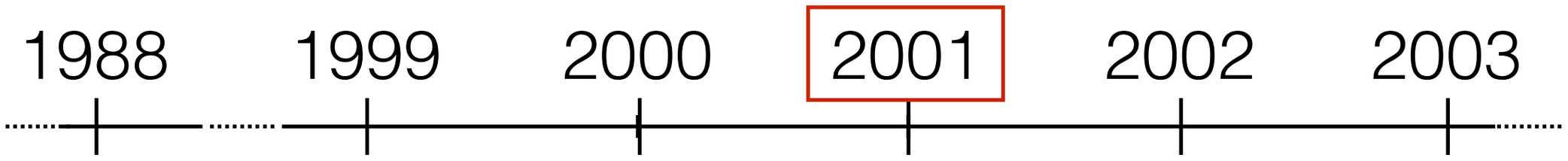


- Morris worm
  - Propagated across machines (too aggressively, thanks to a bug)
  - One way it propagated was a **buffer overflow attack** against a vulnerable version of `fingerd` on VAXes
    - Sent a special string to the finger daemon, which caused it to execute code that created a new worm copy
    - Didn't check OS: caused Suns running BSD to crash
  - End result: $10-100M in damages, probation, community service

Robert Morris is now a professor at MIT

A brief history

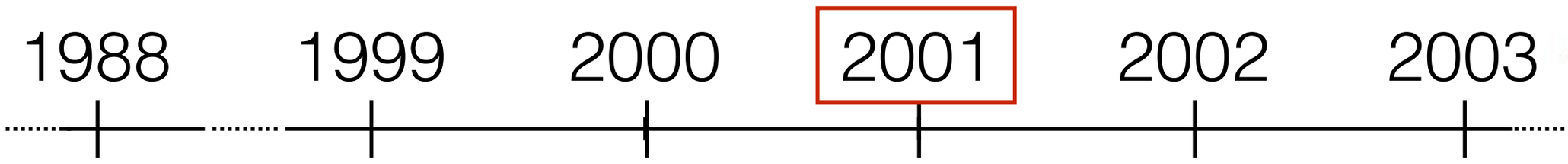
Introduction of “modern day” worms



- CodeRed
  - Propagation: Exploited an overflow in the MS-IIS server
  - Payload 1: website defacement
    - HELLO! Welcome to http://www.worm.com
      Hacked By Chinese!
  - Payload 2: time bomb
    - Day of month 1-20: **Spread**
    - Day of month 20+: **Attack** (flood 198.137.240.91 = whitehouse.gov)
  - 300,000 machines infected in 14 hours

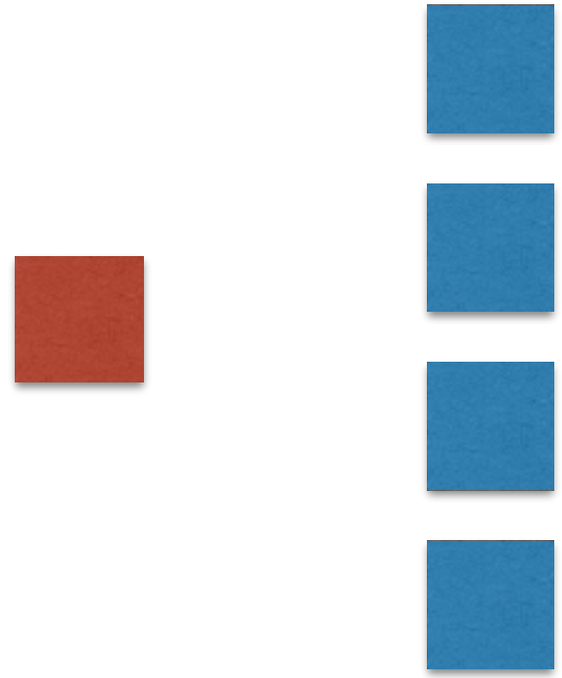
A brief history

Introduction of “modern day” worms



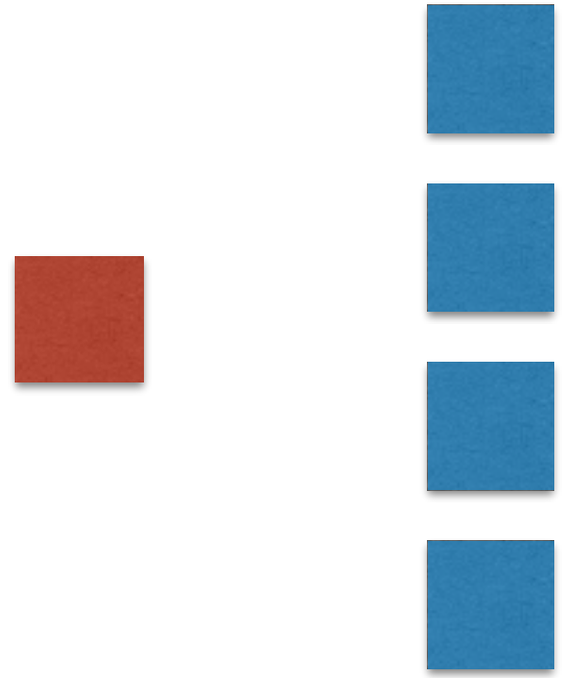
- CodeRed
  - Propagation: Exploited an overflow in the MS-IIS server
  - Payload 1: website defacement
    - HELLO! Welcome to http://www.worm.com
      Hacked By Chinese!
  - Payload 2: time bomb
    - Day of month 1-20: **Spread**
    - Day of month 20+: **Attack** (flood 198.137.240.91 = whitehouse.gov)
  - 300,000 machines infected in 14 hours

CodeRed Propagation (at a high level)



CodeRed Propagation (at a high level)

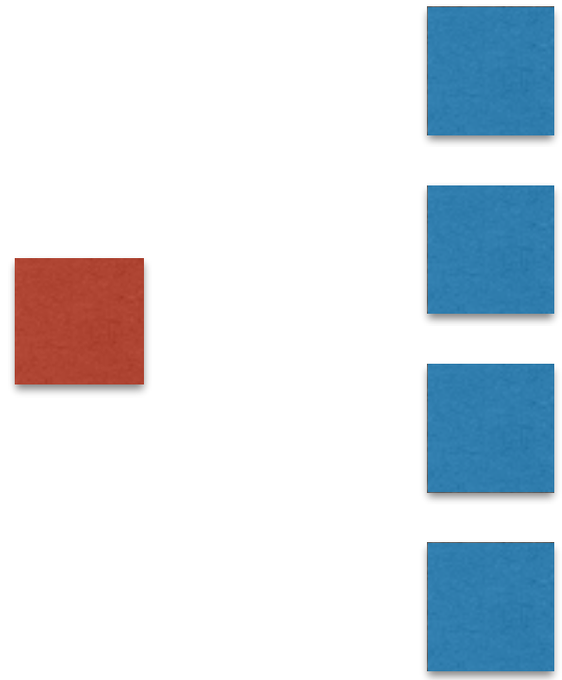
- Goal: spread your virus as widely and as quickly as possible



CodeRed Propagation (at a high level)

- Goal: spread your virus as widely and as quickly as possible

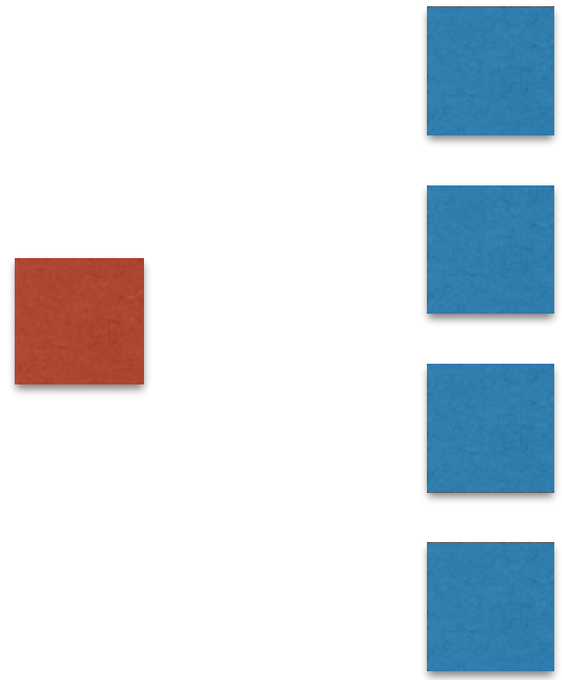
- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)



CodeRed Propagation (at a high level)

- Goal: spread your virus as widely and as quickly as possible

- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)
  - spread(code, state, IP addr)
    copies over the code (plus whatever extra state) to the IP address, and executes it



CodeRed Propagation (at a high level)

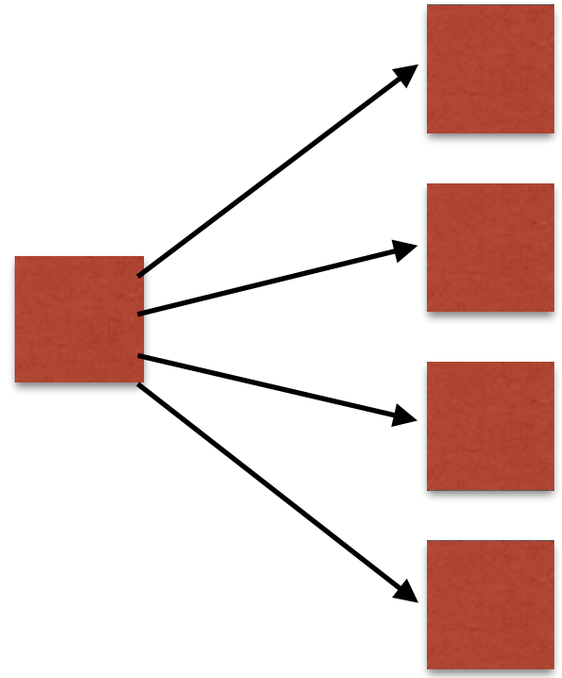
- Goal: spread your virus as widely and as quickly as possible

- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)
  - spread(code, state, IP addr)
    copies over the code (plus whatever extra state) to the IP address, and executes it

CodeRed Propagation (at a high level)

- Goal: spread your virus as widely and as quickly as possible

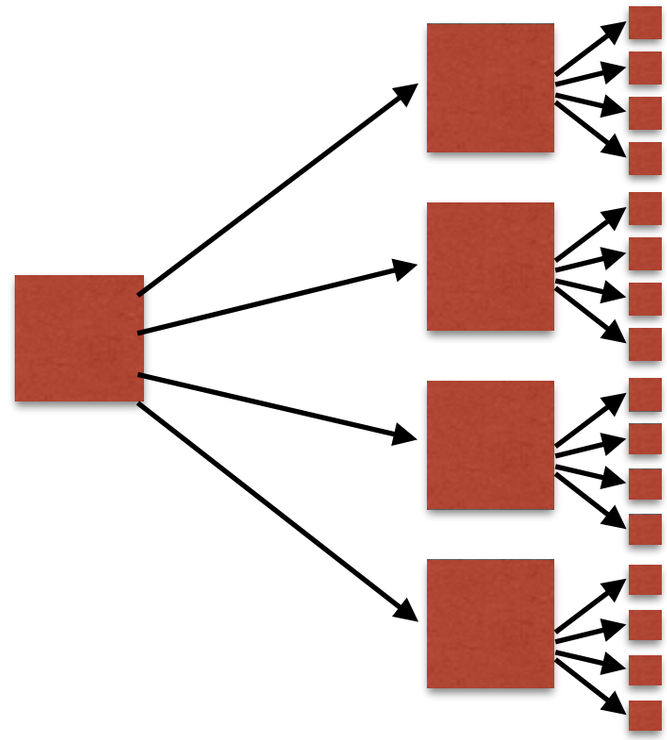
- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)
  - spread(code, state, IP addr)
    copies over the code (plus whatever extra state) to the IP address, and executes it



CodeRed Propagation (at a high level)

- Goal: spread your virus as widely and as quickly as possible

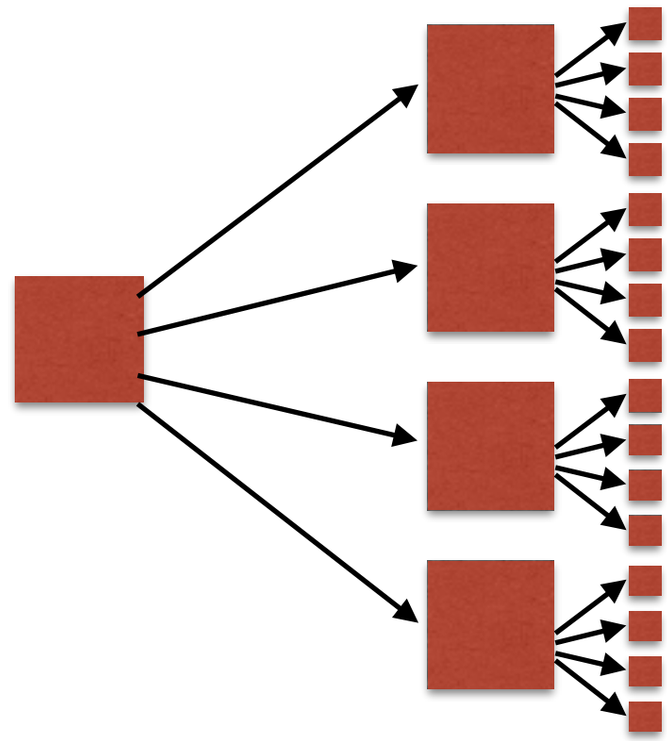
- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)
  - spread(code, state, IP addr)
    copies over the code (plus whatever extra state) to the IP address, and executes it



CodeRed Propagation (at a high level)

- Goal: spread your virus as widely and as quickly as possible

- Mechanisms:
  - attack(IP address)
    tries to connect to and exploit MS-IIS (if it happens to be running on that IP address)
  - spread(code, state, IP addr)
    copies over the code (plus whatever extra state) to the IP address, and executes it



- Question: What IP addresses do you choose? What do your “children” choose?

CodeRed Propagation (how it worked)

- Spread by randomly scanning the entire 32-bit IP address space
  - Pick a pseudorandom 32-bit number = IP addr
  - Try attack(IP addr)
  - Repeat
- This is a very common *but not fundamental* worm technique
- Each instance of the worm used the *same random number seed*

CodeRed Propagation (how it worked)

- Spread by randomly scanning the entire 32-bit IP address space
  - Pick a pseudorandom 32-bit number = IP addr
  - Try attack(IP addr)
  - Repeat

- This is a very common *but not fundamental* worm technique

- Each instance of the worm used the *same random number seed*

Linear

CodeRed Propagation (how it worked)

- Spread by randomly scanning the entire 32-bit IP address space
  - Pick a pseudorandom 32-bit number = IP addr
  - Try attack(IP addr)
  - Repeat

- This is a very common *but not fundamental* worm technique

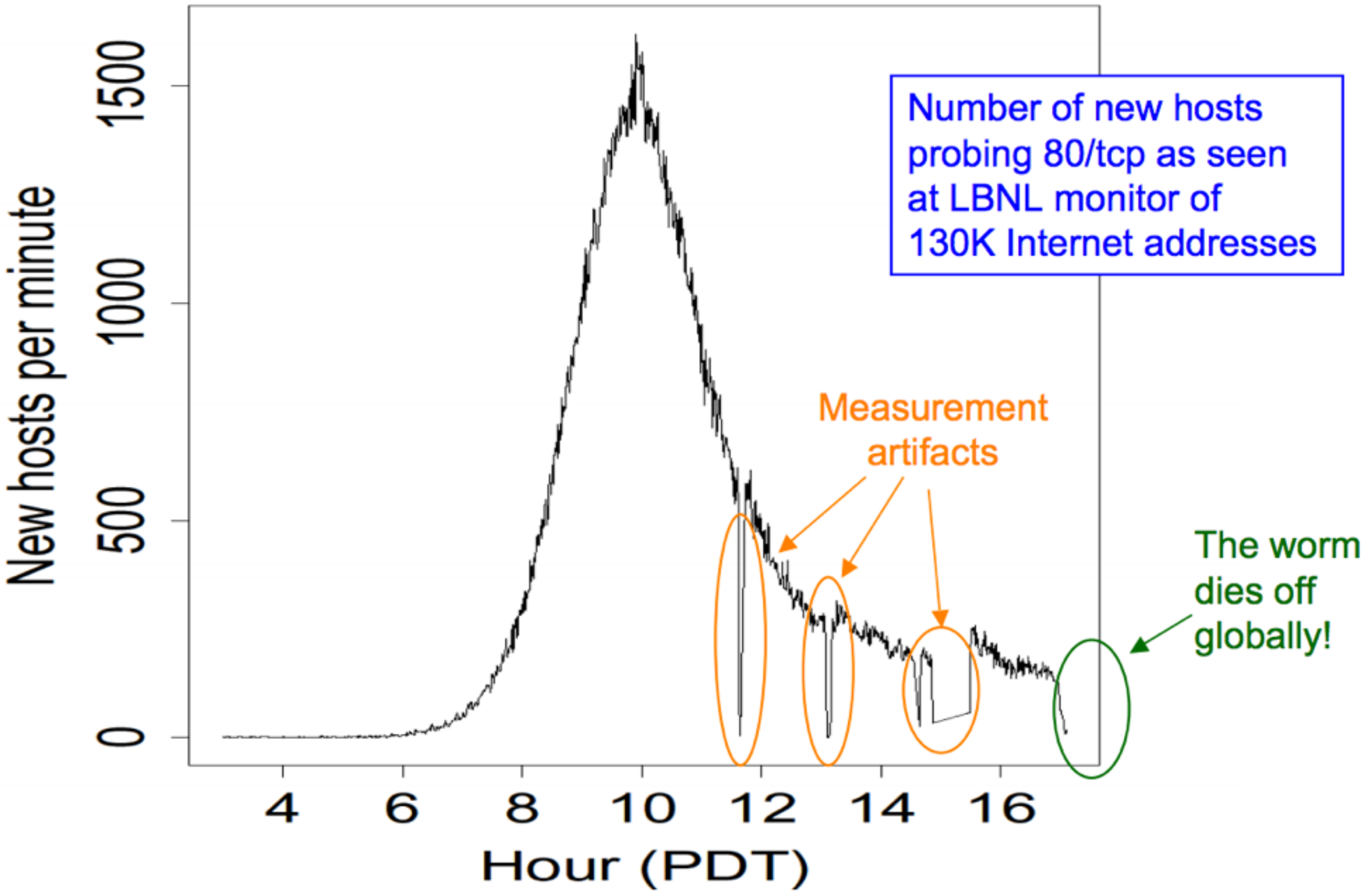
- Each instance of the worm used the *same random number seed*

What would the growth over time be? *Linear*

CodeRed Revision

- Revision released one week later (July 19, 2001)
- Whitehouse.gov **changed** its IP address
  - This caused CodeRed to **die** for date ≥20th of the month
    - .. Author didn't test the code: it was ***buggy***!
- But with this revision, the random number generator was **seeded** properly!

Growth of Code Red Worm



Credit: Vern Paxson's CS 161 at Berkeley

Modeling Worm Spread

- Worm spread is well described as *infectious epidemic*
  - Classic "SI" model (Susceptible-Infectible)

- Model parameters:
  - N: Population size
  - S(t): # Susceptible hosts at time t
  - I(t): # Infected hosts at time t
  - β: contact rate
    - How many population members each infected host communicates with per unit time
    - E.g., if each infected host scans 10 IP addresses per unit time, and 2% of all IP addresses run a vulnerable server, then β = 0.2

$$N = S(t) + I(t)$$
$$S(0) = I(0) = N/2$$

$$i(t) = I(t) / N = \text{fraction of hosts infected}$$

Computing How an Epidemic Progresses

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Computing How an Epidemic Progresses

Increase in
infectibles
per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Total attempted contacts per unit time

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Fraction of contacts expected to succeed

Total attempted contacts per unit time

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Fraction of contacts expected to succeed

Total attempted contacts per unit time

Rewriting using i(t) = I(t) / N and S = N - I:

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Fraction of contacts expected to succeed

Total attempted contacts per unit time

Rewriting using i(t) = I(t) / N and S = N - I:

$$\frac{di}{dt} = \beta \cdot i \cdot (1-i)$$

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Fraction of contacts expected to succeed

Total attempted contacts per unit time

Rewriting using $i(t) = I(t) / N$ and $S = N - I$:

$$\frac{di}{dt} = \beta \cdot i \cdot (1-i) \quad \Rightarrow \quad i(t) = \frac{e^{\beta t}}{1+e^{\beta t}}$$

Computing How an Epidemic Progresses

Increase in # infectibles per unit time

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Fraction of contacts expected to succeed

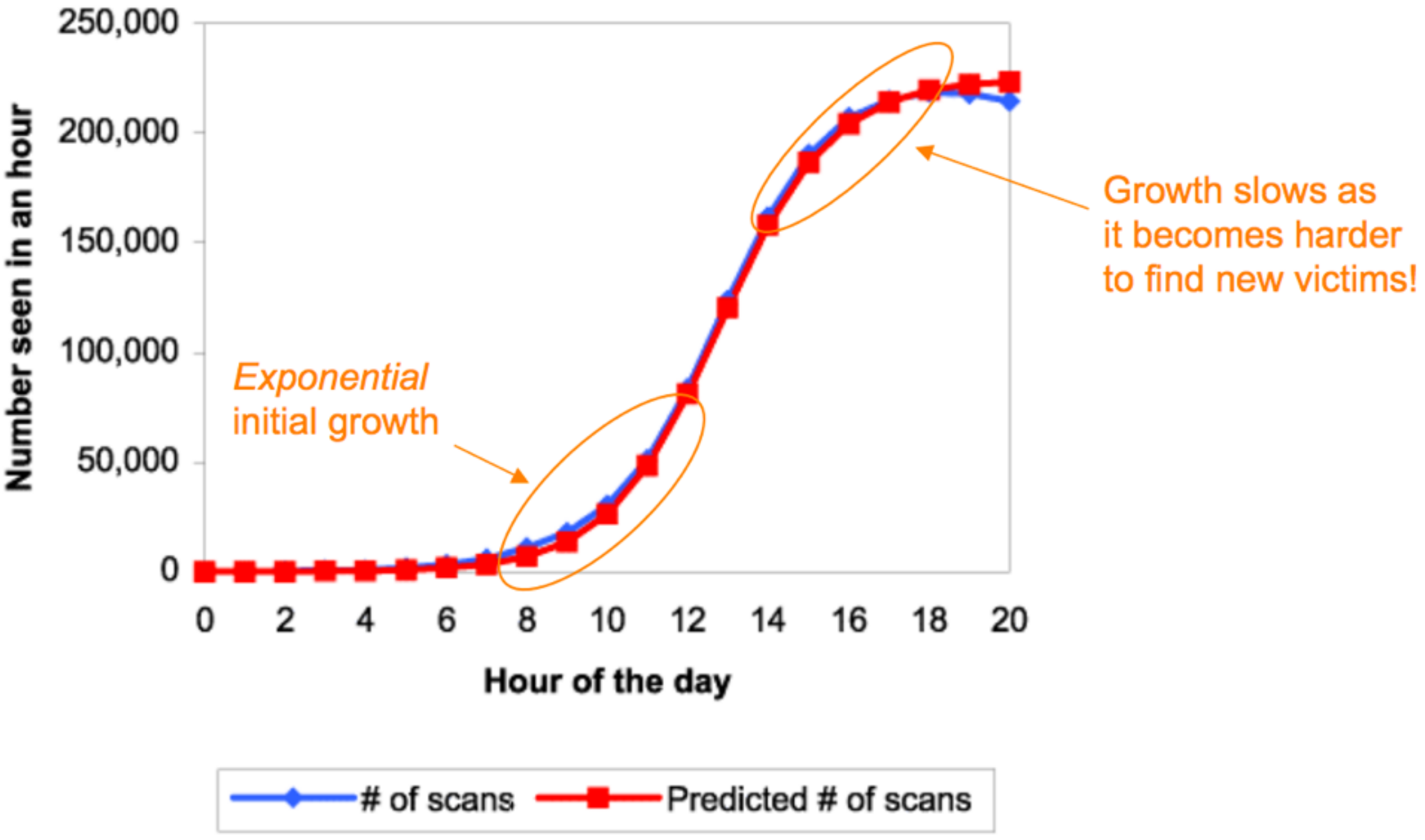
Total attempted contacts per unit time

Rewriting using $i(t) = I(t) / N$ and $S = N - I$:

$$\frac{di}{dt} = \beta \cdot i \cdot (1-i) \quad \Rightarrow \quad i(t) = \frac{e^{\beta t}}{1+e^{\beta t}}$$

Fraction infected grows as a *logistic*

Fitting the model to Code Red



Credit: Vern Paxson's CS 161 at Berkeley

Code Red Spread

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Code Red Spread

- Note that # of new infections scales with contact rate β

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

Code Red Spread

- Note that # of new infections scales with contact rate β

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

- For a scanning worm, β increases with N
  - Larger populations are infected more quickly!
  - More likely that a given scan finds a population member

Code Red Spread

- Note that # of new infections scales with contact rate β

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

- For a scanning worm, β increases with N
  - Larger populations are infected more quickly!
  - More likely that a given scan finds a population member

- Large-scale monitoring found 360K systems were infected with Code Red on July 19
  - Within 13 hours

Code Red Spread

- Note that # of new infections scales with contact rate β

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

- For a scanning worm, β increases with N
  - Larger populations are infected more quickly!
  - More likely that a given scan finds a population member

- Large-scale monitoring found 360K systems were infected with Code Red on July 19
  - Within 13 hours

- That night (the 20th) the worm died due to bug

Code Red Spread

- Note that # of new infections scales with contact rate β

$$\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$$

- For a scanning worm, β increases with N
  - Larger populations are infected more quickly!
  - More likely that a given scan finds a population member

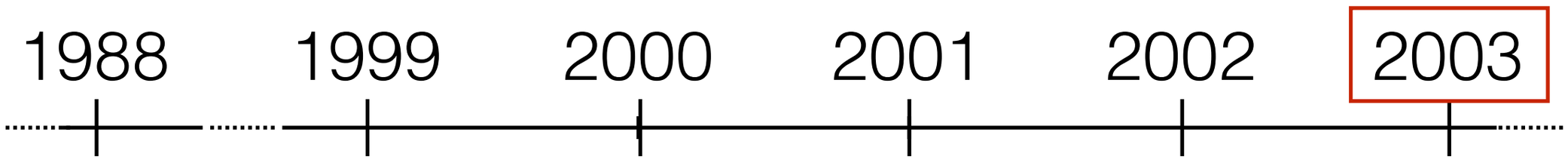
- Large-scale monitoring found 360K systems were infected with Code Red on July 19
  - Within 13 hours

- That night (the 20th) the worm died due to bug

- Successfully managed to restart itself Aug 1
  - … and each successive month for years to come

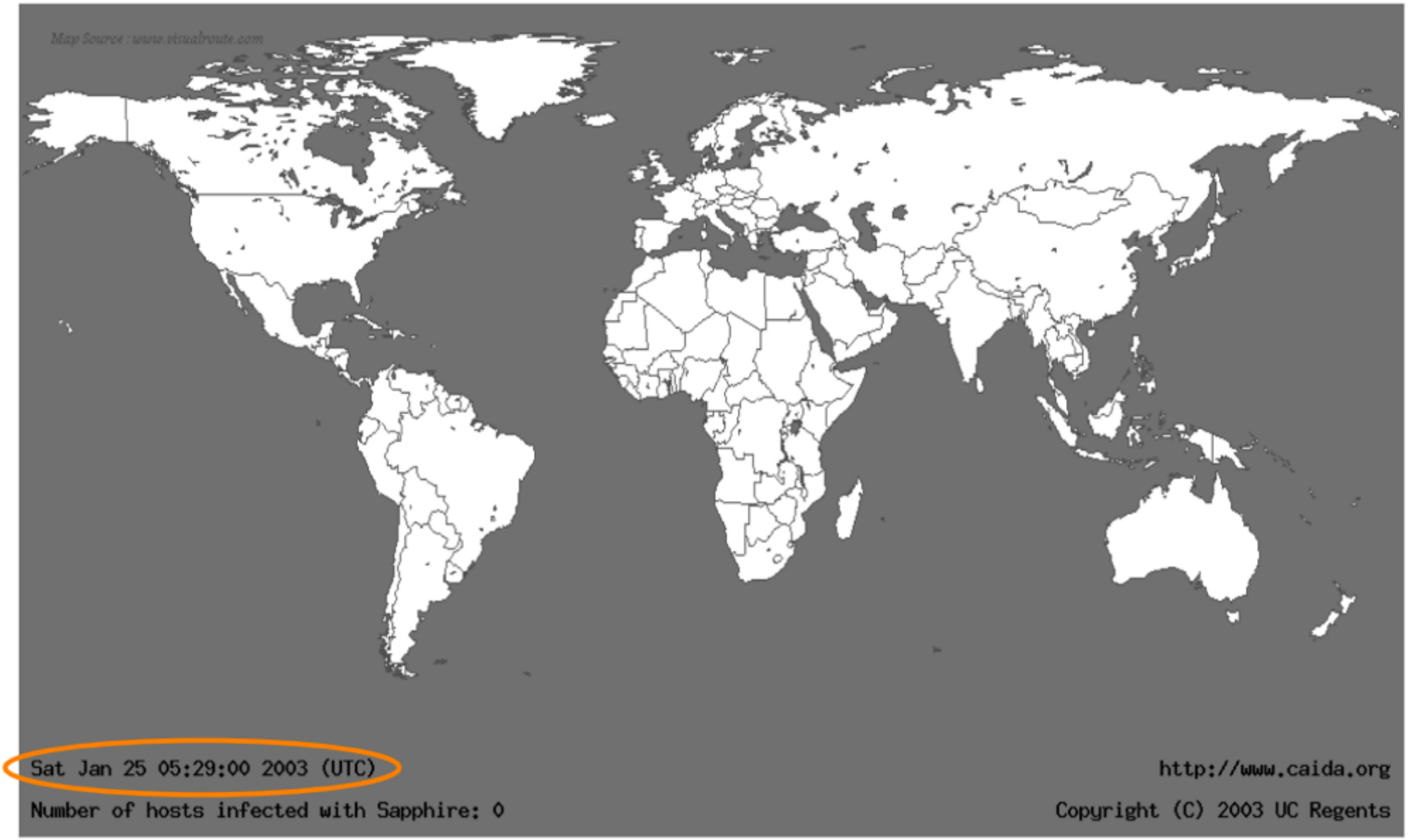
A brief history

The harm can be substantial



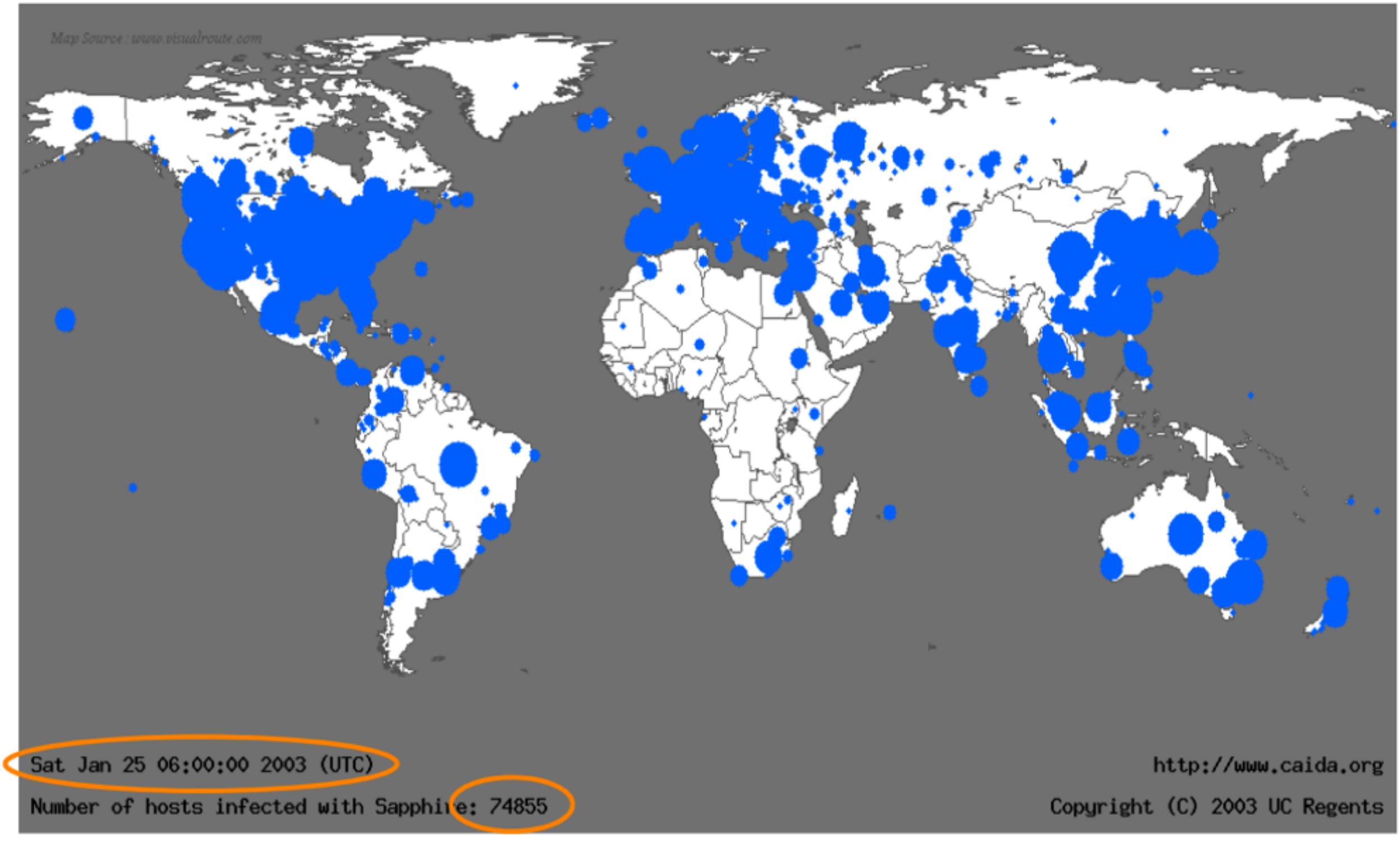
- SQL Slammer
  - Exploited an overflow in the MS-SQL server
  - 75,000 machines infected in 10 *minutes*

Life just before Slammer



Credit: Vern Paxson's CS 161 at Berkeley

Life just after Slammer

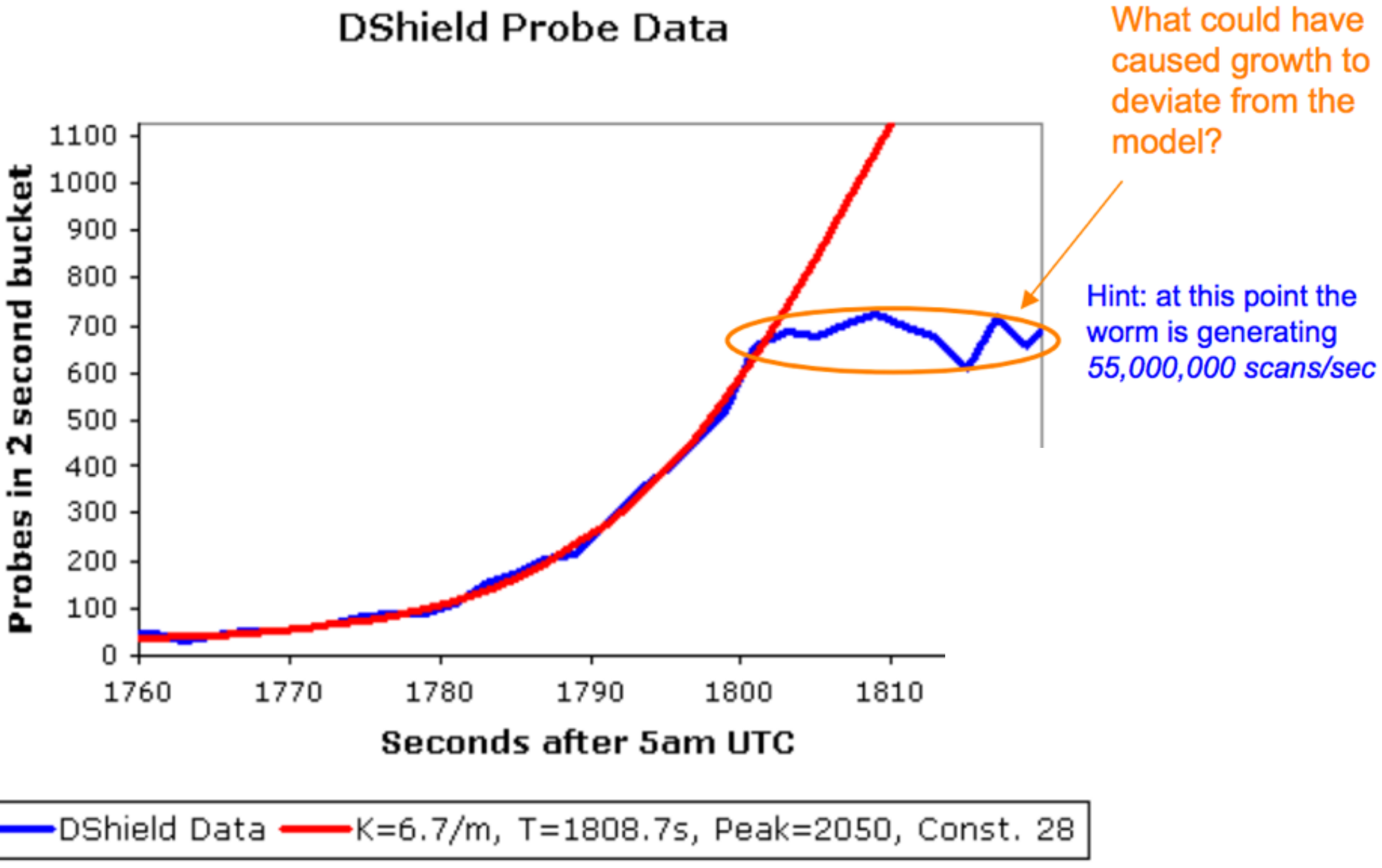


Credit: Vern Paxson's CS 161 at Berkeley

Slammer propagation

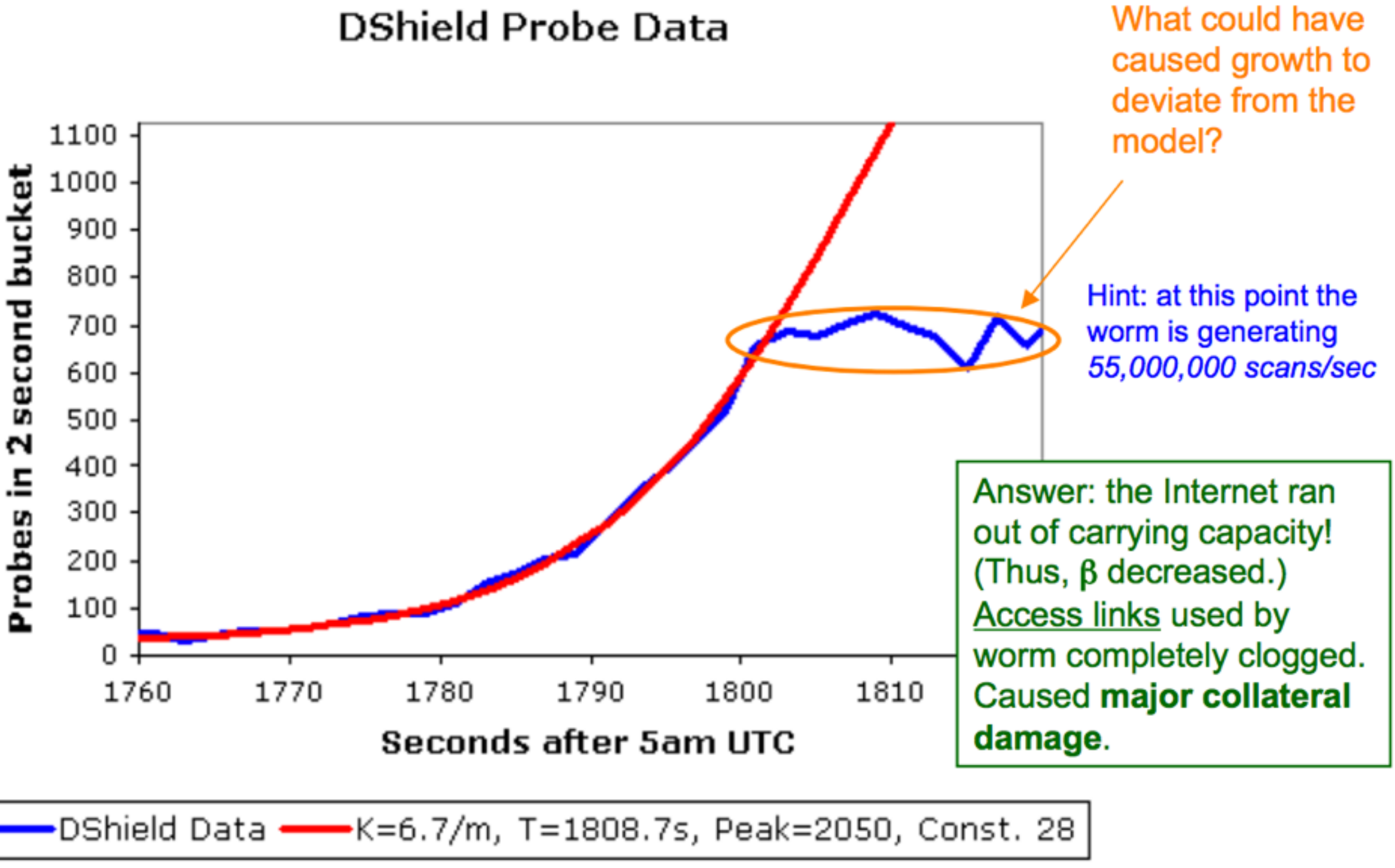
- Slammer exploited connectionless UDP service rather than connection-oriented TCP
- Entire worm fit in a single packet!
- When scanning, the worm could “fire and forget”
  - Stateless!
- Infected 75k+ hosts in < 10 minutes
- At its peak, doubled ever 8.5 seconds

Slammer's growth



Credit: Vern Paxson's CS 161 at Berkeley

Slammer's growth



Credit: Vern Paxson's CS 161 at Berkeley

This time

Continuing with
Software Security

Going digging for
Worms
& virus case studies

- Virus case studies
- How worms work
- Modeling worm spread rates

<u>Required reading for this time:</u>

"How to 0wn the Internet in your Spare Time"

Next time

Continuing with
Software Security

Getting insane with
Input sanitization');
drop table slides

- New attacks and countermeasures:
  - SQL injection
  - Background on web architectures