Malware

CS 161: Computer Security

Guest Lecturer: Paul Pearce

January 31, 2014

Slide credits: Vern Paxson

Announcements

- David is back on Monday
- HW0 due 11:59pm tonight
  - **Pick up account forms today if you haven't!**
- C review session, Saturday 2-4pm, 306 Soda
- Ava's discussion section Tuesday 2-3pm is moving from 105 Latimer to 71 Evans

The Problem of Malware

- Malware = malicious code that runs on a victim's system
- How does it manage to run?
  - Buffer overflow in network-accessible vulnerable service
  - Vulnerable client (e.g. browser) connects to remote system that sends over an attack (a *driveby*)
  - *Social engineering:* trick user into running/installing
  - "Autorun" functionality (esp. from plugging in USB device)
  - Slipped into a system component (at manufacture; compromise of software provider; substituted via MITM)
  - Attacker with local access downloads/runs it directly
    - Might include using a "local root" exploit for privileged access

Malware Driveby Example

- Visit http://facebook.com with your web browser
  - Facebook.com serves a malicious advertisement
  - Malicious advertisement exploits a bug in a browser plugin (Buffer overrun?)
    - (Which plugin? Probably Java. Seriously. Disable Java.)
  - Malicious advertisement injects code into your browser
  - Game Over
  - Actual real world example!
- Browser Driveby is just one example.
  - Another: malicious mp3's

What Can Malware Do?

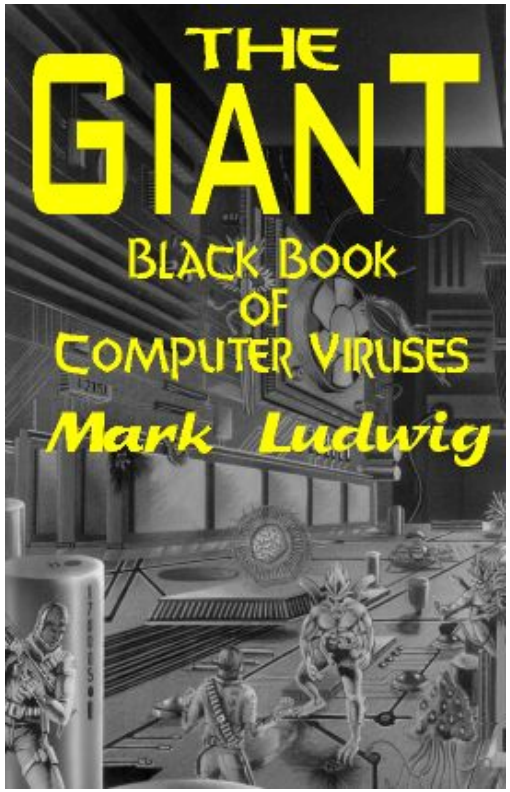
- Pretty much *anything*
  - Payload generally decoupled from how manages to run
  - Only subject to permissions under which it runs
- Examples:
  - Brag or exhort or extort (pop up a message/display)
  - Trash files (just to be nasty)
  - Damage hardware (!)
  - Launch external activity (for $?) (spam, *click fraud*, DoS)
  - Scan files, steal information (*exfiltrate*)
  - Keylogging; screen / audio / camera capture
  - Encrypt files (*ransomware*)
  - Other examples?
- Possibly delayed until condition occurs
  - "time bomb" / "logic bomb"

Malware That Automatically Propagates

- Virus = code that propagates (**replicates**) across systems by arranging to have itself *eventually executed,* creating a new additional instance
  - Generally infects by altering stored code
- Worm = code that self-propagates/replicates across systems by arranging to have itself *immediately executed* (creating new addl. instance)
  - Generally infects by altering running code
  - No user intervention required
  - See supplemental slides for lots of worm examples
- Line between these isn't always so crisp; plus some malware incorporates both styles

The Problem of Viruses

- Opportunistic = code will eventually execute
  - Generally due to user action
    - Running an app, booting their system, opening an attachment
- Separate notions: how it *propagates* vs. what else it does when executed (*payload*)
- General infection strategy: find some code lying around, alter it to include the virus



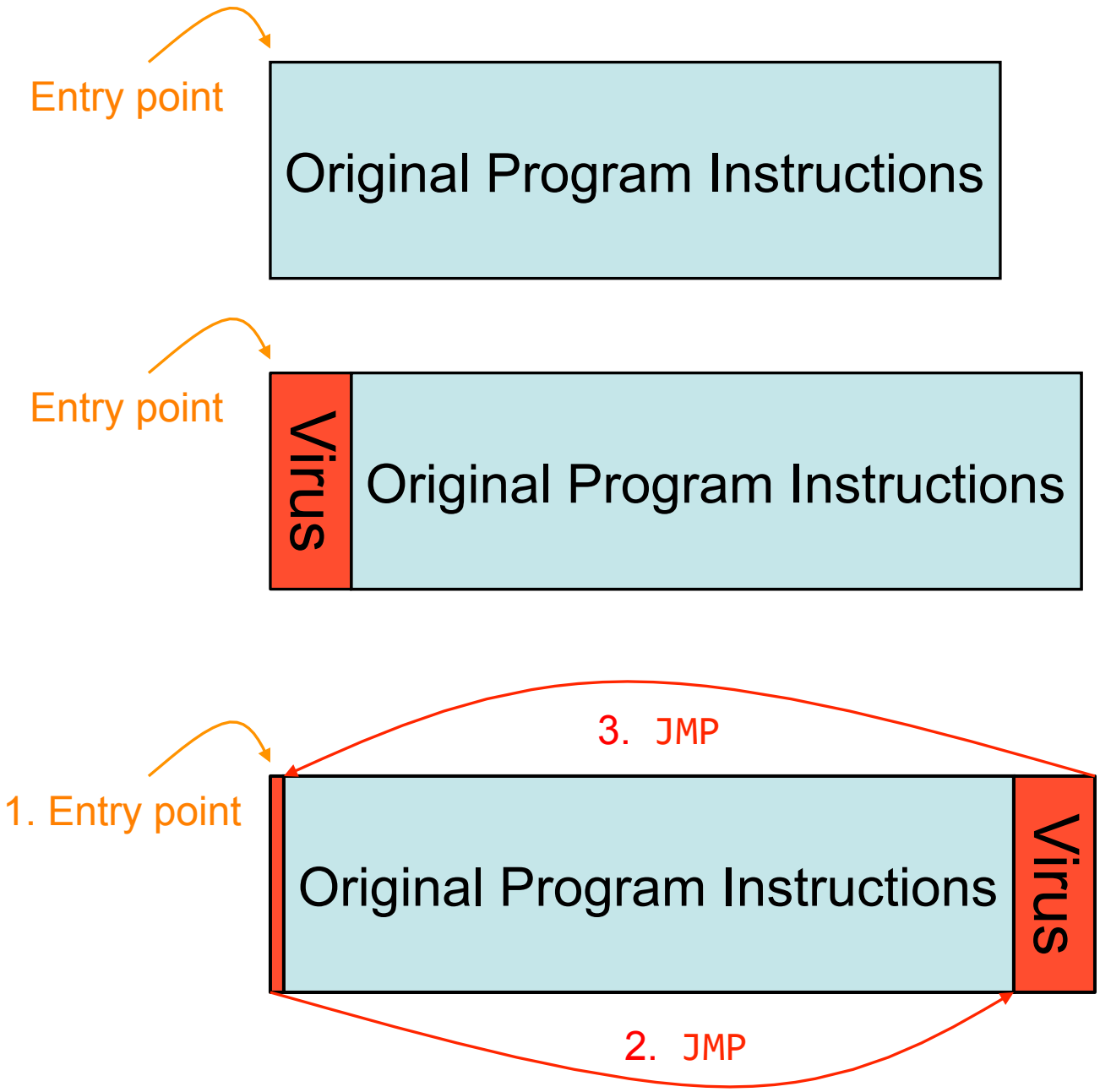
- Have been around for decades …
  - … resulting **arms race** has heavily influenced evolution of modern malware

Propagation

- When virus runs, it looks for an opportunity to infect additional systems
- One approach: look for USB-attached thumb drive, alter any executables it holds to include the virus
  - Strategy: when drive later attached to another system & altered executable runs, it locates and infects executables on new system's hard drive

***autorun** is handy here!*

- Or: when user sends email w/ attachment, virus alters attachment to add a copy of itself
  - Works for attachment types that include programmability
  - E.g., Word documents (macros), PDFs (Javascript)
  - Virus can also send out such email proactively, using user's address book + enticing subject ("I Love You")



Original program instructions can be:

- Application the user runs
- Run-time library / routines resident in memory
- Disk blocks used to boot OS
- Autorun file on USB device
- …

Other variants are possible; whatever manages to get the virus code executed

Detecting Viruses

- Signature-based detection
  - Look for bytes corresponding to injected virus code
  - High utility due to replicating nature
    - If you capture a virus V on one system, by its nature the virus will be trying to infect *many other systems*
    - Can protect those other systems by installing recognizer for V
- Drove development of multi-billion $$ AV industry (AV = "antivirus")
  - So many endemic viruses that detecting well-known ones becomes a "*checklist item*" for security audits
- Using signature-based detection also has de facto utility for (glib) marketing
  - Companies compete on number of signatures …
    - … rather than their quality (harder for customer to assess)

VirusTotal is a free service that **analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware.

SHA256: 71d1723d1269abef2b78d6c46390452058c047bc44949bad8f493446f947c8bc

File name: qvodsetupls27.exe

Detection ratio: 41 / 46

Analysis date: 2013-04-11 11:56:27 UTC (3 days, 10 hours ago)

More details

Analysis | Additional information | Comments | Votes

Antivirus	Result	Update
Agnitum	Trojan.DR.Agent!AmUdZaEHJGw	20130410
AhnLab-V3	Dropper/Win32.Agent	20130410
AntiVir	DR/MicroJoiner.Gen	20130411
Antiy-AVL	-	20130411
Avast	Win32:Microjoin-CD [Trj]	20130411
AVG	Dropper.Tiny.I	20130411
BitDefender	Trojan.Crypt.CG	20130411

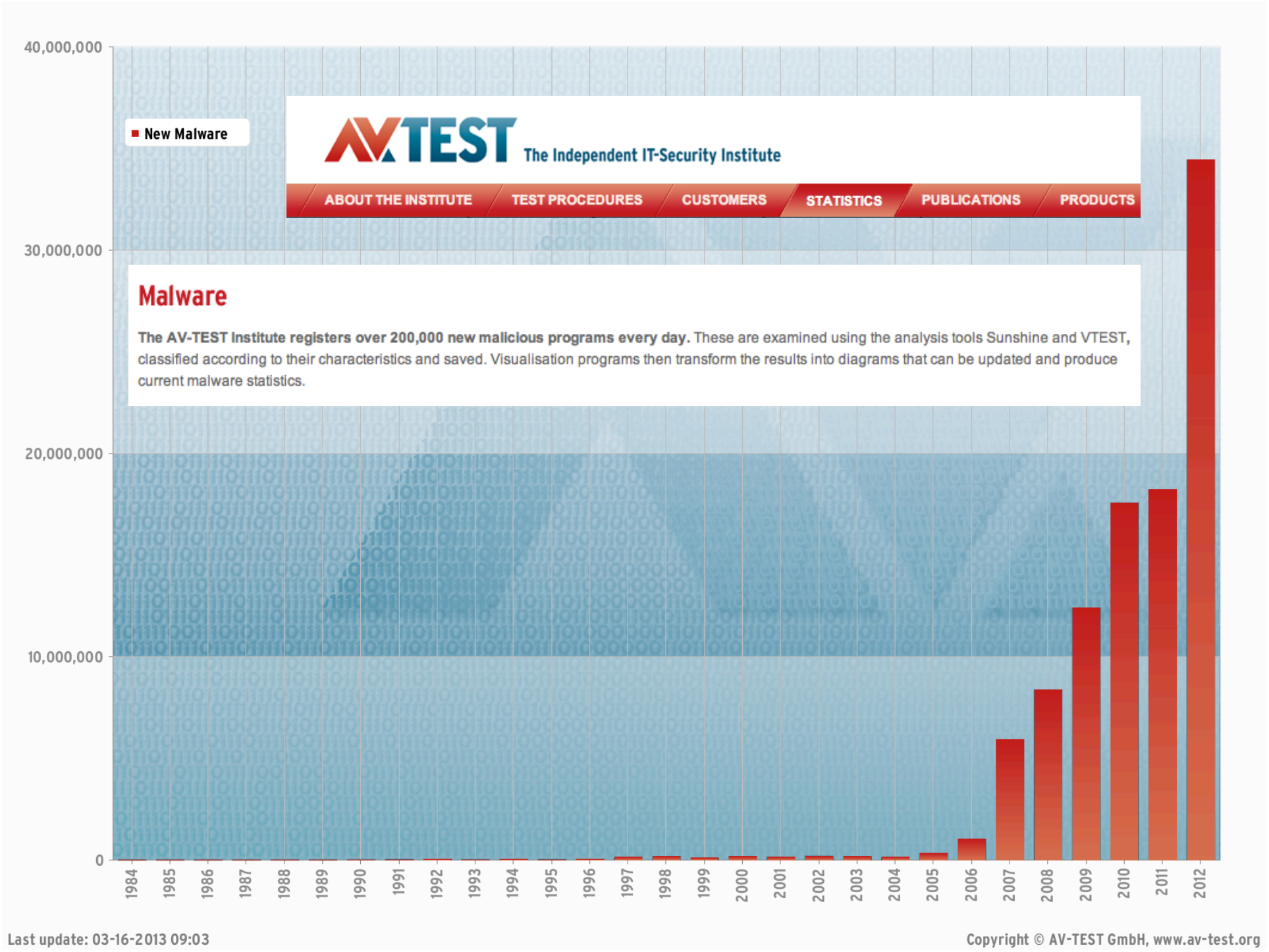
Virus Writer / AV *Arms Race*

- If you are a virus writer and your beautiful new creations don't get very far because each time you write one, the AV companies quickly push out a signature for it ….
  - …. *What are you going to do?*
- Need to keep changing your viruses …
  - … or at least changing their appearance!
- How can you mechanize the creation of new instances of your viruses …
  - … so that whenever your virus propagates, what it injects as a copy of itself looks different? **Repacking**
- See bonus slides for discussion of poly and metamorphic viruses

How Much Malware Is Out There?

- **Repacking** can lead to miscounting a single virus outbreak as instead reflecting 1,000s of *seemingly different* viruses

- Thus take care in interpreting vendor statistics on malcode varieties
  - (Also note: public perception that many varieties exist is *in the vendors' own interest*)



Last update: 03-16-2013 09:03

Infection Cleanup

- Once malware detected on a system, how do we get rid of it?
- May require restoring/repairing many files
  - This is part of what AV companies sell: per-specimen disinfection procedures
- What about if malware executed with adminstrator privileges?

  "*nuke the entire site from orbit. It's the only way to be sure*"

  - ALIENS

  - i.e., rebuild system from **original media + data backups**
- Malware may include a **rootkit**: *kernel patches* to hide its presence (its existence on disk, processes)

Botnets

- Collection of compromised machines (bots) under (unified) control of an attacker (botmaster)
- Method of compromise decoupled from method of control
  - Launch a worm / virus / drive-by infection / project 1 / etc.
- Upon infection, new bot "*phones home*" to rendezvous w/ botnet *command-and-control* (**C&C**)
- Lots of ways to architect C&C:
  - Star topology; hierarchical; peer-to-peer
  - Encrypted/stealthy communication
- Botmaster uses C&C to push out commands and updates

Example of C&C Messages

1. Activation (report from bot to botmaster)
2. Email address harvests
3. Spamming instructions
4. Delivery reports
5. Denial-Of-Service instructions
6. Sniffed passwords report

From the "Storm" botnet circa 2008

Fighting Bots / Botnets

- How can we defend against bots / botnets?

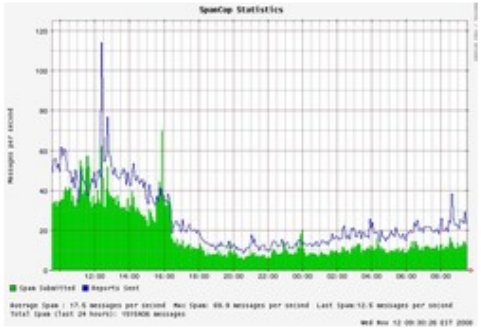
- Defense #1: prevent the initial bot infection
  - Equivalent to preventing malware infections in general …. HARD
- Defense #2: Take down the C&C master server
  - Find its IP address, get associated ISP to pull plug

Security Fix

Brian Krebs on Computer Security

Spam Volumes Drop by Two-Thirds After Firm Goes Offline

The volume of junk e-mail sent worldwide plummeted on Tuesday after a Web hosting firm identified by the computer security community as a major host of organizations engaged in spam activity was taken offline. (**Note**: A link to the full story on McColo's demise is available here.)



Experts say the precipitous drop-off in spam comes from Internet providers unplugging **McColo Corp.**, a hosting provider in Northern California that was the home base for machines responsible for coordinating the sending of roughly 75 percent of all spam each day.

In an alert sent out Wednesday morning, e-mail security firm **IronPort** said:

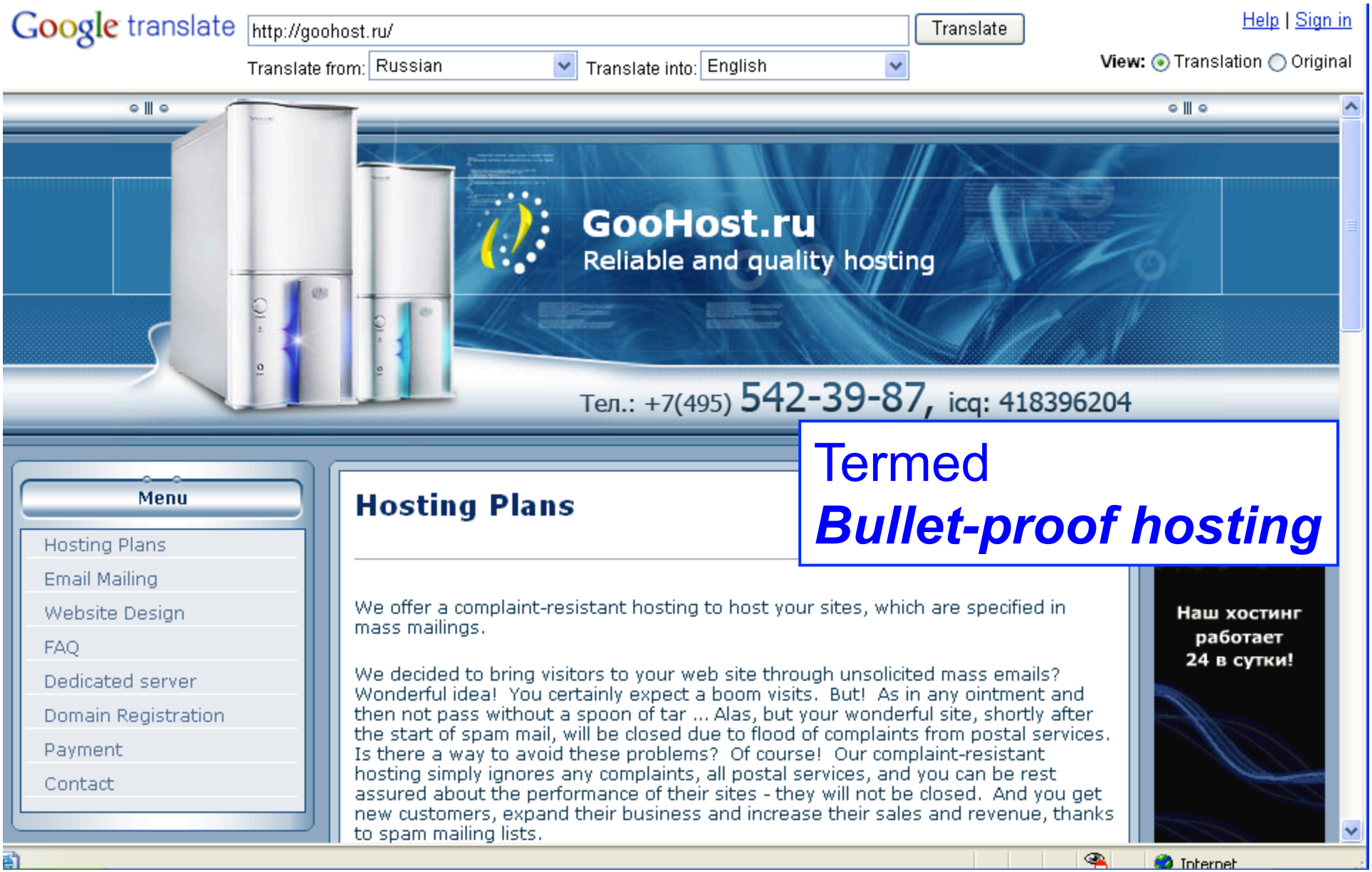
> In the afternoon of Tuesday 11/11, IronPort saw a drop of almost 2/3 of overall spam volume, correlating with a drop in IronPort's SenderBase queries. While we investigated what we thought might be a technical problem, a major spam network, McColo Corp., was shutdown, as reported by The Washington Post on Tuesday evening.

Spamcop.net's graphic shows a similar decline, from about 40 spam e-

Fighting Bots / Botnets

- How can we defend against bots / botnets?

- Defense #1: prevent the initial bot infection
  - Equivalent to preventing malware infections in general …. HARD
- Defense #2: Take down the C&C master server
  - Find its IP address, get associated ISP to pull plug
- Botmaster countermeasures?
  - Counter #1: keep moving around the master server
    - Bots resolve a domain name to find it (e.g. `c-and-c.evil.com`)
    - Rapidly alter address associated w/ name ("fast flux")
  - Counter #2: buy off the ISP …


Google translate
http://goohost.ru/
Translate
Translate from: Russian
Translate into: English
Help | Sign in
View: Translation Original
GooHost.ru
Reliable and quality hosting
Тел.: +7(495) 542-39-87, icq: 418396204
Termed
Bullet-proof hosting
Menu
Hosting Plans
Email Mailing
Website Design
FAQ
Dedicated server
Domain Registration
Payment
Contact
Hosting Plans
We offer a complaint-resistant hosting to host your sites, which are specified in mass mailings.
We decided to bring visitors to your web site through unsolicited mass emails? Wonderful idea! You certainly expect a boom visits. But! As in any ointment and then not pass without a spoon of tar ... Alas, but your wonderful site, shortly after the start of spam mail, will be closed due to flood of complaints from postal services. Is there a way to avoid these problems? Of course! Our complaint-resistant hosting simply ignores any complaints, all postal services, and you can be rest assured about the performance of their sites - they will not be closed. And you get new customers, expand their business and increase their sales and revenue, thanks to spam mailing lists.
Наш хостинг работает 24 в сутки!
Internet

Obuzoustoychivy hosting is more expensive than usual, but you will have the full guarantee that your site no one ever closes, it will always be available to your customers!

MINI PLAN	
Volume disc	400 MB
Domains	1
Traffic *	Unlimited
FTP-access	there is
MySQL database	there is
Control panel	there is
COST	4 000 rub. / 1 month.

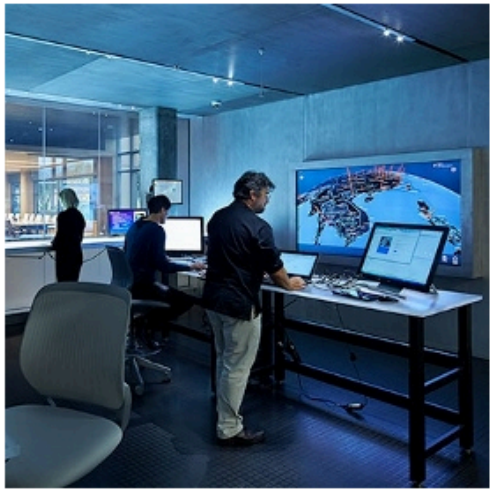
STARTER PLAN	
Volume disc	500 mb
Domains	3
Traffic *	Unlimited
FTP-access	there is
MySQL database	there is
Control panel	there is
COST	5 000 rub. / 1 month.
BUSINESS PLAN	
Volume disc	1000 mb
Domains	7
Traffic *	Unlimited
FTP-access	there is
MySQL database	there is
Control panel	there is
COST	7 000 rub. / 1 month.
PREMIUM PLAN	

Fighting Bots / Botnets, con't

- Defense #3: Legal action
  - Use law enforcement to seize the domain names and IP addresses used for C&C
  - This is what's currently often used, often to good effect …

Microsoft, Feds Disrupt ZeroAccess Botnet

By Chloe Albanesius December 6, 2013 11:55am EST 6 Comments



Microsoft today announced that it has "successfully disrupted" the ZeroAccess botnet, which has infected nearly 2 million computers all over the world, and cost online advertisers more than $2.7 million each month.

Redmond worked in conjunction with Europol's European Cybercrime Centre (EC3), the FBI, and tech firms like A10 Networks to take action against ZeroAccess, also known as Sirefef.

Microsoft also filed suit in Texas district court that seeks a preliminary injunction directing U.S. Internet Service Providers and other entities in control of the Internet domains and IP Addresses to disable access to the botnet and preserve any content and material associated with it to help with Microsoft's case.

Microsoft noted that the sophisticated nature of ZeroAccess means that it has not been fully eliminated, but "we do expect this legal and technical action will significantly disrupt the botnet's operation by disrupting the cybercriminals' business model and forcing them to rebuild their criminal infrastructure, as well as preventing victims' computers from committing the fraudulent schemes," Richard Domingues Boscovich, assistant general counsel with Microsoft's Digital Crimes Unit, said in a statement.

Fighting Bots / Botnets, con't

- Defense #3: Legal action
  - Use law enforcement to seize the domain name and IP addresses used for C&C
  - Botmaster counter-measure?
  - Each day (say), bots generate large list of possible domain names using a Domain Generation Algorithm
    - Large = 50K, in some cases
  - Bots then try a random subset looking for a C&C server
    - Server **cryptographically signs** its replies, so bot can't be duped
    - Attacker just needs to hang on to a small portion of names to retain control over botnet
- This is becoming state-of-the-art …
- Counter-counter measure?
  - Behavioral signature: look for hosts that make a lot of failed DNS lookups (research)

Addressing The Botnet Problem

- What are our prospects for securing the Internet from the threat of botnets? What angles can we pursue?
- Angle #1: detection/cleanup
  - Detecting infection of individual bots hard as it's the *defend-against-general-malware* problem
  - Detecting bot doing C&C likely a losing battle as attackers improve their sneakiness & crypto
  - Cleanup today lacks oomph:
    - **Who's responsible?** … and do they **care**? (*externalities*)
    - Landscape could greatly change with different model of liability
- Angle #2: go after the C&C systems / botmasters
  - Difficult due to ease of Internet anonymity & complexities of international law
    - But: a number of recent successes in this regard
    - Including some via peer pressure rather than law enforcement (McColo)

Addressing The Problem, con't

- Angle #3: prevention
  - Bots require installing new executables or modifying existing ones
  - Perhaps via infection …
    - … or perhaps just via user being fooled / imprudent
- In general, preventing malware infection is *hard.* Really hard
- What if we were able to provably secure 99% of all desktops!
  - (Good luck with that)
  - Is this good enough? Are we now safe?
  - No!
  - This is an asymmetric problem
    - Defenders must defend everything
    - Attackers need only a handful of targets

Addressing The Problem, con't

- Better models?
- We could lock down systems so OS prohibits user from changing configuration
  - Sacrifices flexibility
  - How does this work for home users?
  - => Mobile (Android/iOS). Did this solve the problem?
- Or: structure OS/browser using Privilege Separation
  - Does this solve the problem?
  - Depends on how granular the privileges are … and how secure the privileged components are

Summary

- Malware = malicious code that runs on a victim's system
  - Infection can occur in a variety of ways
- Some malware propagates automatically
  - Viruses
  - Worms
- Botnet = set of compromised machines
  - Botnets are a modern, persistent, and very real threat
  - Extremely hard problem

Closing Thought…

- As long as criminals can continue to monetize malware, the malware threat is likely to remain
  - Stay tuned for upcoming Cybercrime and Underground Economy lectures for more

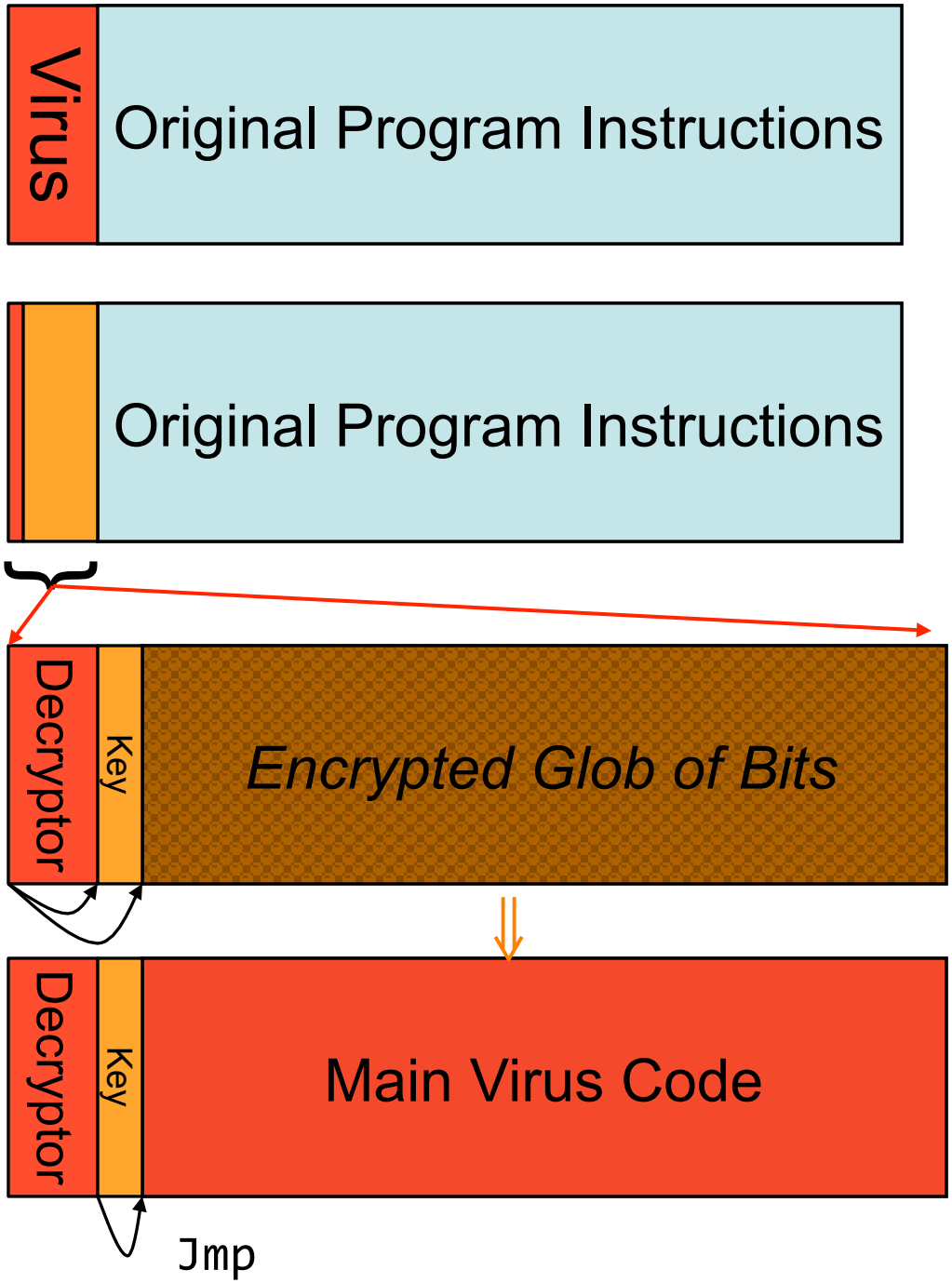
Questions?

Bonus Slides!

- You are not responsible for the content of these bonus slides

Polymorphic Code

- Later you will see technology for creating a representation of data apparently completely unrelated to the original: encryption!
- Idea: every time your virus propagates, it inserts a newly encrypted copy of itself
  - Clearly, encryption needs to vary
    - Either by using a different key each time
    - Or by including some random initial padding (like an IV)
  - Note: weak (but simple/fast) crypto algorithm works fine
    - No need for truly strong encryption, just obfuscation
- When injected code runs, it decrypts itself to obtain the original functionality



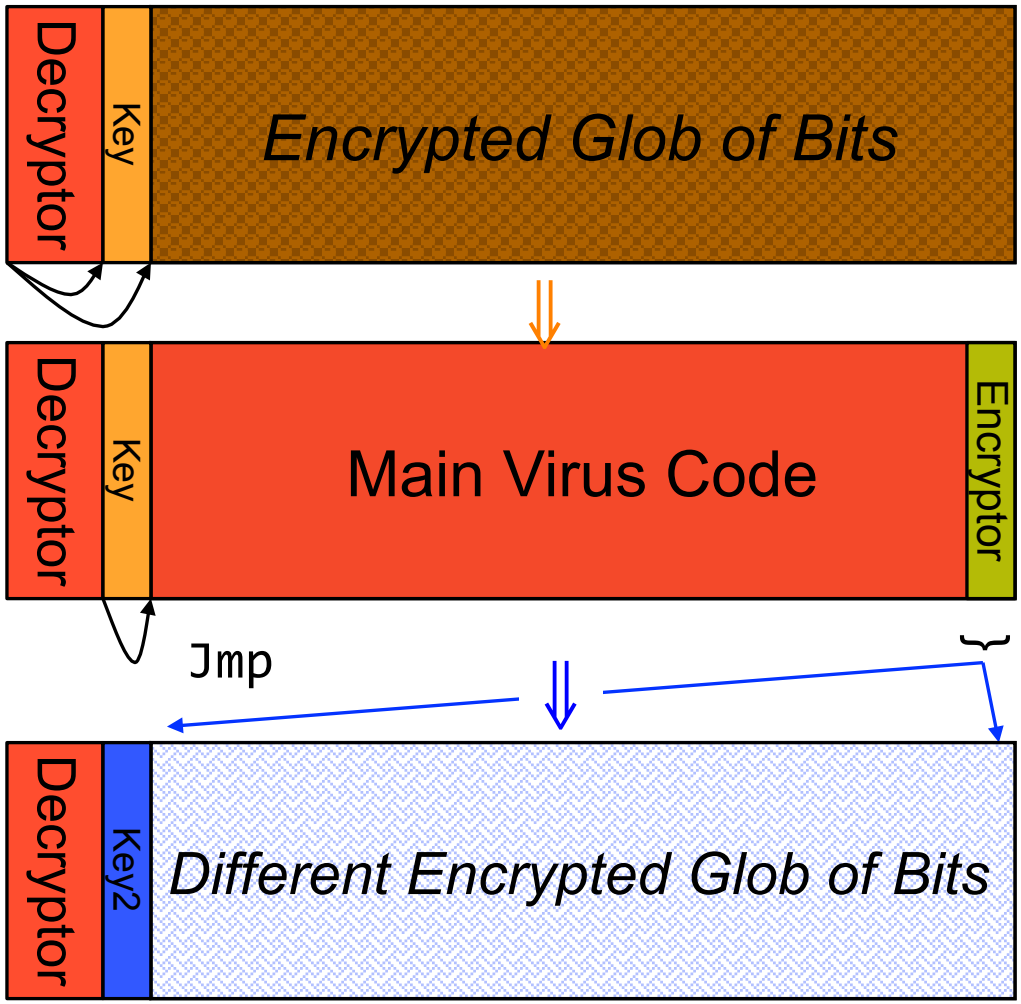
Instead of this …

Virus has *this* initial structure

When executed, decryptor applies key to decrypt the glob …

… and jumps to the decrypted code once stored in memory

Polymorphic Propagation



Once running, virus uses an *encryptor* with a new key to propagate

New virus instance bears little resemblance to original

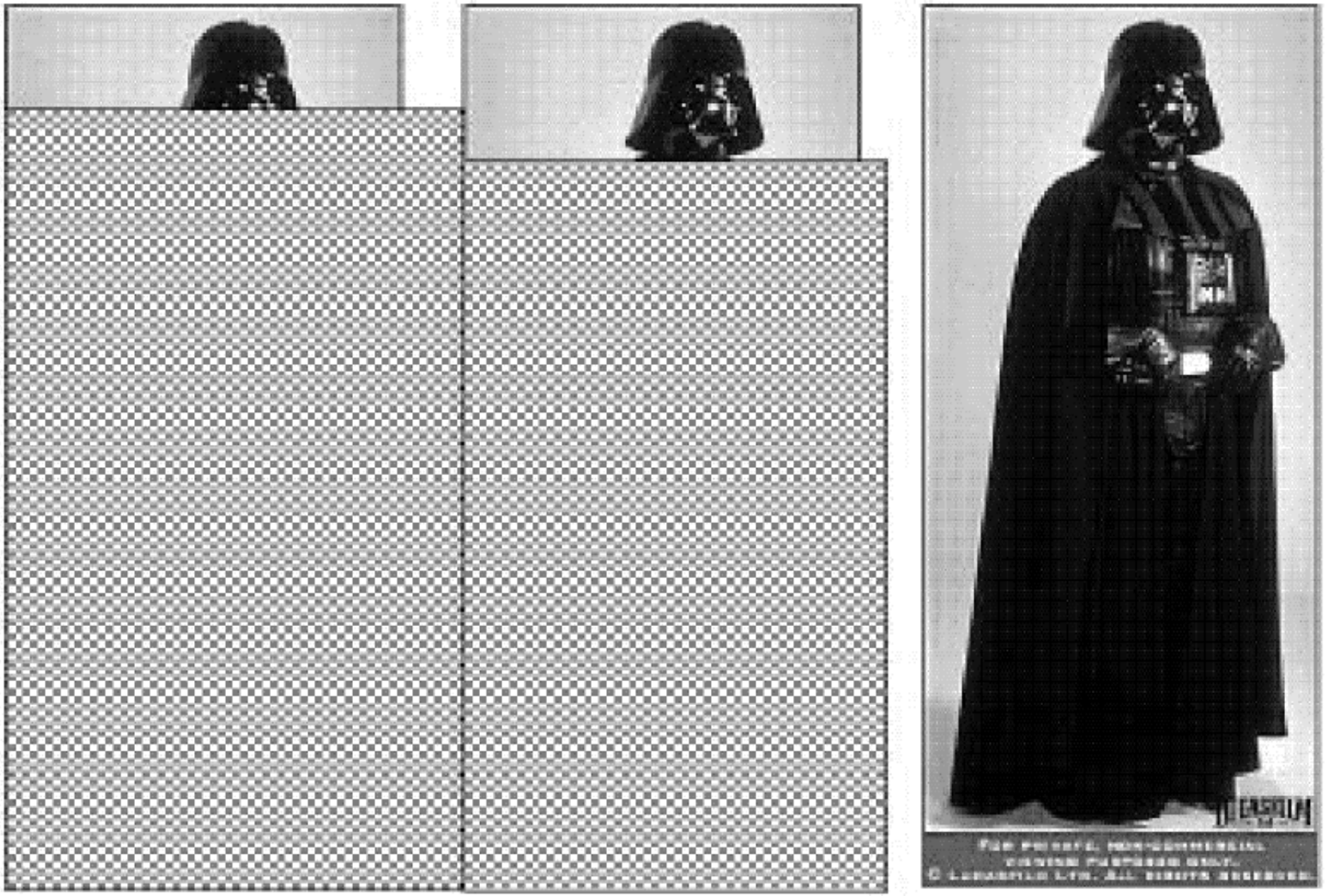
Arms Race: Polymorphic Code

- Given polymorphism, how might we then detect viruses?
- Idea #1: use narrow sig. that targets decryptor
  - Issues?
    - Less code to match against ⇒ more false positives
    - Virus writer spreads decryptor across existing code
- Idea #2: execute (or statically analyze) suspect code to see if it decrypts!
  - Issues?
    - Legitimate "*packers*" perform similar operations (decompression)
    - How long do you let the new code execute?
      - If decryptor only acts after lengthy legit execution, difficult to spot
- Virus-writer countermeasures?

Metamorphic Code

- Idea: every time the virus propagates, generate *semantically* different version of it!
  - Different semantics only at immediate level of execution; higher-level semantics remain same
- How could you do this?
- Include with the virus a code rewriter:
  - Inspects its own code, generates random variant, e.g.:
    - Renumber registers
    - Change order of conditional code
    - Reorder operations not dependent on one another
    - Replace one low-level algorithm with another
    - Remove some do-nothing padding and replace with different do-nothing padding ("chaff")
      - Can be very complex, legit code … if it's never called!

Polymorphic Code In Action



Hunting for Metamorphic, Szor & Ferrie, Symantec Corp., Virus Bulletin Conference, 2001

Metamorphic Code In Action

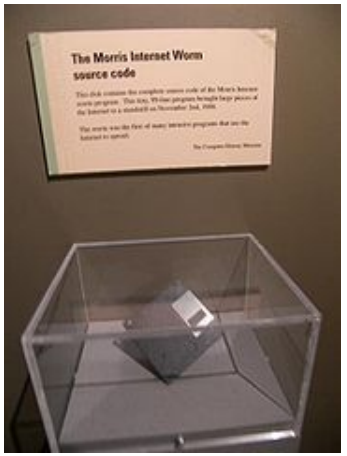
Hunting for Metamorphic, Szor & Ferrie, Symantec Corp., Virus Bulletin Conference, 2001

Detecting Metamorphic Viruses?

- Need to analyze execution behavior
  - Shift from syntax (*appearance* of instructions) to semantics (*effect* of instructions)
- Two stages: (1) AV company analyzes new virus to find behavioral signature; (2) AV software on end systems analyze suspect code to test for match to signature
- What countermeasures will the virus writer take?
  - Delay analysis by taking a long time to manifest behavior
    - Long time = await particular condition, or even simply clock time
  - Detect that execution occurs in an analyzed environment and if so behave differently
    - E.g., test whether running inside a debugger, or in a Virtual Machine
- Counter-countermeasure?
  - AV analysis looks for these tactics and skips over them
- Note: attacker has edge as *AV products supply an* ***oracle***

The Arrival of Internet Worms

- Worms date to Nov 2, 1988 - the *Morris Worm*
- ***Way*** ahead of its time
- Employed whole suite of tricks to infect systems …
  - *Multiple* buffer overflows
  - Guessable passwords
  - "Debug" configuration option that provided shell access
  - Common user accounts across multiple machines
- … and of tricks to find victims
  - Scan local subnet
  - Machines listed in system's network config
  - Look through user files for mention of remote hosts



Arrival of Internet Worms, con't

- Modern Era began Jul 13, 2001 with release of initial version of Code Red
- Exploited known buffer overflow in Microsoft IIS Web servers
  - *On by default* in many systems
  - Vulnerability & fix announced previous month
- Payload part 1: web site defacement
  - **`HELLO! Welcome to http://www.worm.com! Hacked By Chinese!`**
  - Only done if language setting = English

Code Red of Jul 13 2001, con't

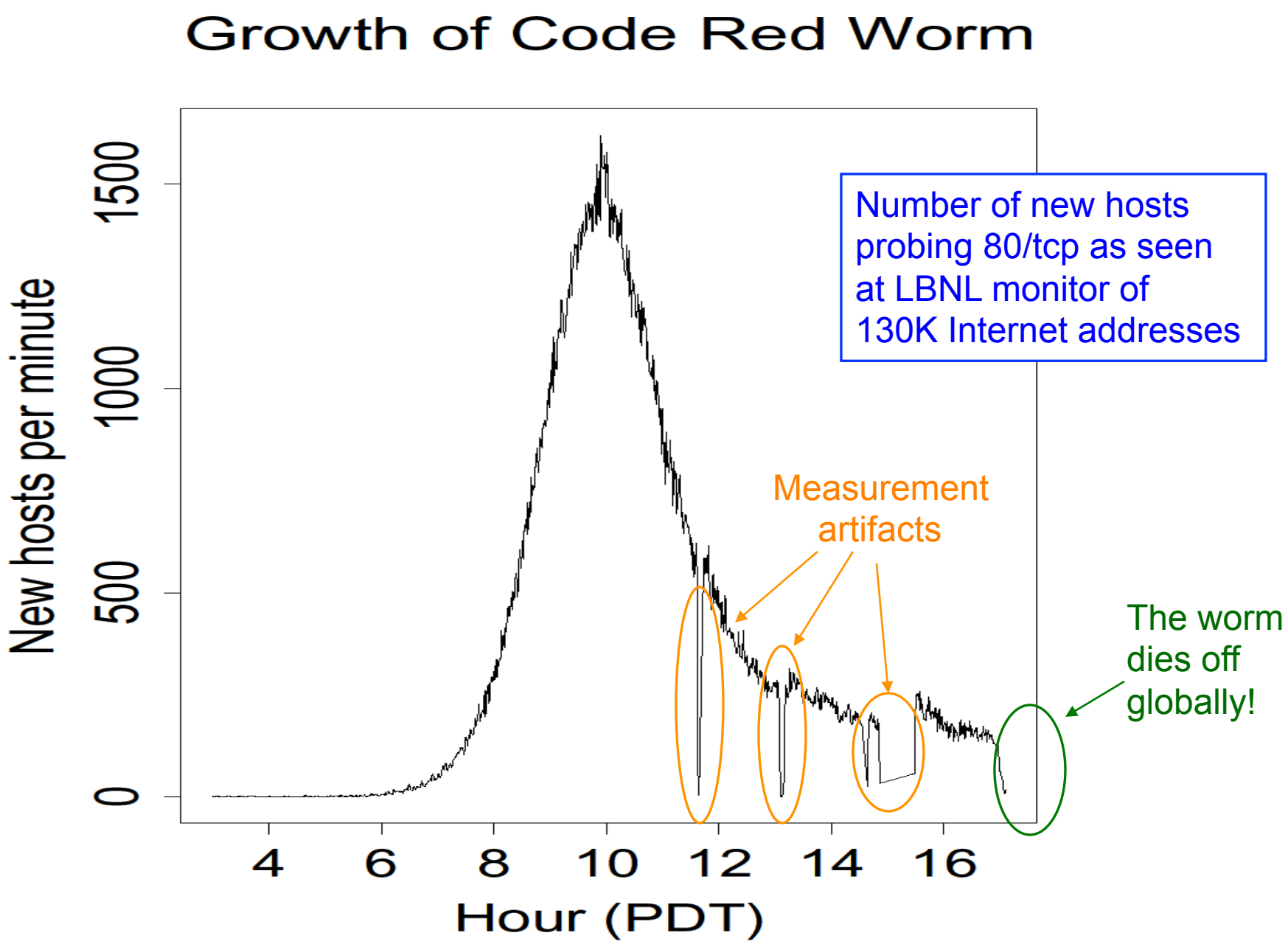
- Payload part 2: check day-of-the-month and …
  - … 1st through 20th of each month: spread
  - … 20th through end of each month: attack
    - Flooding attack against 198.137.240.91 …
    - … i.e., *www.whitehouse.gov*
- Spread: via *random scanning* of 32-bit IP address space
  - Generate pseudo-random 32-bit number; try connecting to it; if successful, try infecting it; repeat
  - Very common (but not fundamental) worm technique
- Each instance used same random number seed
  - How well does the worm spread?

Linear growth rate

Code Red, con't

- Revision released July 19, 2001.
- White House responds to threat of flooding attack by changing the address of *www.whitehouse.gov*
- Causes Code Red to die for date ≥ 20th of the month due to failure of TCP connection to establish.
  - Author didn't carefully test their code - buggy!

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


Growth of Code Red Worm
New hosts per minute
0
500
1000
1500
4
6
8
10
12
14
16
Hour (PDT)
Number of new hosts probing 80/tcp as seen at LBNL monitor of 130K Internet addresses
Measurement artifacts
The worm dies off globally!

Modeling Worm Spread

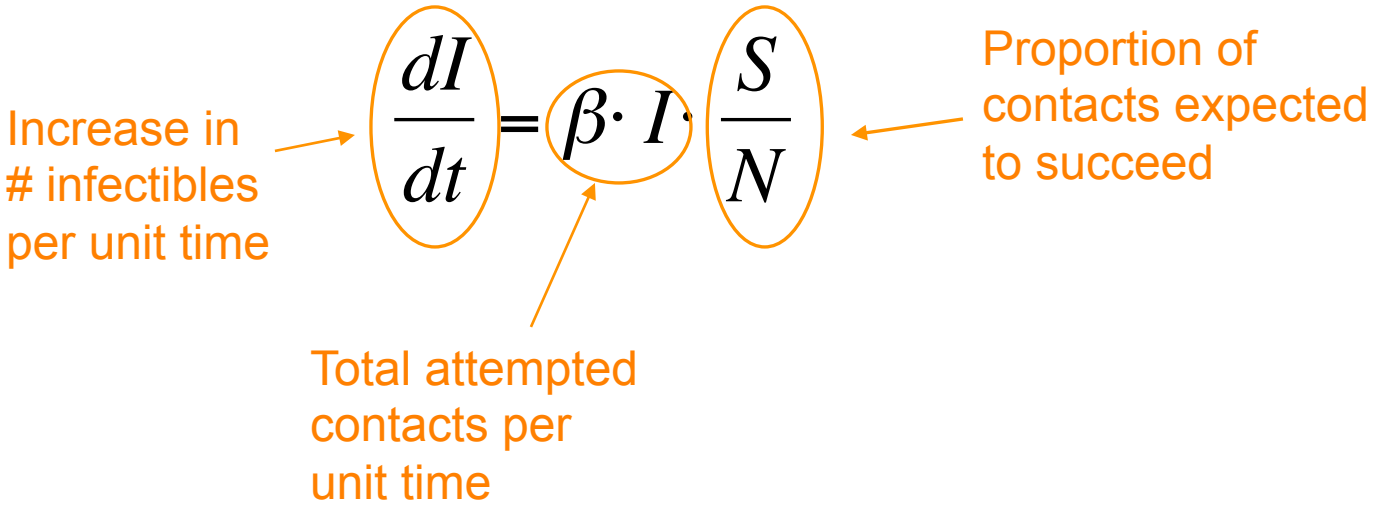
- Worm-spread often well described as *infectious epidemic*
  - Classic SI model: homogeneous random contacts
    - SI = Susceptible-Infectible
- Model parameters:
  - N: population size
  - S(t): susceptible hosts at time t.
  - I(t): infected hosts at time t.
  - β: *contact rate*
    - How many population members each infected host communicates with per unit time
    - E.g., if each infected host scans 10 Internet addresses per unit time, and 2% of Internet addresses run a vulnerable server $\Rightarrow \beta = 0.2$

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

- Normalized versions reflecting relative proportion of infected/susceptible hosts
  - $s(t) = S(t)/N \quad i(t) = I(t)/N \quad s(t) + i(t) = 1$

Computing How An Epidemic Progresses

- In continuous time:

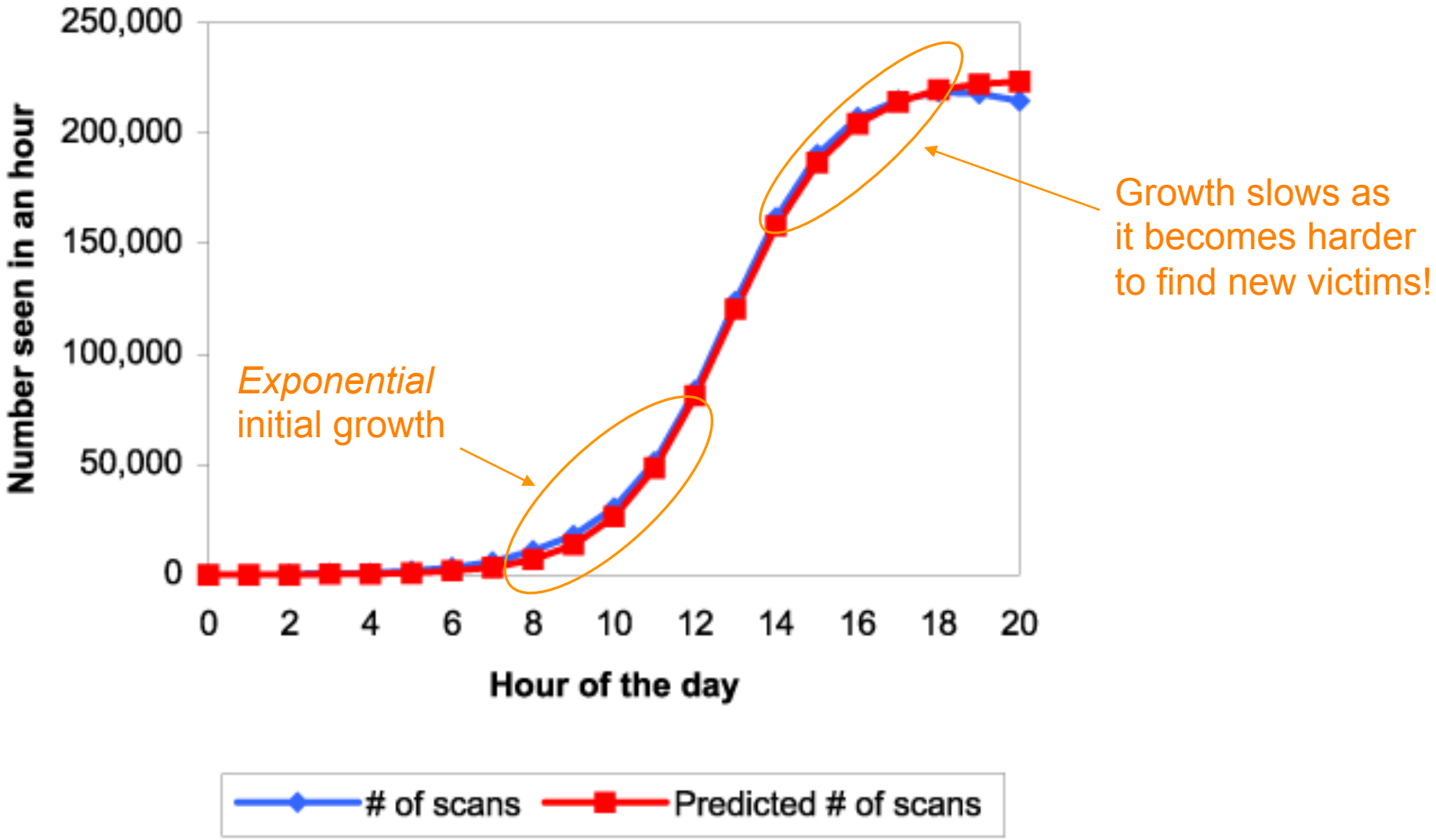


- Rewriting by using i(t) = I(t)/N, S = N - I:

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

Fraction infected grows as a *logistic*

Fitting the Model to Code Red



Spread of Code Red, con't

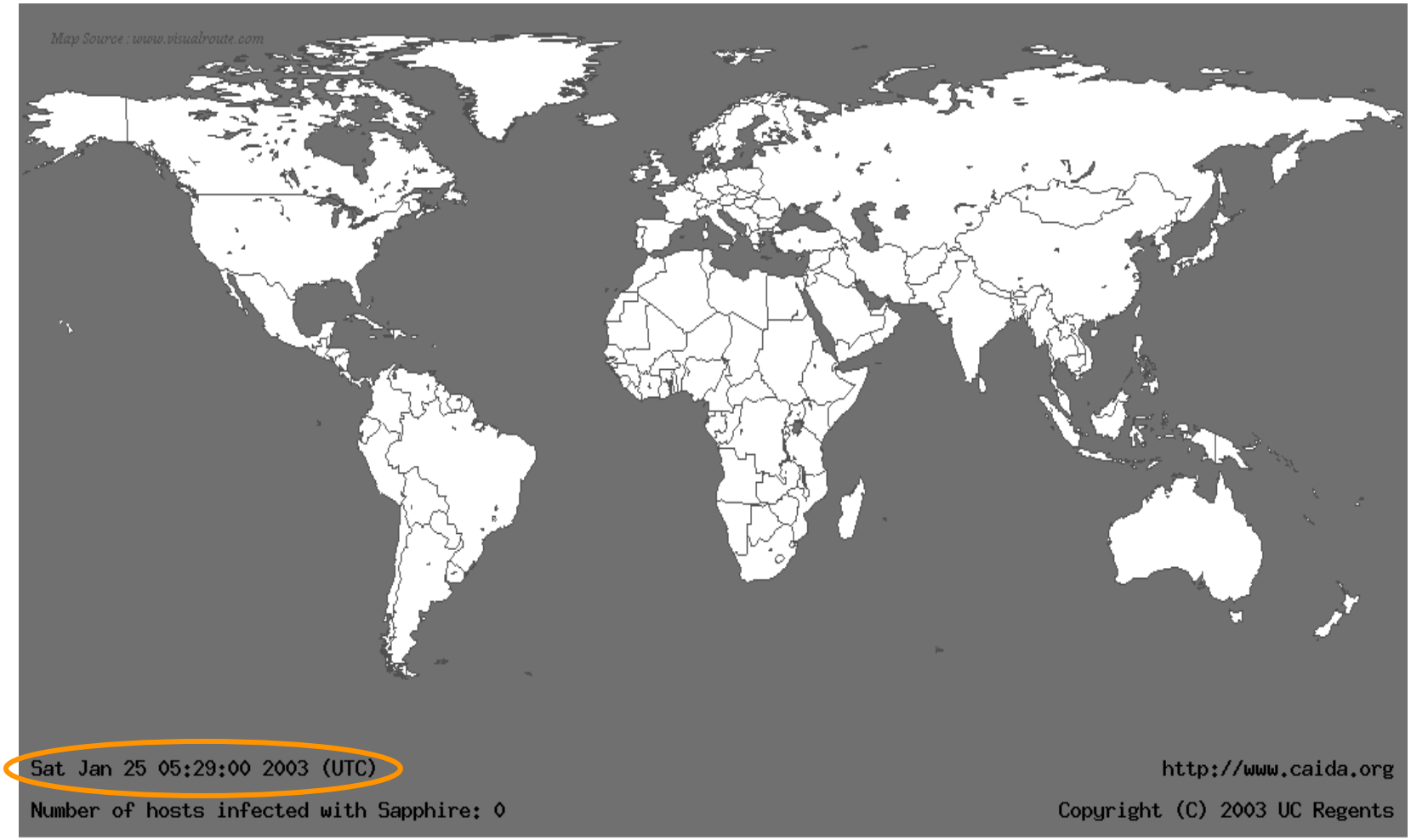
- Recall that # of new infections scales with contact rate β

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

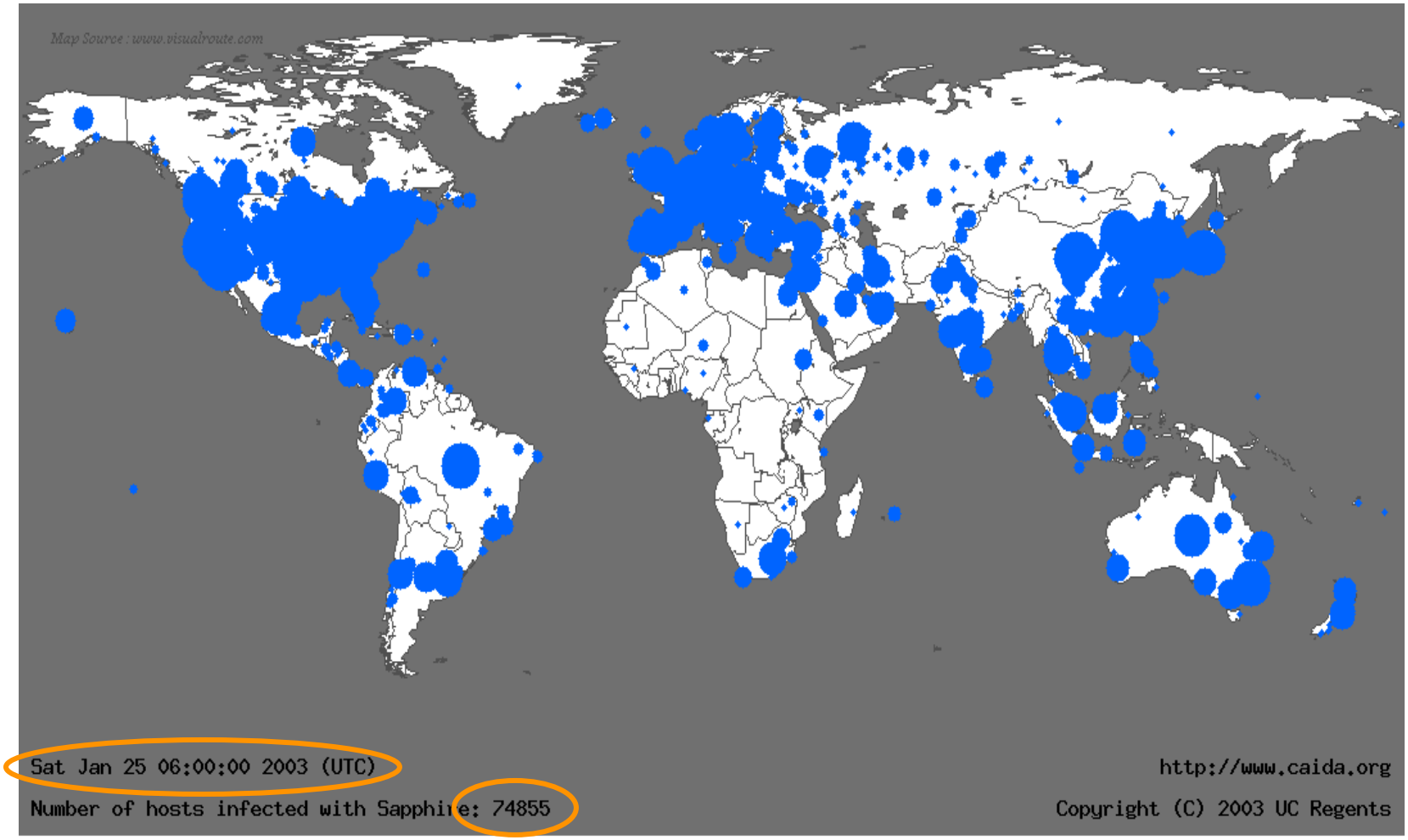
- For a scanning worm, β *increases* with N
  - Larger populations infected more quickly!
    - More likely that a given scan finds a population member
- Large-scale monitoring finds 360K systems infected with Code Red on July 19
  - Worm got them in 13 hours
- That night ($\Rightarrow 20^{th}$), worm dies due to DoS bug
- Worm actually managed to restart itself Aug. 1
  - … and each successive month for years to come!

Emergent behavior

Life Just Before Slammer



Life Just After Slammer



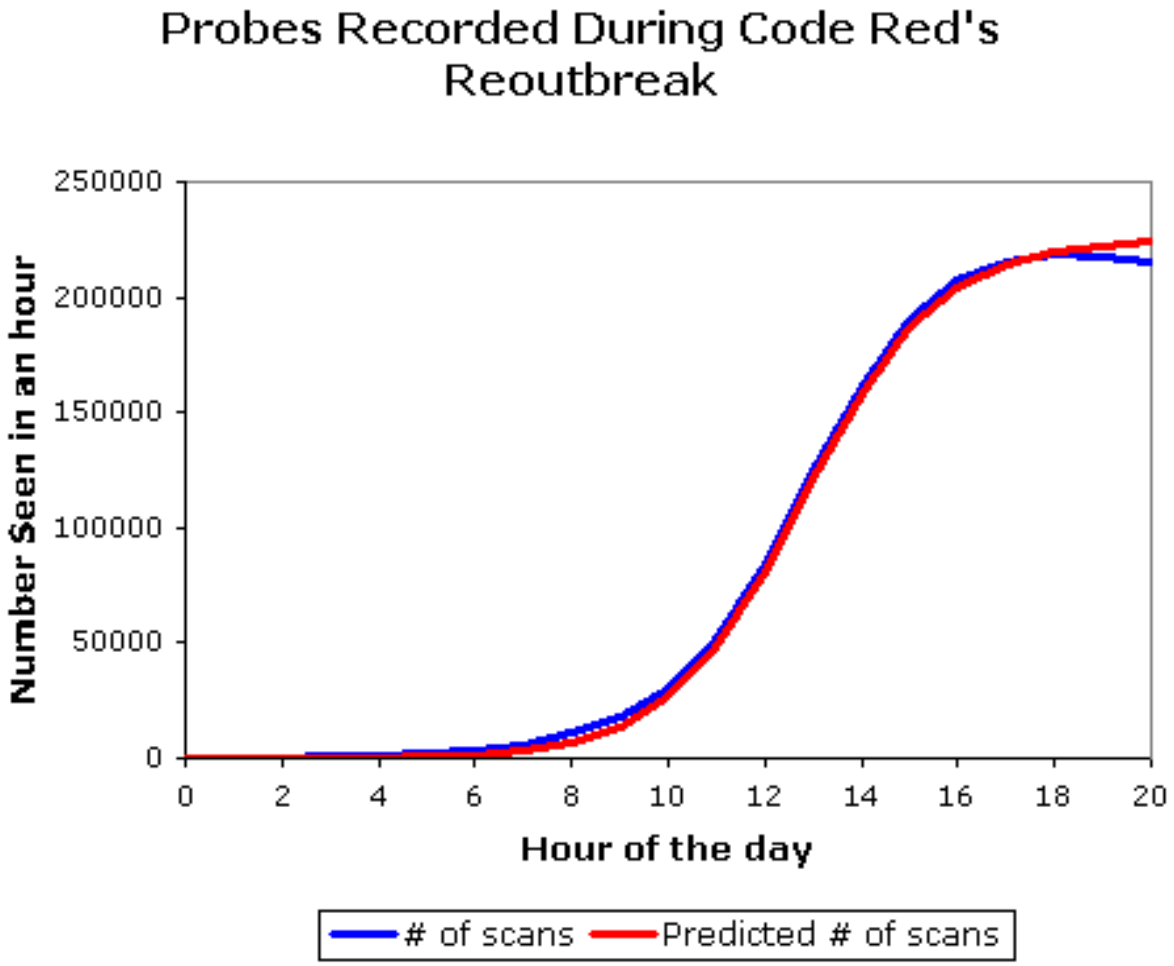
Going Fast: *Slammer*

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

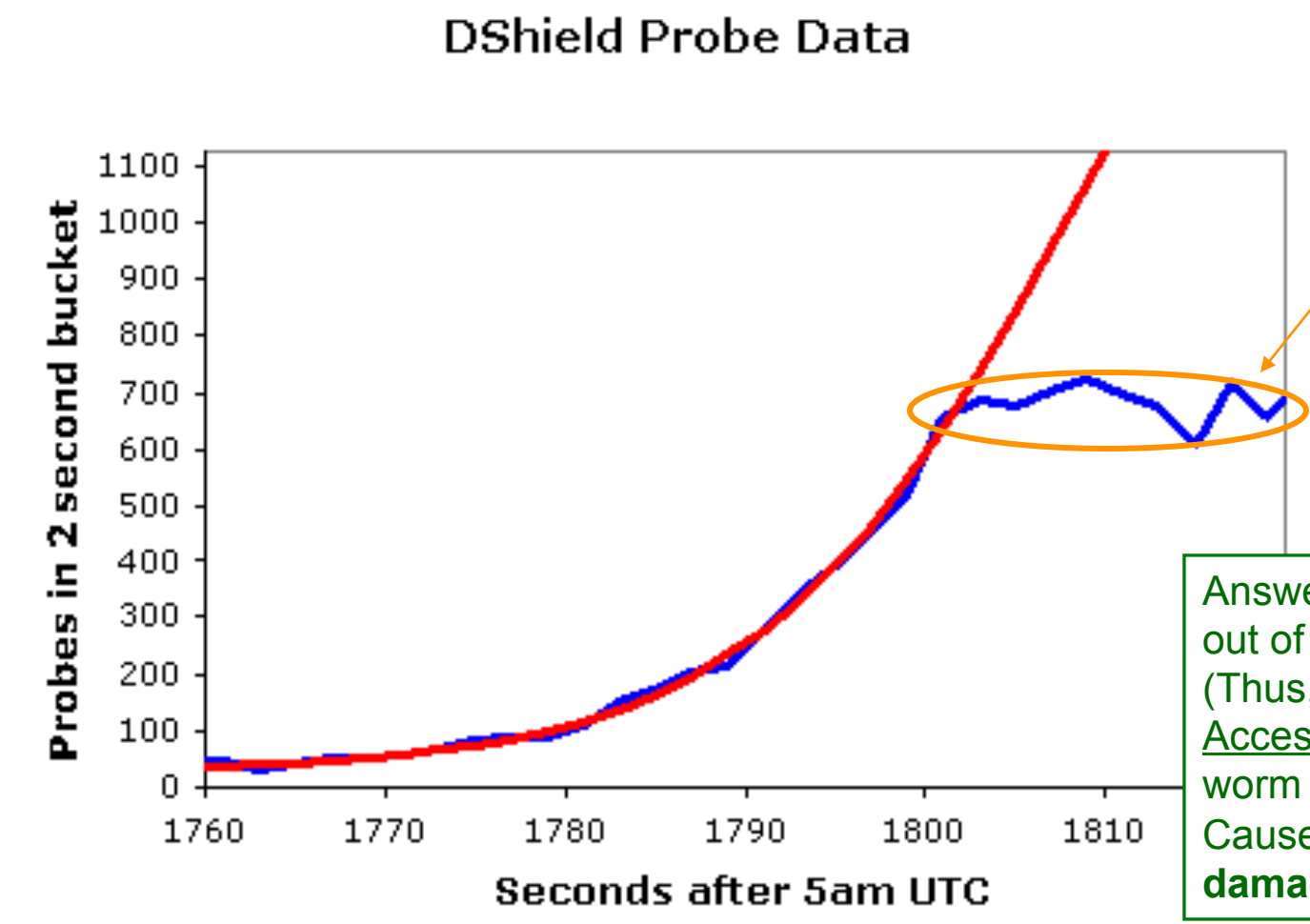
⇒ When scanning, worm could "fire and forget"
Stateless!

- Worm infected 75,000+ hosts in *<< 10 minutes*
- At its peak, **doubled every 8.5 seconds**

The Usual Logistic Growth



Slammer's Growth

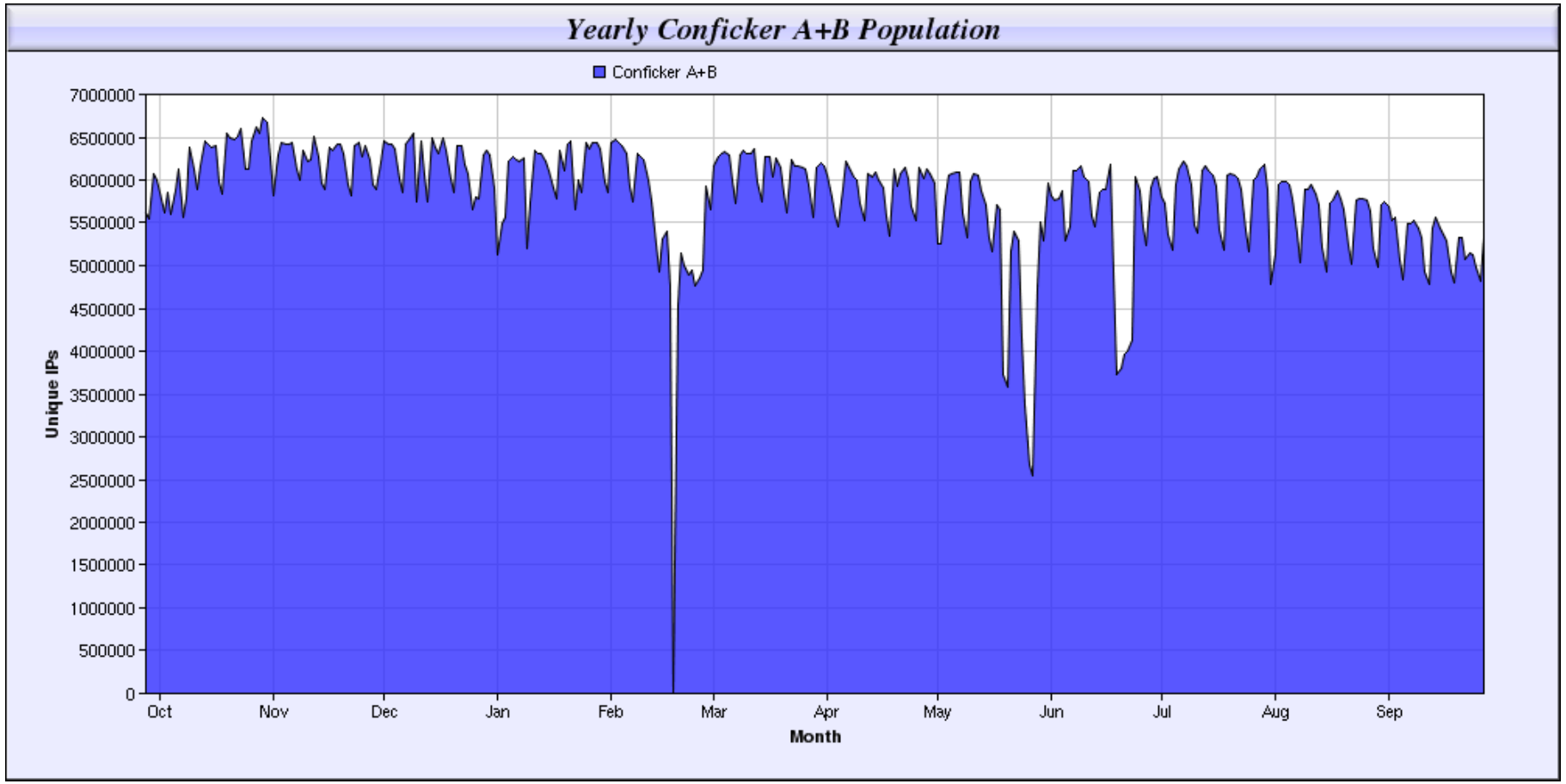


What could have caused growth to deviate from the model?

Hint: at this point the worm is generating *55,000,000 scans/sec*

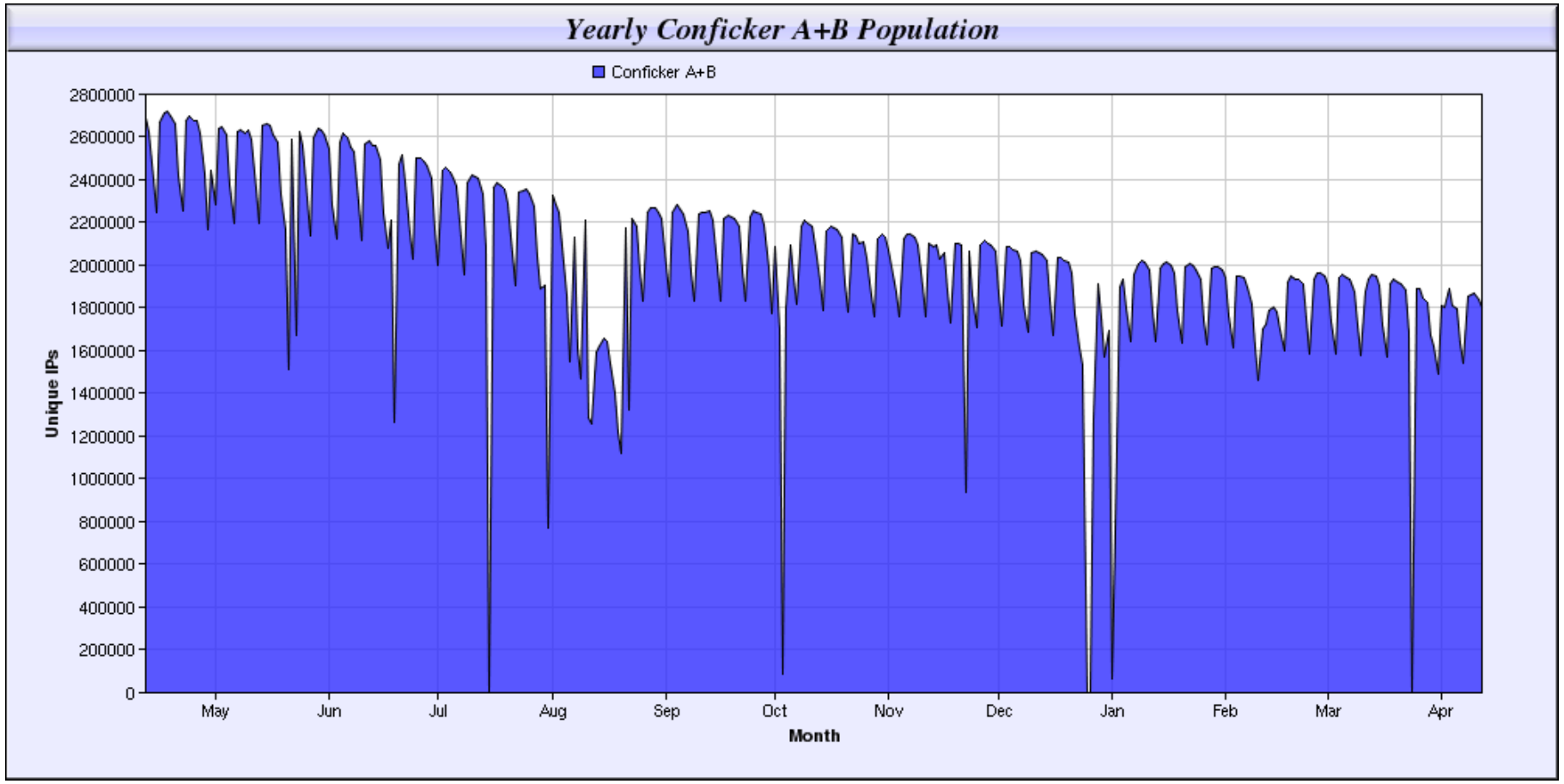
Answer: the Internet ran out of carrying capacity! (Thus, β decreased.) Access links used by worm completely clogged. Caused **major collateral damage**.

Big Worms: Conficker



2009 - 2010

Big Worms: Conficker



2012 - 2013

Stuxnet

- Discovered July 2010. (Released: Mar 2010?)
- Multi-mode spreading:
  - Initially spreads via USB (virus-like)
  - Once inside a network, quickly spreads internally using Windows RPC
- Kill switch: programmed to die June 24, 2012
- Targeted *SCADA* systems
  - Used for industrial control systems, like manufacturing, power plants
- Symantec: infections geographically clustered
  - Iran: 59%; Indonesia: 18%; India: 8%

Stuxnet, con't

- Used four *Zero Days*
  - Unprecedented expense on the part of the author
- "Rootkit" for hiding infection based on installing Windows drivers with valid digital signatures
  - Attacker stole private keys for certificates from two companies in Taiwan
- Payload: do nothing …
  - … unless attached to particular models of frequency converter drives operating at 807-1210Hz
  - … like those made in Iran (and Finland) …
  - … and used to operate centrifuges for producing *enriched uranium for nuclear weapons*

Stuxnet, con't

- Payload: do nothing …
  - … unless attached to particular models of frequency converter drives operating at 807-1210Hz
  - … like those made in Iran (and Finland) …
  - … and used to operate centrifuges for producing *enriched uranium for nuclear weapons*
- For these, worm would slowly increase drive frequency to 1410Hz …
  - … enough to cause centrifuge to **fly apart** …
  - … while sending out fake readings from control system indicating everything was okay …
- … and then drop it back to normal range

Israel Tests on Worm Called Crucial in Iran Nuclear Delay

By WILLIAM J. BROAD, JOHN MARKOFF and DAVID E. SANGER
Published: January 15, 2011

*This article is by **William J. Broad, John Markoff** and **David E. Sanger.***

Enlarge This Image

Nicholas Roberts for The New York Times

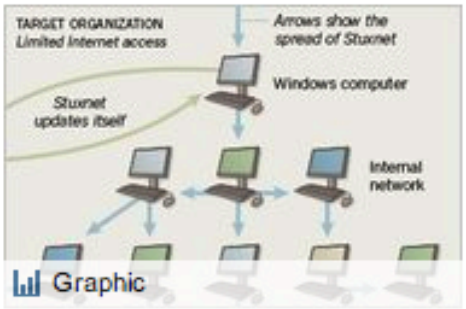
Ralph Langner, an independent computer security expert, solved Stuxnet.

The Dimona complex in the Negev desert is famous as the heavily guarded heart of Israel's never-acknowledged nuclear arms program, where neat rows of factories make atomic fuel for the arsenal.

Over the past two years, according to intelligence and military experts familiar with its operations, Dimona has taken on a new, equally secret role — as a critical testing ground in a joint American and Israeli effort to undermine Iran's efforts to make a bomb of its own.

Behind Dimona's barbed wire, the experts say, Israel has spun nuclear centrifuges virtually identical to Iran's at Natanz, where Iranian scientists are struggling to enrich uranium. They say Dimona tested the effectiveness of the Stuxnet computer worm, a destructive program that appears to have wiped out roughly a fifth of Iran's nuclear

Multimedia



How Stuxnet Spreads

Worm Take-Aways

- Potentially enormous reach/damage

  ⇒ *Weapon*

- Hard to get right
- Emergent behavior / surprising dynamics
- Remanence: worms stick around
  - E.g. Slammer still seen in 2013!

- *Propagation faster than human response*

Large-Scale Malware

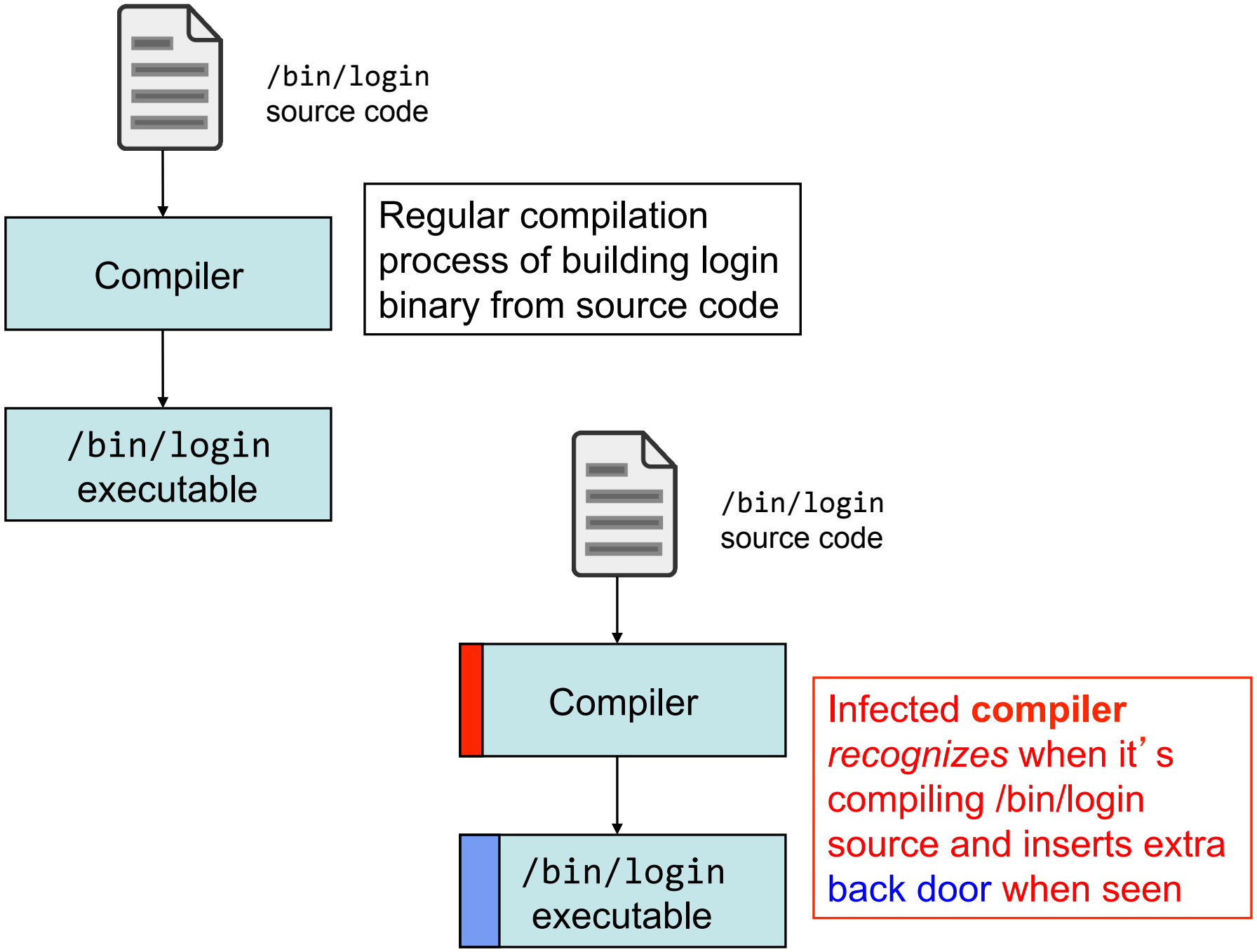
- Worm = code that self-propagates/replicates across systems by arranging to have itself immediately executed
  - Generally infects by altering running code
  - No user intervention required

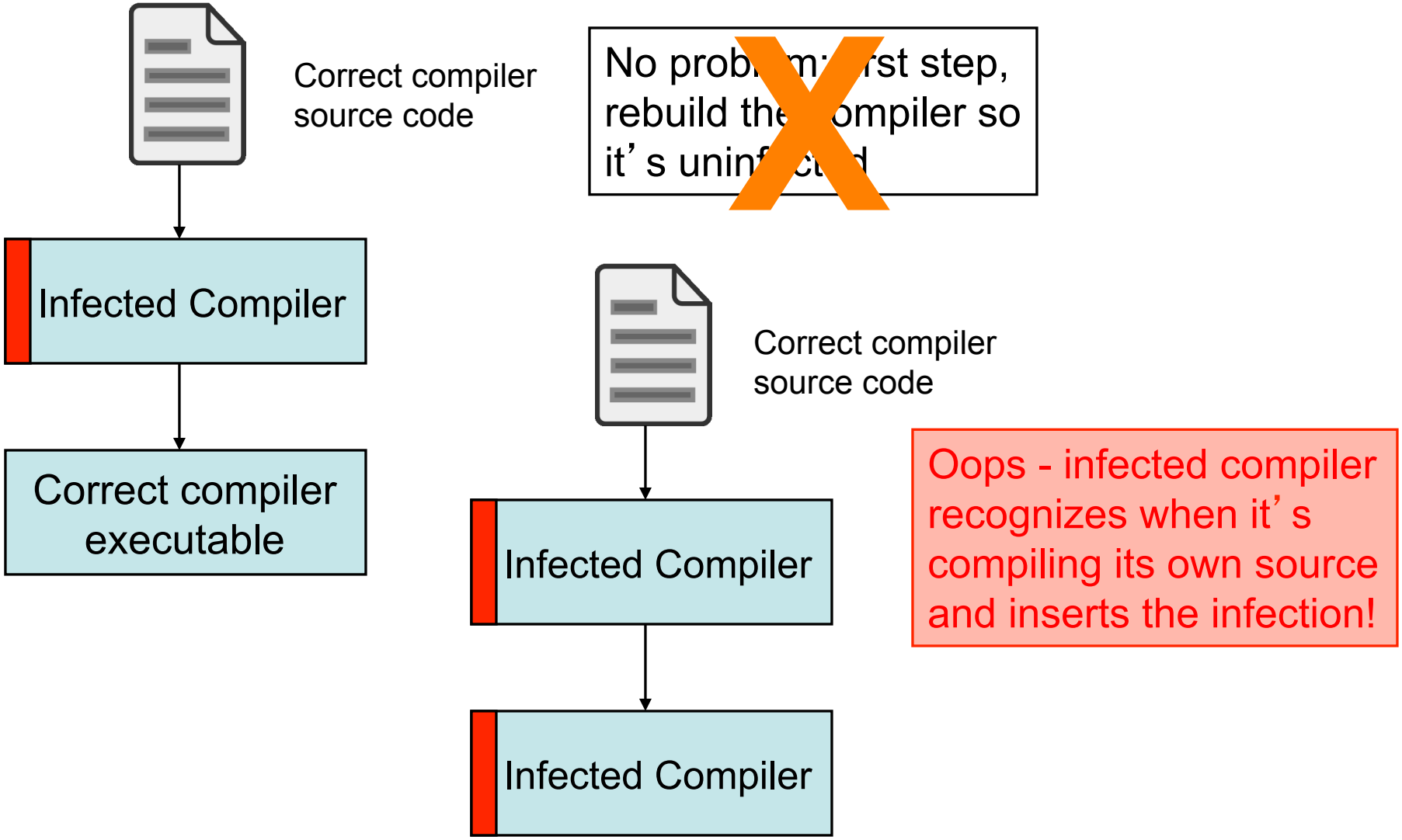
Infection Cleanup, con't

- If we have complete source code for system, we could rebuild from that instead, couldn't we?
- No!
- Suppose forensic analysis shows that virus introduced a backdoor in `/bin/login` executable
  - (Note: this threat isn't specific to viruses; applies to any malware)
- Cleanup procedure: rebuild `/bin/login` from source …
  - How's your complier doing…

Infection Cleanup, con't

- Cleanup procedure: rebuild `/bin/login` from source …


/bin/login
source code
Compiler
/bin/login
executable
Regular compilation process of building login binary from source code
/bin/login
source code
Compiler
/bin/login
executable
Infected **compiler** *recognizes* when it's compiling /bin/login source and inserts extra back door when seen



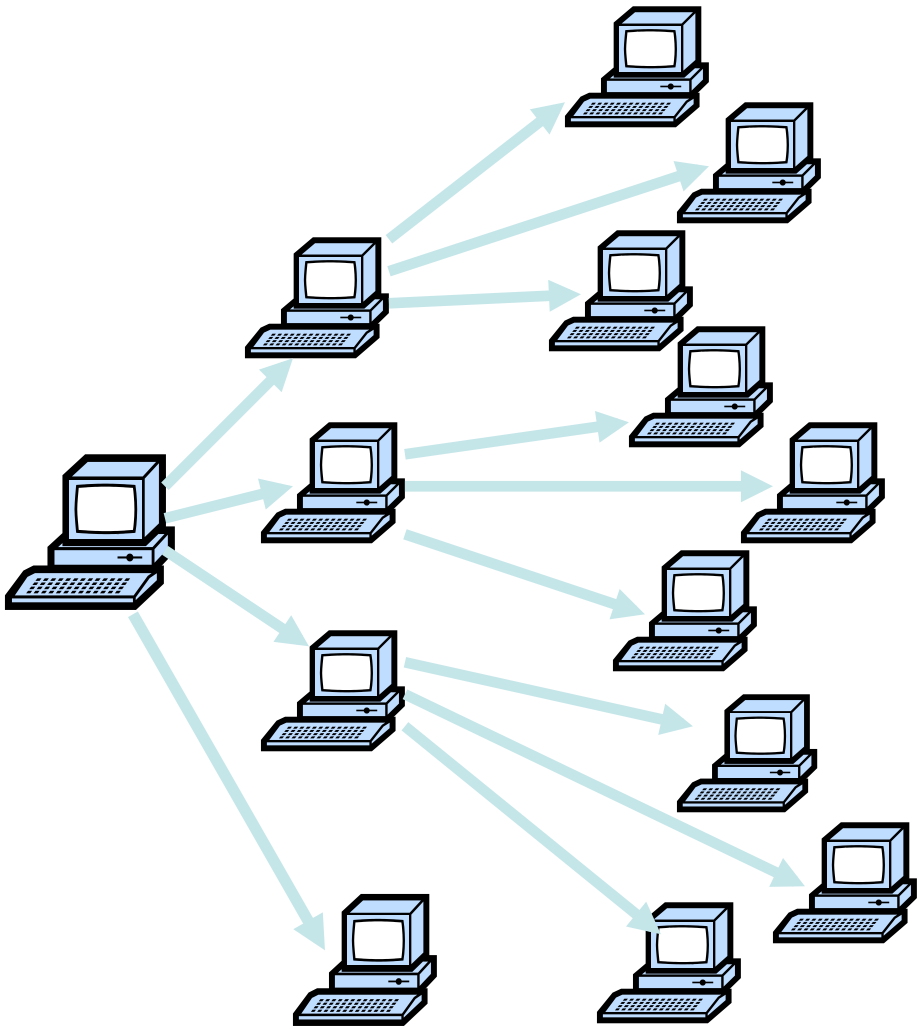
No amount of careful source-code scrutiny can prevent this problem.
And if the *hardware* has a back door …

Reflections on Trusting Trust
Turing-Award Lecture, Ken Thompson, 1983

Rapid Propagation

Worms can potentially spread quickly because they **parallelize** the process of propagating/ replicating.

Same holds for viruses, but they often spread more slowly since require some sort of user action to trigger each propagation.



Large-Scale Malware

- Worm = code that self-propagates/replicates across systems by arranging to have itself immediately executed
  - Generally infects by altering running code
  - No user intervention required
- Propagation includes notions of *targeting* & *exploit*
  - How does the worm **find** new prospective victims?
  - How does worm get code to **automatically run**?

- Botnet = set of compromised machines ("bots") under a common *command-and-control* (**C&C**)
  - Attacker might use a worm to get the bots, or other techniques; orthogonal to bot's use in botnet

Original URL: http://www.theregister.co.uk/2010/03/03/mariposa_botnet_bust_analysis/

How FBI, police busted massive botnet

12m zombie machines run by 3 admins

By **John Leyden**

Posted in Malware, 3rd March 2010 15:56 GMT

Analysis More details have emerged about a cybercrime investigation that led to the takedown of a botnet containing 12m zombie PCs and the arrest of three alleged kingpins who built and ran it.

As previously reported, the Mariposa botnet was principally geared towards stealing online login credentials for banks, email services and the like from compromised Windows PCs. The malware infected an estimated 12.7 million computers in more than 190 countries.

The Mariposa Working Group infiltrated the command-and-control structure of Mariposa to monitor the communication channels that relayed information from compromised systems back to the hackers who run the botnet. Analysis of the command system laid the groundwork for the December 2009 shutdown of the botnet, as well as shedding light on how the malware operated and provided a snapshot of the current state of the underground economy.

The botmasters made money by selling parts of the botnet to other cybercrooks,

Netkairo finally regained control of Mariposa and launched a denial of service attack against Defence Intelligence using all the bots in his control. This attack seriously impacted an ISP, leaving numerous clients without an Internet connection for several hours, including several Canadian universities and government institutions.

The botmasters made money by selling parts of the botnet to other cybercrooks,

laundering stolen bank login credentials and credit card details via an international network of money mules. Search engine manipulation and serving pop-up ads was also part of the illegal business model behind the botnet.

The criminal gang behind Mariposa called themselves the DDP (Días de Pesadilla or Nightmare Days) Team. They nearly always connected to the Mariposa controlled servers from anonymous VPN (Virtual Private Network) services, preventing investigators from identifying their real IP addresses.

However when the December shutdown operation happened, the gang's leader, alias Netkairo, panicked in his efforts to regain control of the botnet. Netkairo made the fatal error of connecting directly from his home computer instead of using the VPN, leaving a trail of digital fingerprints that led to a series of arrests two months later.

Netkairo finally regained control of Mariposa and launched a denial of service attack against Defence Intelligence using all the bots in his control. This attack seriously impacted an ISP, leaving numerous clients without an Internet connection for several hours, including several Canadian universities and government institutions.

Once again, the Mariposa Working Group managed to prevent the DDP Team from accessing Mariposa. We changed the DNS records, so the bots could not connect to the C&C servers and receive instructions, and at that moment we saw exactly how many bots were reporting. We were shocked to find that more than 12 million IP addresses were connecting and sending information to the C&C servers, making Mariposa one of the largest botnets in history.

alleged lieutenants “Ostiator” and “Johnyloleante” have been charged with cybercrime offences. More arrests are expected to follow.

Under Spanish law suspects are not named at this stage of proceedings. Pedro Bustamante, senior research advisor at Panda Security, said: “Our preliminary analysis indicates that the botmasters did not have advanced hacking skills.

"This is very alarming because it proves how sophisticated and effective malware distribution software has become, empowering relatively unskilled cyber criminals to inflict major damage and financial loss." ®