Malcode Continued

"I've been working day and night trying to secure the Internet of Things.

I finally made a breakthrough and it's called:

VLAN of Thing."

- Taylor Swift

Announcements...

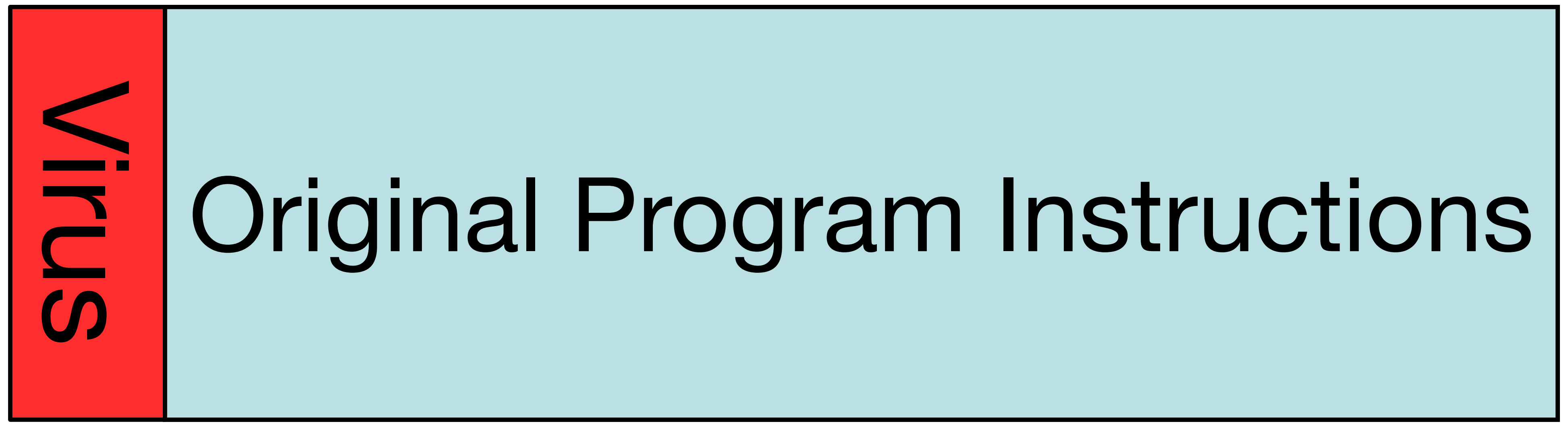
- Estimated grades should be out Real Soon Now
    - But remember, these are estimates....
    - And also remember, for most Grades Don't Matter (much)
- RRR week
    - Schedule of discussions will be up soon
    - No lecture
- Final will be very much like the midterm

Virus Writer / AV Arms Race

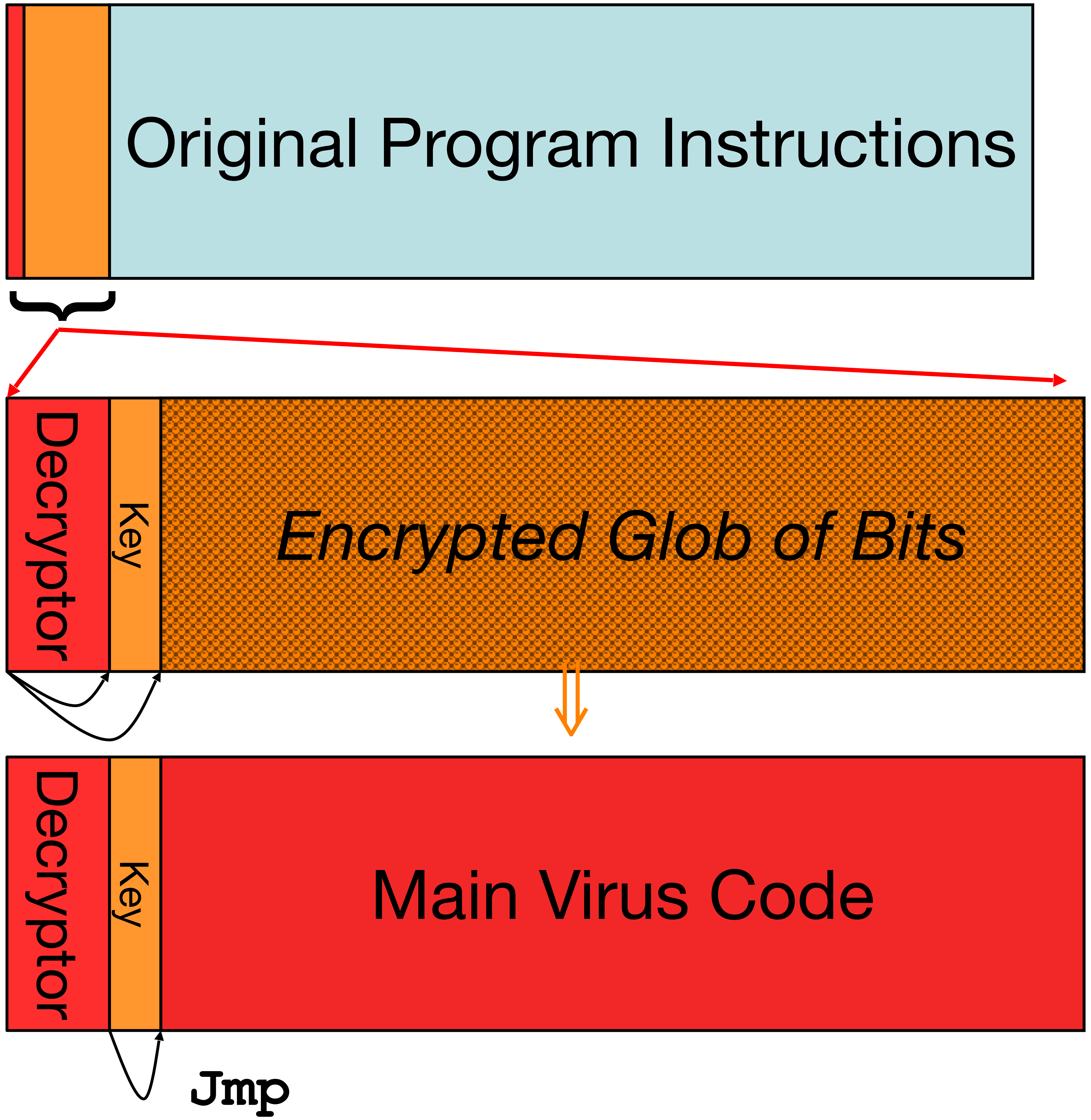
- If you are a virus writer and your beautiful new creations don’t get very far because each time you write one, the AV companies quickly push out a signature for it ….
  - …. What are you going to do?
- Need to keep changing your viruses …
  - … or at least changing their appearance!
- How can you mechanize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

Polymorphic Code

- We've already seen 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 …

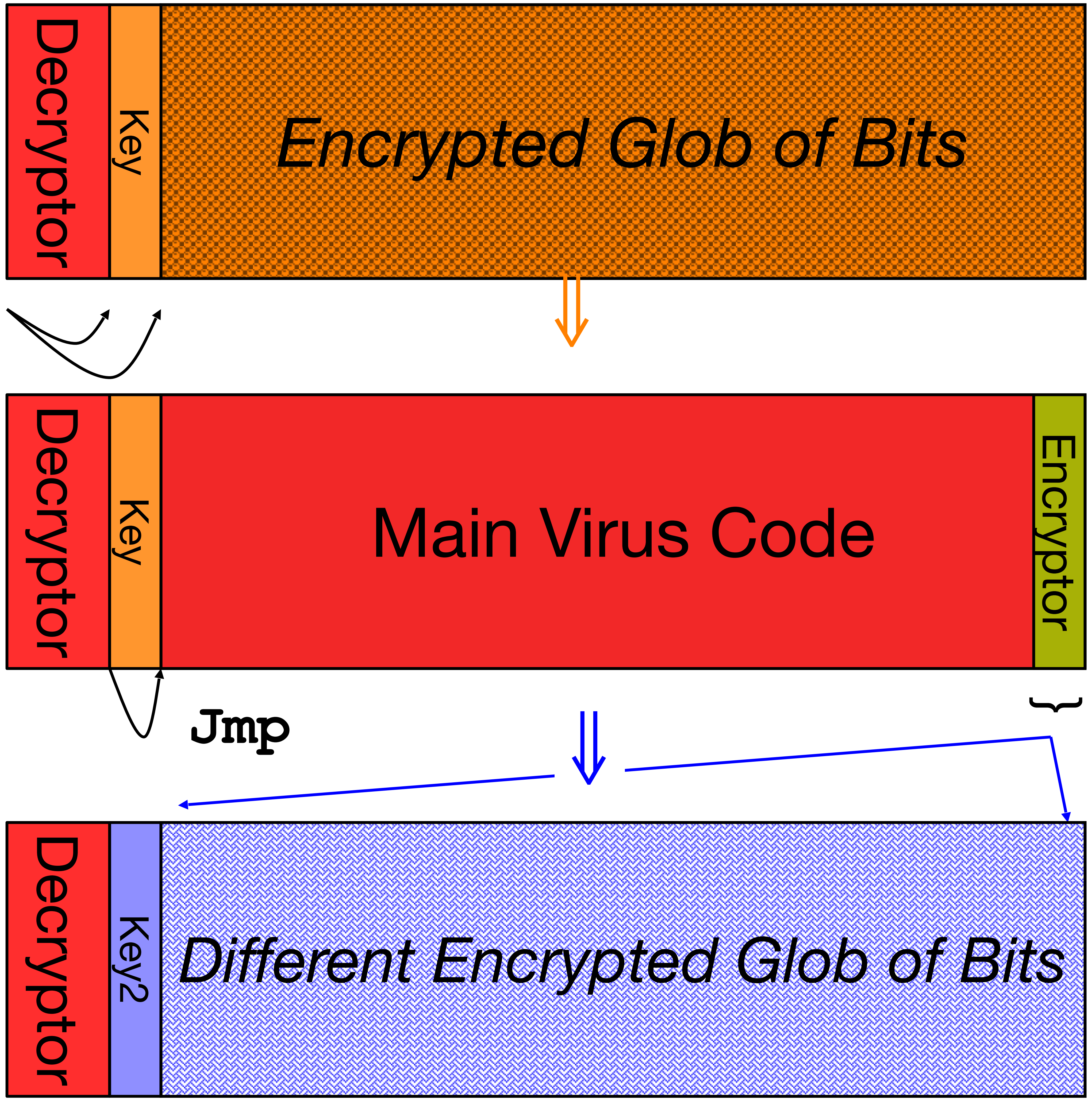


Virus has *this*
initial structure

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

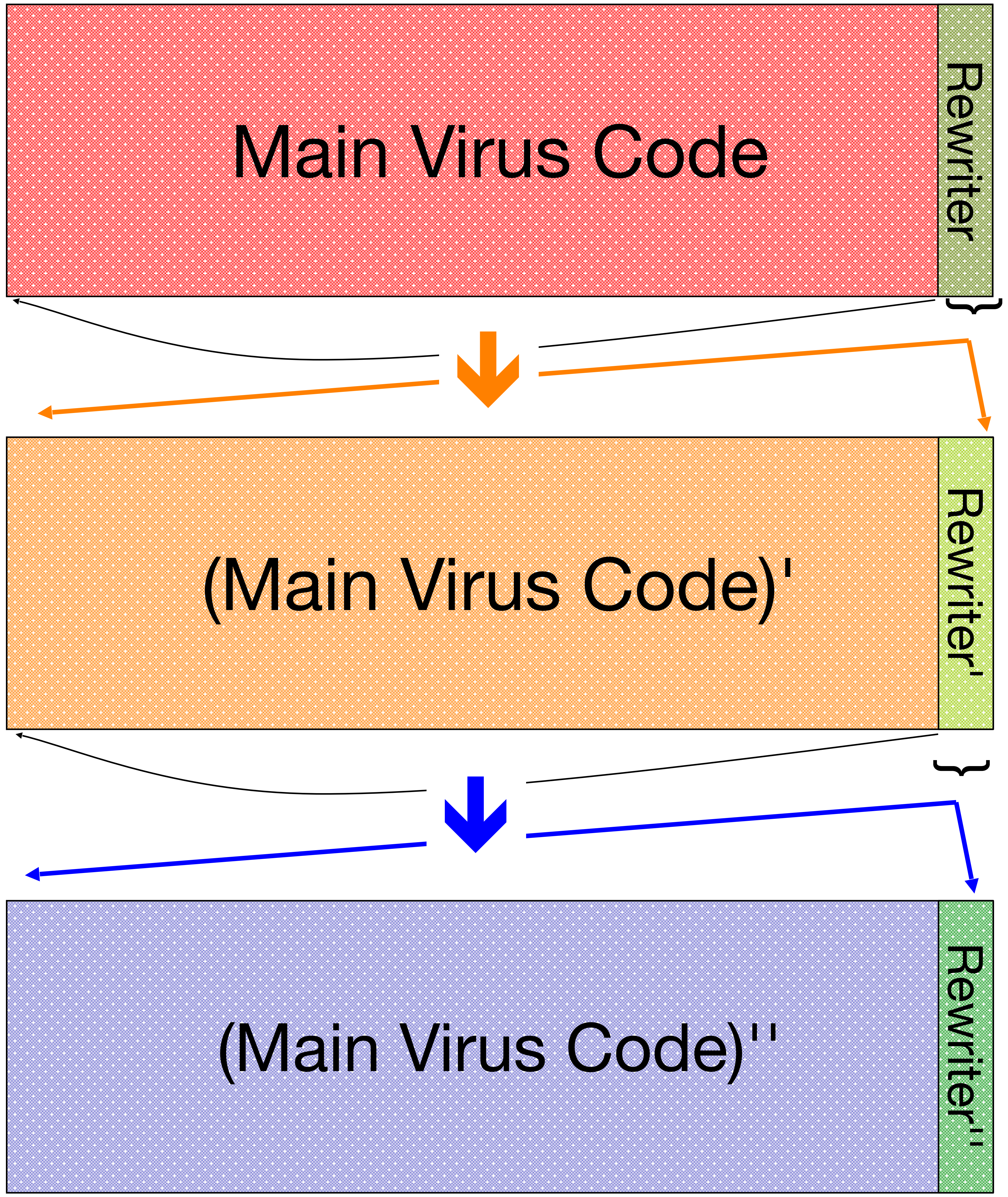
Arms Race: Polymorphic Code

- 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!

Metamorphic Propagation



When ready to propagate, virus invokes a randomized *rewriter* to construct new but semantically equivalent code (*including the rewriter*)

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

Malcode Wars and the Halting Problem...

- Cyberwars are not won by solving the halting problem...
  Cyberwars are won by making some other poor sod solve the halting problem!!!
  - In the limit, it is ***undecidable*** to know "is this code bad?"
- Modern focus is instead "is this code ***new?***"
  - Use a secure cryptographic hash (so sha-256 not md5)
  - Check hash with central repository:
    If ***not*** seen before, treat binary as inherently more suspicious
- Creates a bind for attackers:
  - Don't make your code *morphic:
    Known bad signature detectors find it
  - Make your code *morphic:
    It always appears as new and therefore ***inherently*** suspicious

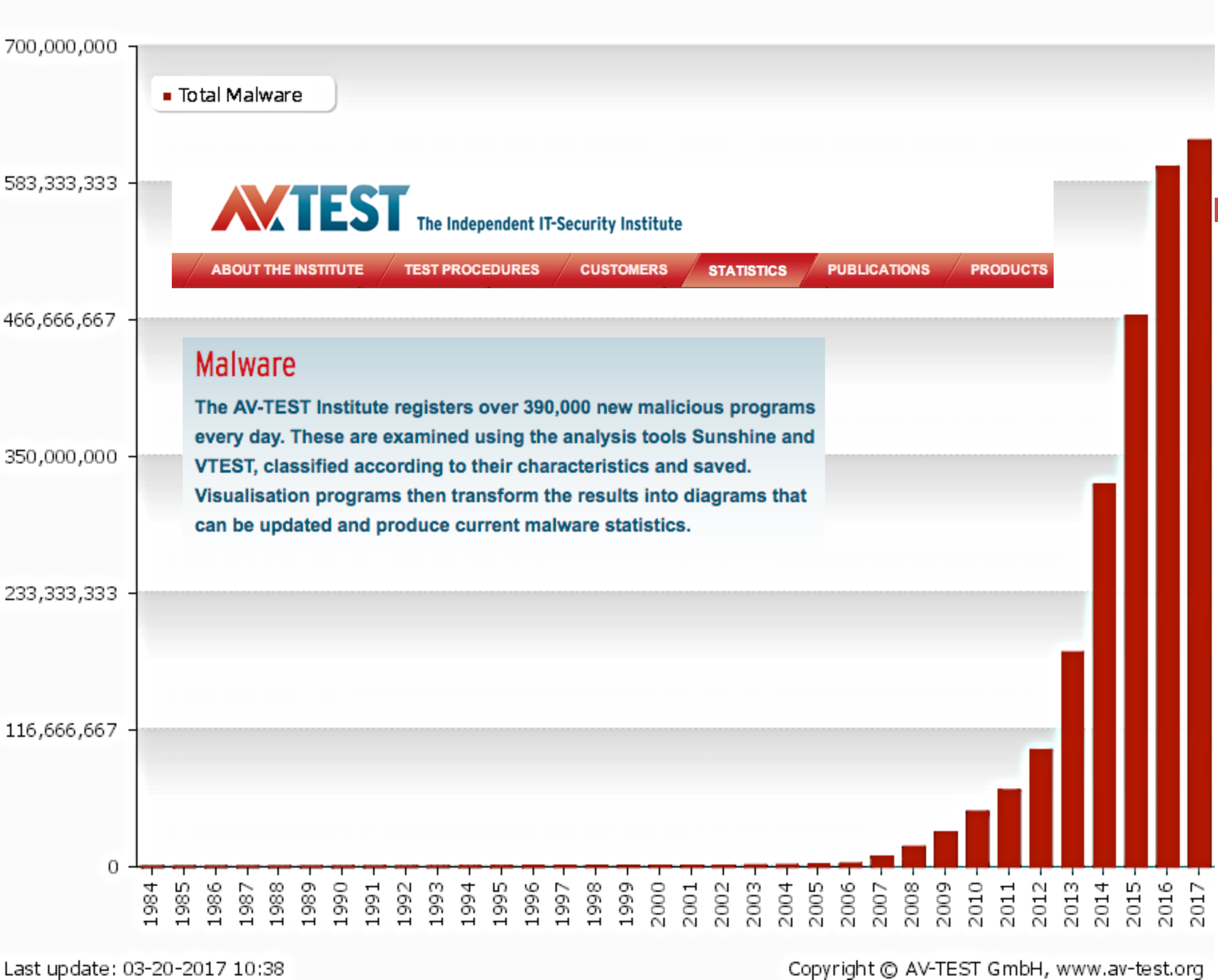
CYBERWARS ARE NOT WON BY SOLVING THE HALTING PROBLEM
CYBERWARS ARE WON BY MAKING SOME OTHER POOR SOD SOLVE THE HALTING PROBLEM
imgflip.com

Creating binds is very powerful...

- You have a detector D for some bad behavior...
  - So bad-guys come up with a way of avoiding the detector D
- So come up with a detection strategy for ***avoiding detector D***
  - So to avoid ***this*** detector, the attacker ***must not*** try to avoid D
- When you can do it, it is very powerful!

How Much Malware Is Out There?

- A final consideration re polymorphism and metamorphism:
  - Presence can lead to mis-counting 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 huge malware populations exist is in the vendors' own interest)



Malware

The AV-TEST Institute registers over 390,000 new malicious programs every day. These are examined using the analysis tools Sunshine and VTEST, classified according to their characteristics and saved. Visualisation programs then transform the results into diagrams that can be updated and produce current malware statistics.

Last update: 03-20-2017 10:38

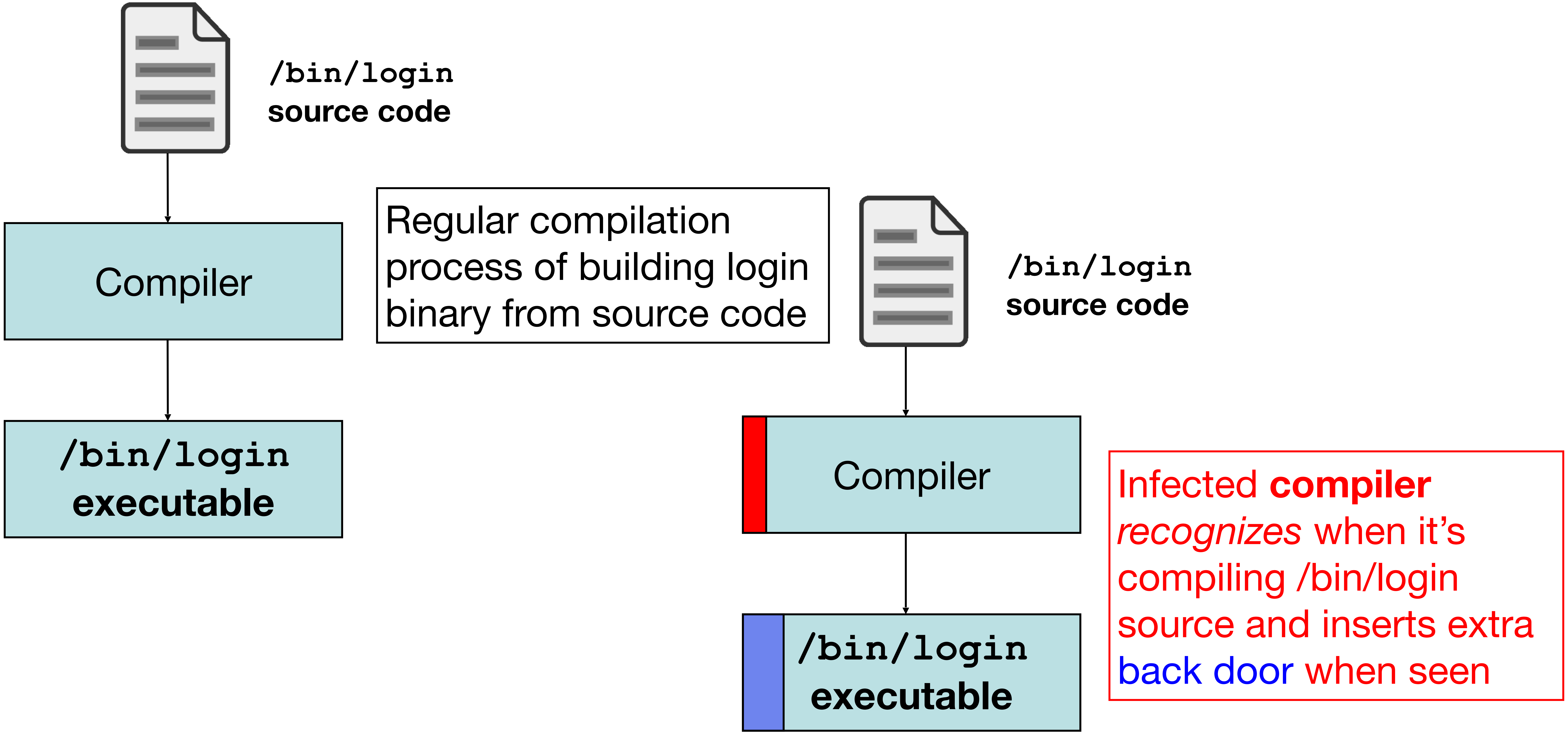
Copyright © AV-TEST GmbH, www.av-test.org

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?
  - "Game over man, Game Over!"
  - “Dust off and 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)

Infection Cleanup, con't

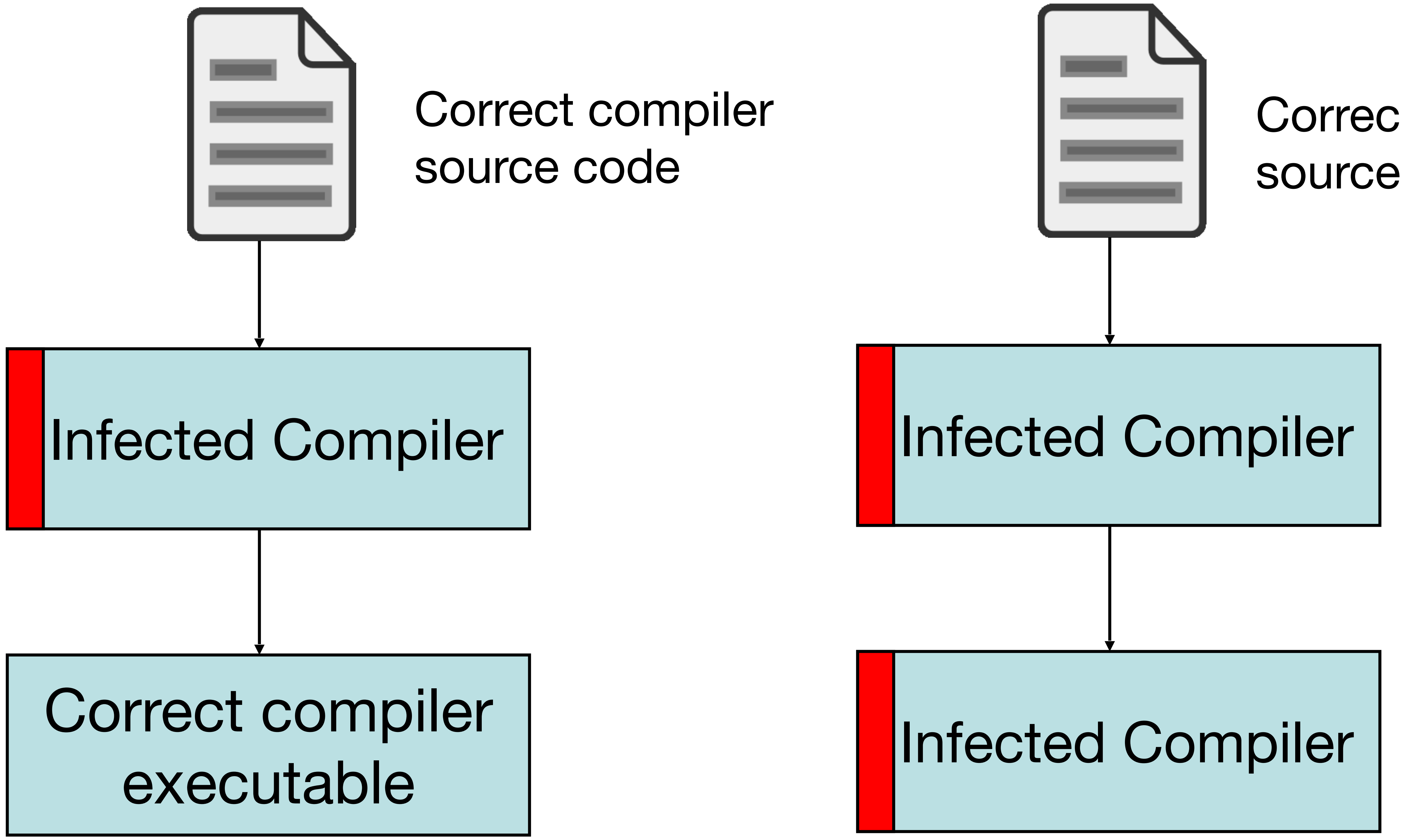
- If we have complete source code for system, we could rebuild from that instead, couldn't we?
- No!
- Suppose forensic analysis shows that virus introduced a backdoor in /bin/login executable
  - (Note: this threat isn't specific to viruses; applies to any malware)
- Cleanup procedure: rebuild /bin/login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



Regular compilation process of building login binary from source code

Infected **compiler** *recognizes* when it’s compiling /bin/login source and inserts extra back door when seen

~~No problem: first step, rebuild the compiler so it's uninfected~~


Correct compiler source code
Infected Compiler
Correct compiler executable
Correct compiler source code
Infected Compiler
Infected Compiler

Oops - infected compiler recognizes when it's compiling its own source and inserts the infection!

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

More On "Rootkits"

- If you control the operating system...
  - You can hide extremely well
- EG, your malcode is on disk...
  - So it will persist across reboots
- But if you try to ***read the disk***...
  - The operating system just says "Uhh, this doesn't exist!"

Even More Places To Hide!

- In the BIOS/EFI Firmware!
  - So you corrupt the BIOS which corrupts the OS...
  - Really hard to find:
    Defense, ***only*** run cryptographically signed BIOS code as part of the Trusted Base
- In the disk controller firmware!
  - So the master boot record, when read on boot up corrupts the OS...
  - But when you try to read the MBR later... It is just "normal"
  - Again, defense is ***signed code:*** The Firmware will only load a signed operating system
    - Make sure the disk itself is ***not trusted!***

Robust Rootkit Detection: Detect the act of hiding...

- Do an "in-system" scan of the disk...
  - Record it to a USB drive
- Reboot the system with trusted media
  - So a known good operating system
- Do the same scan!
  - If the scans are different, you found the rootkit!
- For windows, you can also do a "high/low scan" on the Registry:
  - Forces the bad guy to understand the registry as well as Mark Russinovich (the guy behind Sysinternals who's company Microsoft bought because he understood the Registry better than Microsoft's own employees!)
- Forces a bind on the attacker:
  - Hide and persist? You can be detected
  - Hide but don't persist? You can't survive reboots!

Which Means ***Proper*** Malcode Cleanup...

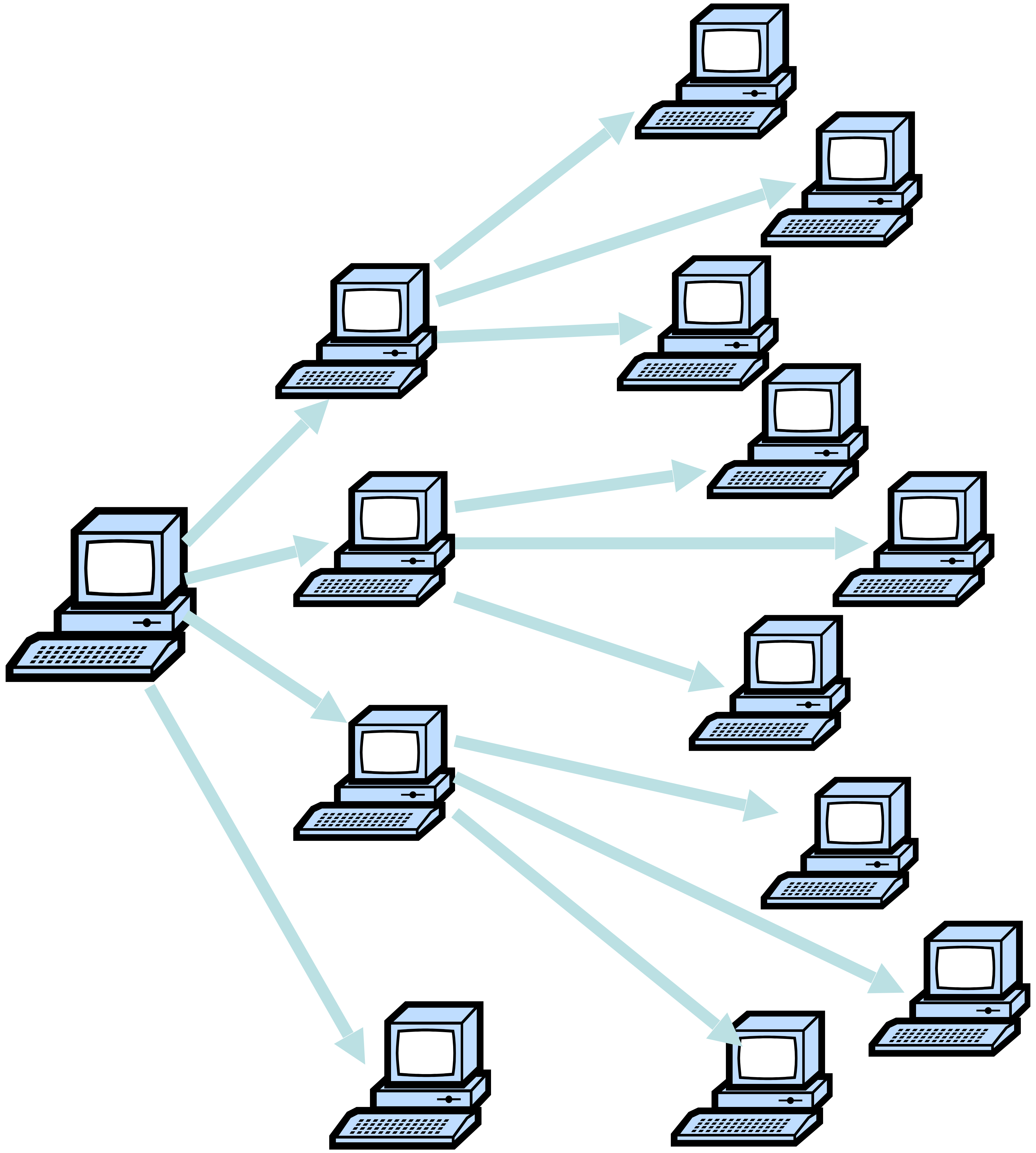
Large-Scale Malware

- 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

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.

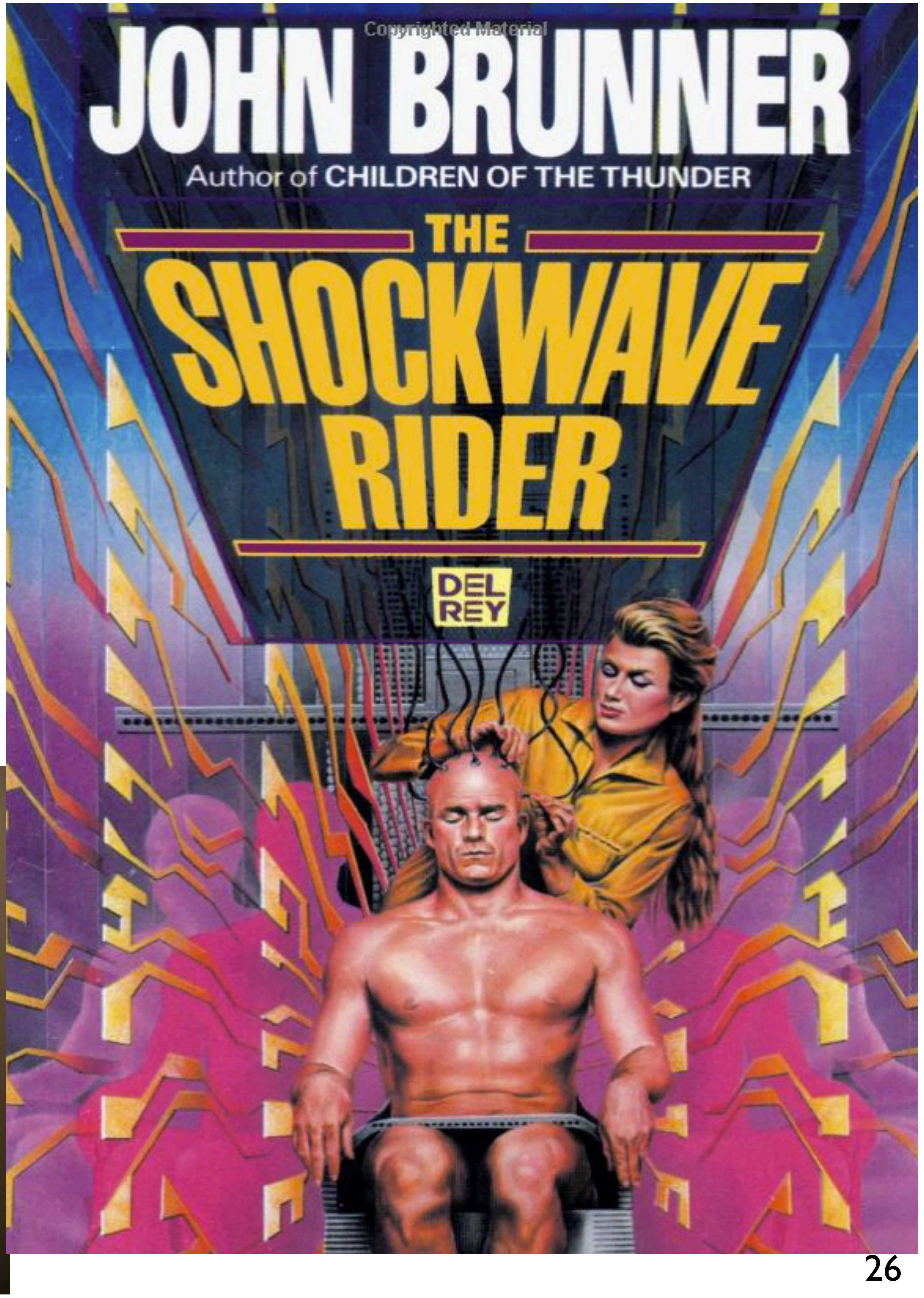


Worms

- 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?
    - One common approach: random scanning of 32-bit IP address space
      - Generate pseudo-random 32-bit number; try connecting to it; if successful, try infecting it; repeat
    - But for example “search worms” use Google results to find victims
  - How does worm get code to automatically run?
    - One common approach: buffer overflow ⇒ code injection
    - But for example a web worm might propagate using XSS

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

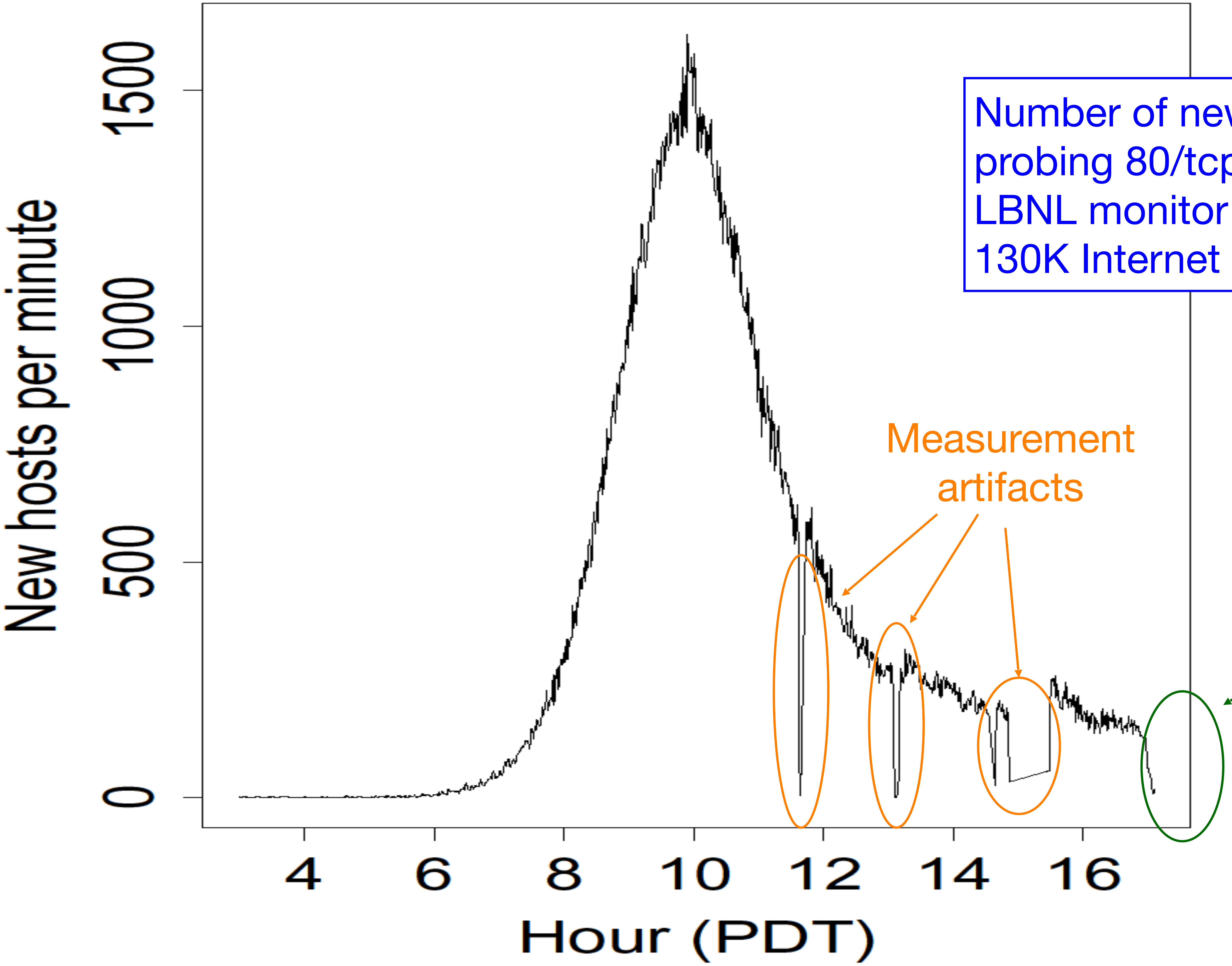
- Payload part 2: check day-of-the-month and …
  - … 1st through 20th of each month: spread
  - … 20th through end of each month: attack
    - Flooding attack against 198.137.240.91 …
    - … i.e., *www.whitehouse.gov*
- Spread: via *random scanning* of 32-bit IP address space
  - Generate pseudo-random 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



Number of new hosts probing 80/tcp as seen at LBNL monitor of 130K Internet addresses

Measurement artifacts

The worm dies off globally!

Nick's Reaction to Code Red

- Come on, we are computer people…
  - What do we do that EVER takes 13 hours?!?!?
- How to speed up
  - Preseed to skip the initial ramp-up
  - Scan faster (100x/second rather than 10x)
  - Scan smarter
    - Self-coordinated scanning techniques with shutoff strategies
  - Validated in ***simulation!***
- The “Warhol Worm” concept…
  - Implications that any worm defense needs to be automatic
  - "How to 0wn the Internet in your Spare Time"

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 250 Internet addresses per unit time, and 2% of Internet addresses run a vulnerable (maybe already infected) server ⇒ β = 5
    - For scanning worms, larger (= denser) vulnerable pop. ⇒ higher β ⇒ faster worm!

$$\boxed{\begin{array}{c} N = S(t) + I(t) \\ S(0) = I(0) = N/2 \end{array}}$$

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

Computing How An Epidemic Progresses

- In continuous time:

$$\underbrace{\frac{dI}{dt}}_{\text{Increase in \# infectibles per unit time}} = \underbrace{\beta \cdot I}_{\text{Total attempted contacts per unit time}} \cdot \underbrace{\frac{S}{N}}_{\text{Proportion of contacts expected to succeed}}$$

Increase in # infectibles per unit time

Total attempted contacts per unit time

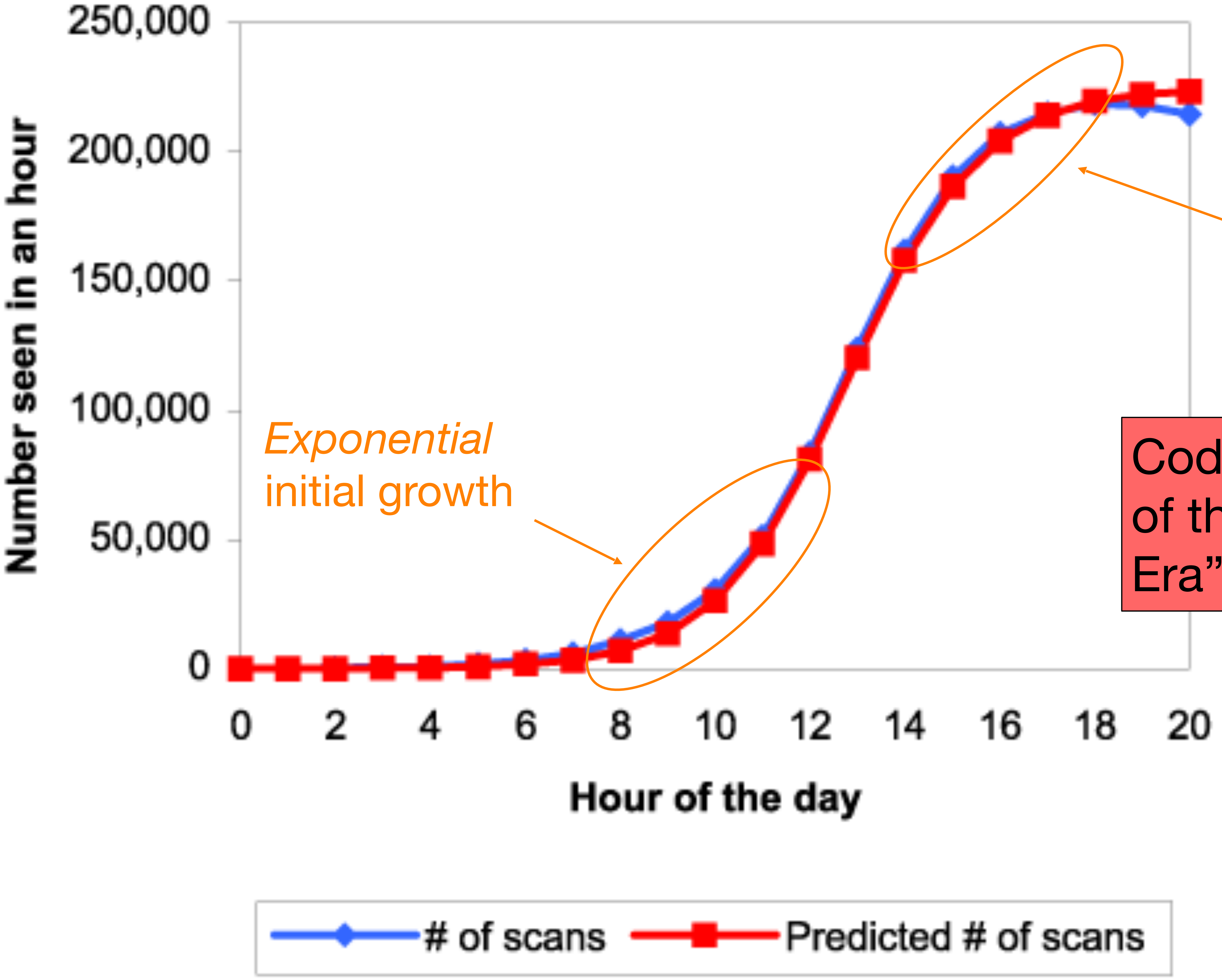
Proportion of contacts expected to succeed

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

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

Fraction infected grows as a *logistic*

Fitting the Model to “Code Red”



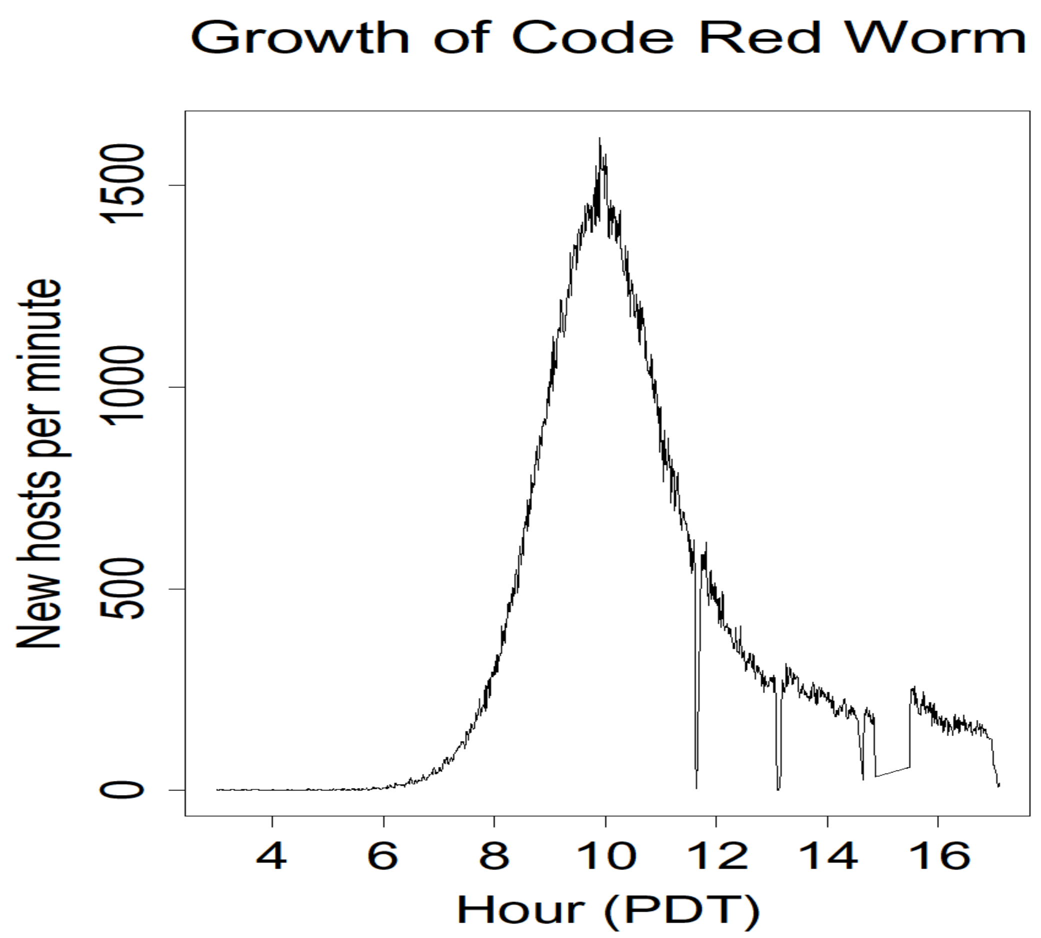
Exponential initial growth

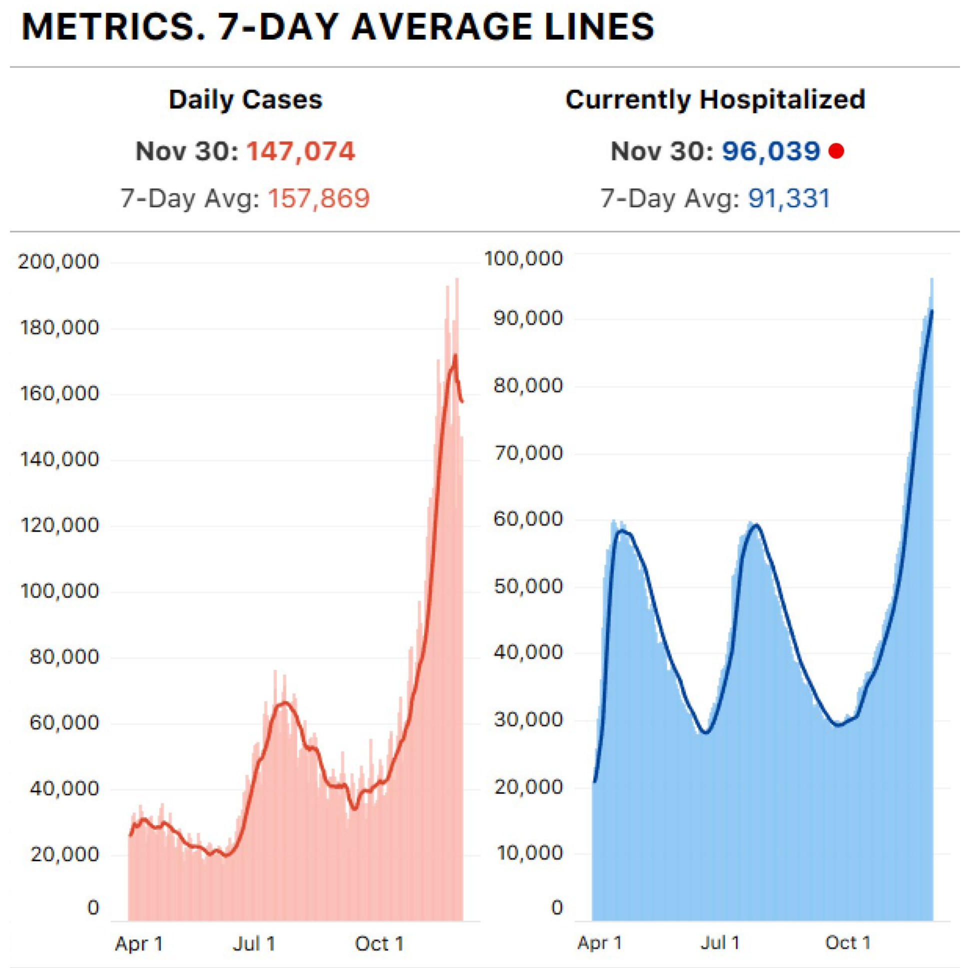
Growth slows as it becomes harder to find new victims!

Code Red = first worm of the “Modern Worm Era”, circa 2001.

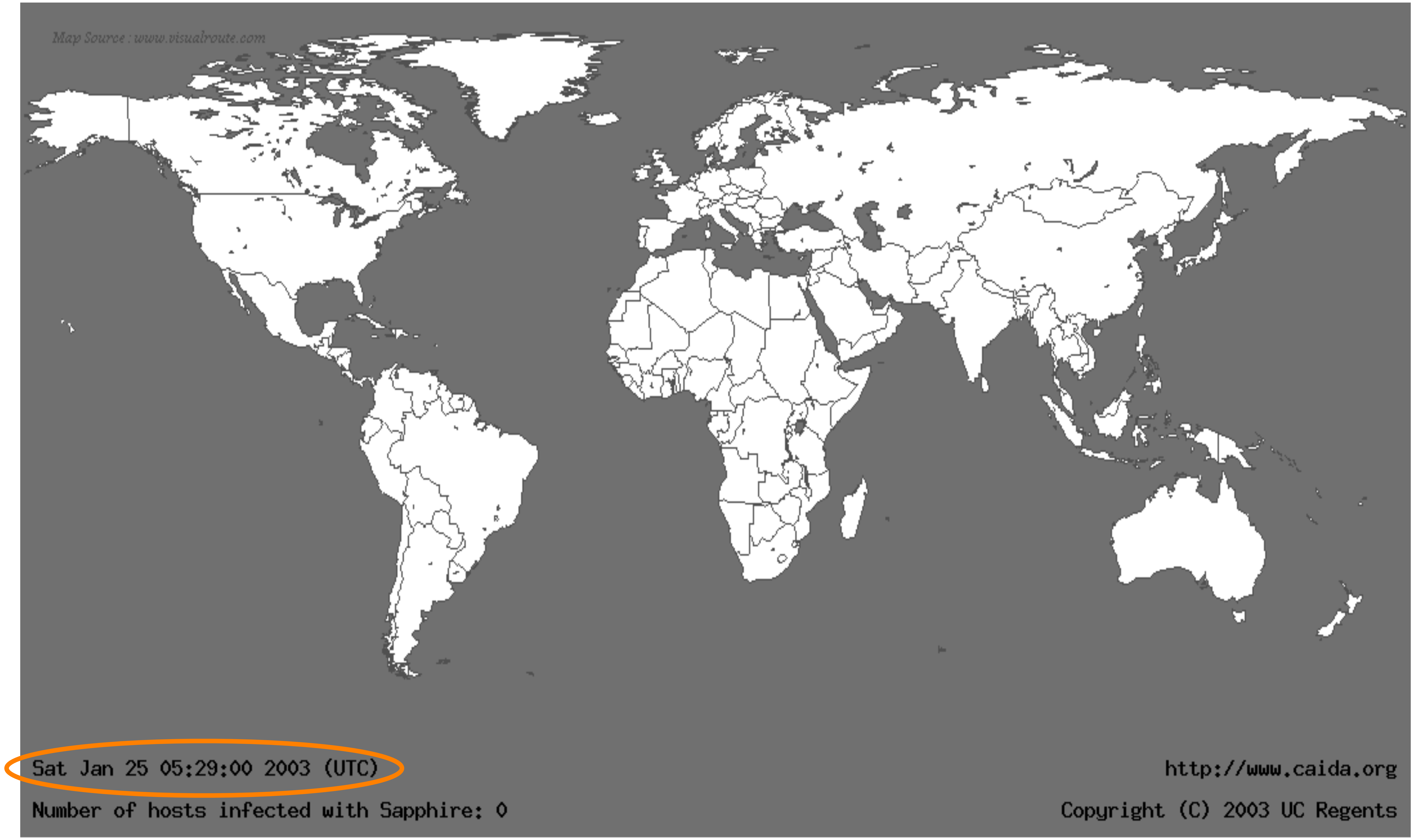
And We See This in COVID too....

- COVID's spread is a bit more complicated
  - Dominated by super-spreader events combined with in-household spread
  - Makes it a bit burstier/noisier
- But you do get these infection explosions and then when controls get into place, it ramps back down...





Life Just Before Slammer



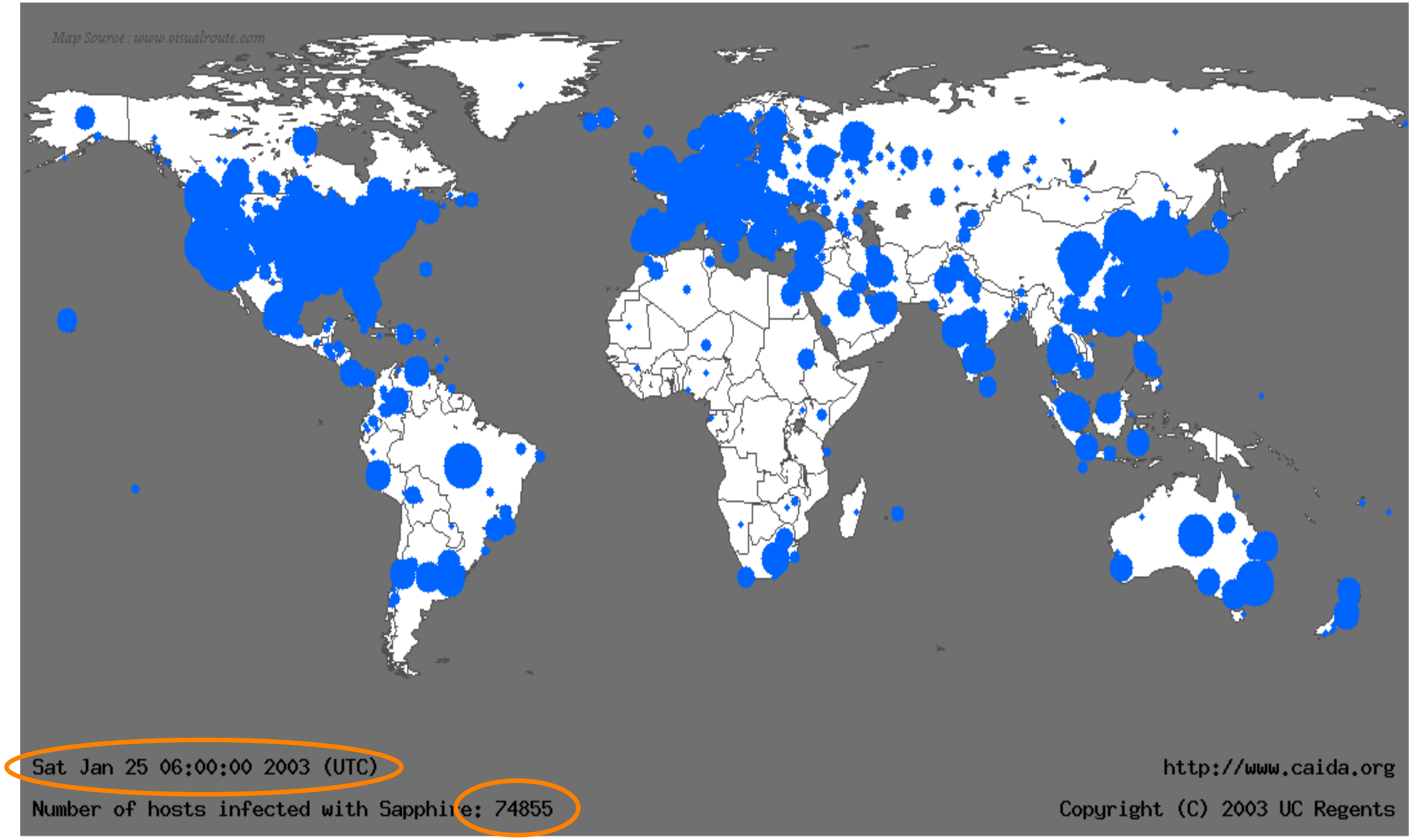
Sat Jan 25 05:29:00 2003 (UTC)

Number of hosts infected with Sapphire: 0

http://www.caida.org

Copyright (C) 2003 UC Regents

Life 10 Minutes After Slammer

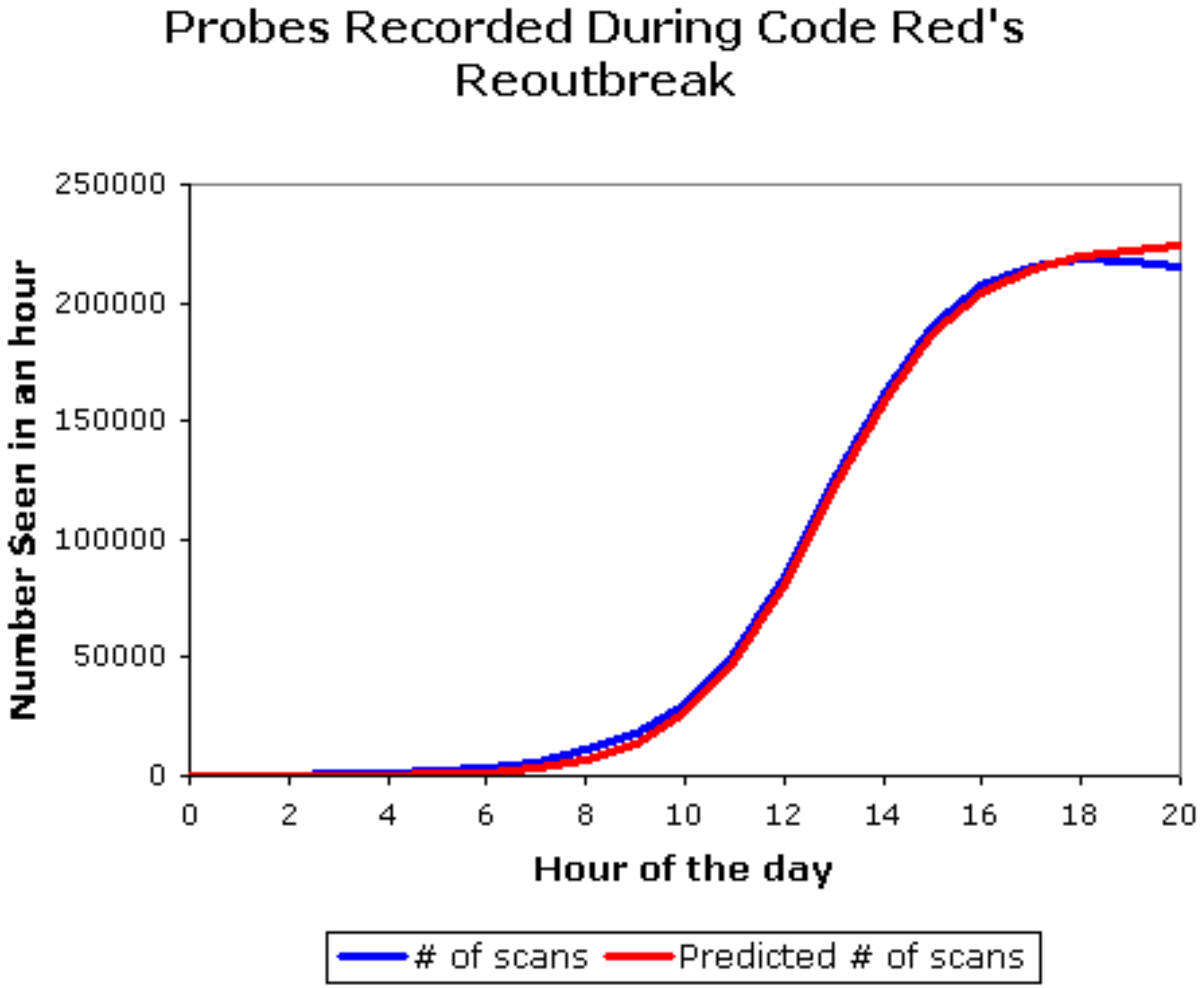

Map Source : www.visualroute.com
Sat Jan 25 06:00:00 2003 (UTC)
Number of hosts infected with Sapphire: 74855
http://www.caida.org
Copyright (C) 2003 UC Regents

Going Fast: Slammer

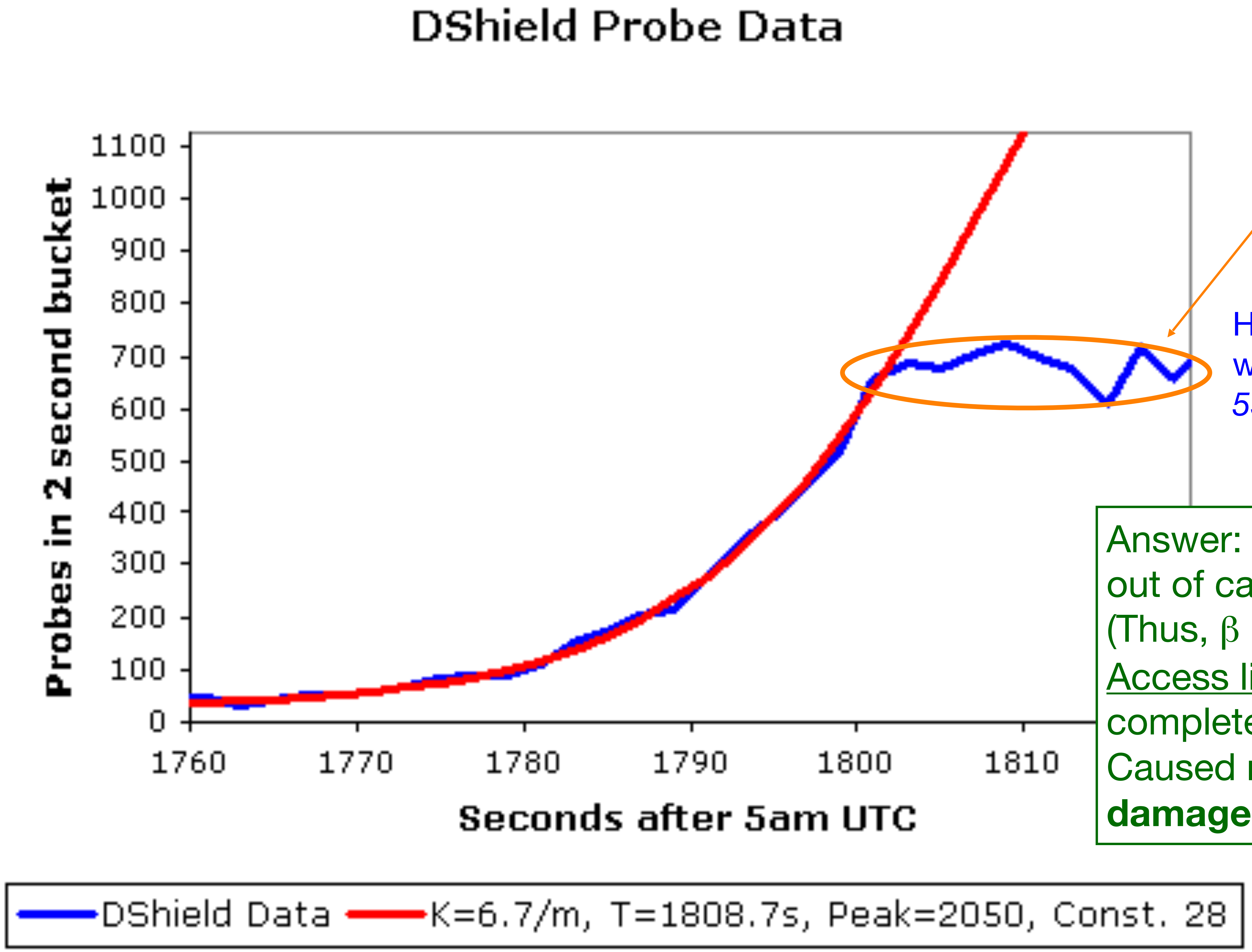
- Slammer exploited connectionless UDP service, rather than connection-oriented TCP
- Entire worm fit in a single packet!
- ⇒ When scanning, worm could “fire and forget”

  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**.

Witty...

- A worm like Slammer but with a twist...
  - Targeted network intrusion detection sensors!
  - Released ~36 hours after vulnerability disclosure and patch availability!
- Payload wasn't just spreading, however...

  -
    ```
    while true {
      for i := range(20000){
        send self to random target;
      }
      select random disk (0-7)
      if disk exists {
         select random block, erase it;
      }}
    ```

Stuxnet

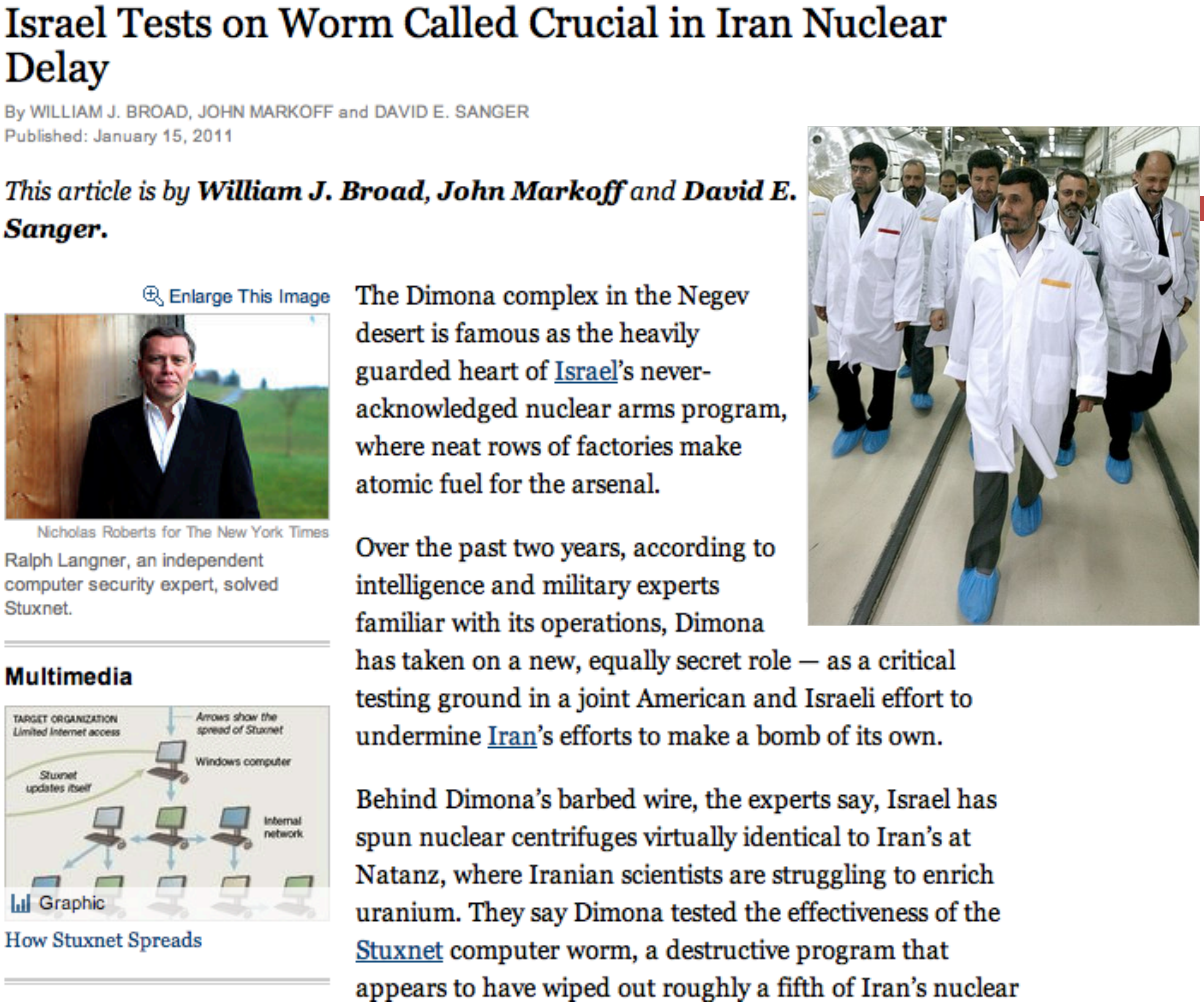
- Discovered July 2010. (Released: Mar 2010?)
- Multi-mode spreading:
  - Initially spreads via USB (virus-like)
  - Once inside a network, quickly spreads internally using Windows RPC scanning
- Kill switch: programmed to die June 24, 2012
- Targeted SCADA systems
  - Used for industrial control systems, 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.

Multimedia

How Stuxnet Spreads

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

The "Toddler" Attack Payload...

- Stuxnet was very carefully engineered...
  - Designed to only go off under ***very specific*** circumstances
- But industrial control systems are inherently vulnerable
  - They consist of sensors and actuators
  - And safety is a ***global*** property
- Generic Boom:
  - At zero hour, the payload sees that it is on control system:
    map the sensors and actuators, see which ones are low speed vs high speed
  - T+30 minutes: Start replaying sensor data, switch actuators in low-speed system
  - T+60 minutes: Switch all actuators at high speed...
- This ***has been done***:
  A presumably Russian test attack on the Ukranian power grid! ("CrashOverride" attack)

Then who "WannaCry"?

- The modern way of making profit from computer crime: Ransomware!
  - "Give us X Bitcoin or you'll never see your data again!"
  - The North Koreans apparently are doing this as a matter of government policy?!?!?
- So lets combine a ransomware payload with a self-spreading worm...
  - Then sit back and PROFIT!!!!
- Oh, wait...
  - The worm escaped early and the ransomware payload wasn't fully tested!
  - A ton of work for absolutely no profit:
    🇰🇵 -> 😭
    Everyone else -> 🤦🏼‍♂️🤣 if it didn't happen to disrupt a lot of businesses and destroy a lot of data.

And NotPetya...

- NotPetya was a worm deliberately launched by Russia against Ukraine
  - Initial spread: A corrupted update to MeDoc Ukranian Tax Software
  - Then spread within an institution using "Eternal Blue" (Windows vulnerability) and "Mimikatz"
    - Mimikatz is way ***way more*** powerful:
      Takes advantage of windows transitive authorization...
    - IF you are running on the admin's machine, you can take over the domain controller
    - IF you are running on the domain controller, you can take over ***every computer***!!!
- Then wiped machines as fake ransomware
  - Give a veneer of deniability...
  - Shut down Mersk and many other global companies!

And Overall Taxonomy of Spread

- Scanning
  - Look for targets
  - Can be bandwidth limited
- "Target Lists"
  - Pregenerated (Hitlist)
  - On-the-host (Topological)
  - Query a third party server that lists servers (Metaserver)
- Passive
  - Wait for a contact: Infect with the counter-response
- More detailed taxonomy here:
  - http://www.icir.org/vern/papers/taxonomy.pdf

Putting CS161 in Context: Nick's Self Defense Strategies...

- ***How*** and ***why*** do I protect myself online and in person...
  - ***How*** I decide what to prepare for (and what not to prepare for)
  - ***Why*** I've drunk the Apple Kool-Aid™
  - ***Why*** I use my credit card everywhere but not a debit card
- And my future nightmares:
  - What do I see as the security problems of tomorrow...

My Personal Threats: The Generic Opportunist

- There are a ***lot*** of crooks out there
  - And they are rather organized...
- But at the same time, these criminals are generally economically rational
  - So ***this*** is a bear race: I don't need perfect security, I just need ***good enough*** security
- I use this to determine security/convenience tradeoffs all the time
  - So no password reuse (use a password manager instead)
  - Full disk encryption & passwords on devices:
    Mitigates the damage from theft
  - Find my iPhone turned on:
    Increases probability of theft recovery

My Personal Threats: The ***Lazy*** Nation State

- OK, I'm a high ***enough*** profile to have to worry about the "Advanced Persistent Threats"...
  - Trying for a reasonably high profile on computer policy issues
  - A fair amount of stuff studying the NSA's toys and other nation-state tools
  - But only at the Annoying Pestilent Teenager level:
    I'm worth some effort but not an extraordinary amount
- So its only ***slightly*** more advanced than the everyday attackers...
  With one ***huge*** exception: Crossing borders
  - Every nation maintains the right to conduct searches of all electronic contents at a border checkpoint

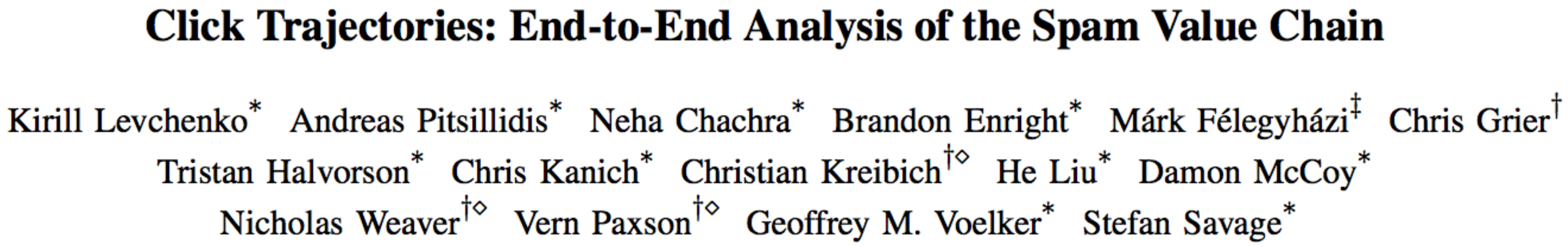
My Border Crossing Policy: Low Risk Borders

- Not very sensitive borders: Canada, Europe, US, etc...
  - I use full disk encryption with strong passwords on all devices
    - Primary use is to prevent theft from also losing data
  - I have a ***very robust*** backup strategy
    - Time machine, archived backups in a safe deposit box, working sets under version control backed up to remote systems...
- So, as the plane lands:
  - Power off my devices
    - Device encryption is only ***robust*** when you aren't logged in
  - Go through the border
- If my devices get siezed...
  - "Keep it, we'll let the lawyers sort it out"

High Risk Borders

- Middle East or, if, god forbid, I visit China or Russia...
  - Need something that doesn't just resist compromise but can also ***tolerate compromise***
- A "burner" iPhone SE with a Bluetooth keyboard
  - The cheapest secure device available
  - Set it up with ***independent*** computer accounts for both Google and Apple
    - Temporarily forward my main email to a temporary gmail account
    - All workflow accessible through Google apps on that device
  - Bluetooth keyboard does leak keystrokes, so don't use it for passwords but its safe for everything else
- Not only is this device very hard to compromise...
  - But there is very low value in ***successfully compromising it***: The attacker would only gain access to dummy accounts that have no additional privileges
- And bonus, I'm not stuck dragging a computer to the ski slopes in Dubai...
  - Since the other unique threat in those environments is the "Evil maid" attack

My Personal Threats: The Russians... Perhaps



Click Trajectories: End-to-End Analysis of the Spam Value Chain

Kirill Levchenko[*] Andreas Pitsillidis[*] Neha Chachra[*] Brandon Enright[*] Márk Félegyházi[‡] Chris Grier[†]
Tristan Halvorson[*] Chris Kanich[*] Christian Kreibich[†⋄] He Liu[*] Damon McCoy[*]
Nicholas Weaver[†⋄] Vern Paxson[†⋄] Geoffrey M. Voelker[*] Stefan Savage[*]

- This is the paper that killed the Viagra® Spam business
  - A $100M a year set of organized criminal enterprises in Russia...
    And they put the ***organized*** in organized crime...
- I've adopted a ***detection and response*** strategy:
  - The Russians have higher priority targets: The first authors, the last authors, and Brian Krebs
  - If anything suspicious happens to Brian, Kirill, or Stefan, ***then*** I will start sleeping with a rifle under my bed

Excluded Threats: Sorta…

- Intimate Partner Threats…
  - But I’ve had at least one colleague caught up with that.
- Agressive Nation States…
  - $50M will buy the latest version of Pegasus malcode
- The US government…
  - The surveillance powers of the US government are awesome and terrifying to behold…

Passwords and 2-Factor....

- I ***love*** security keys:
  - I have one in each of my main computers...
    and one on the keychain
- ANY site that supports multiple security keys has that as the primary 2-factor method
  - Both more convenient ***and*** more secure than the alternatives...
- I also religiously use a password manager
  - "Credential stuffing" is the biggest threat individuals face
- I personally use 1password, but others are equally good
  - If I was in charge of UC IT I'd get a site license for everyone...
    And seriously nag people to use it every time they log into the single sign on

The Apple Kool-Aid...

- The iPhone is perhaps the most secure commodity device available...
  - Not only does it receive patches but since the 5S it gained a dedicated cryptographic coprocessor
- The ***Secure Enclave Processor*** is the trusted base for the phone
  - Even the main operating system isn't fully trusted by the phone!
- A dedicated ARM v7 coprocessor
  - Small amount of memory, a true RNG, cryptographic engine, etc...
  - Important: A collection of ***randomly*** set fuses
    - Should not be able to extract these bits without taking the CPU apart:
      Even the Secure Enclave can only use them as keys to the AES engine, not read them directly!
  - But bulk of the memory is shared with the main CPU
- GOOD documentation:
  - The iOS security guide is something you should at least skim….
    I find that the design decisions behind how iOS does things make ***great*** final exam questions
- But it isn't perfect: Nation-state actors will pay big $ for exploits
  - So keep it patched

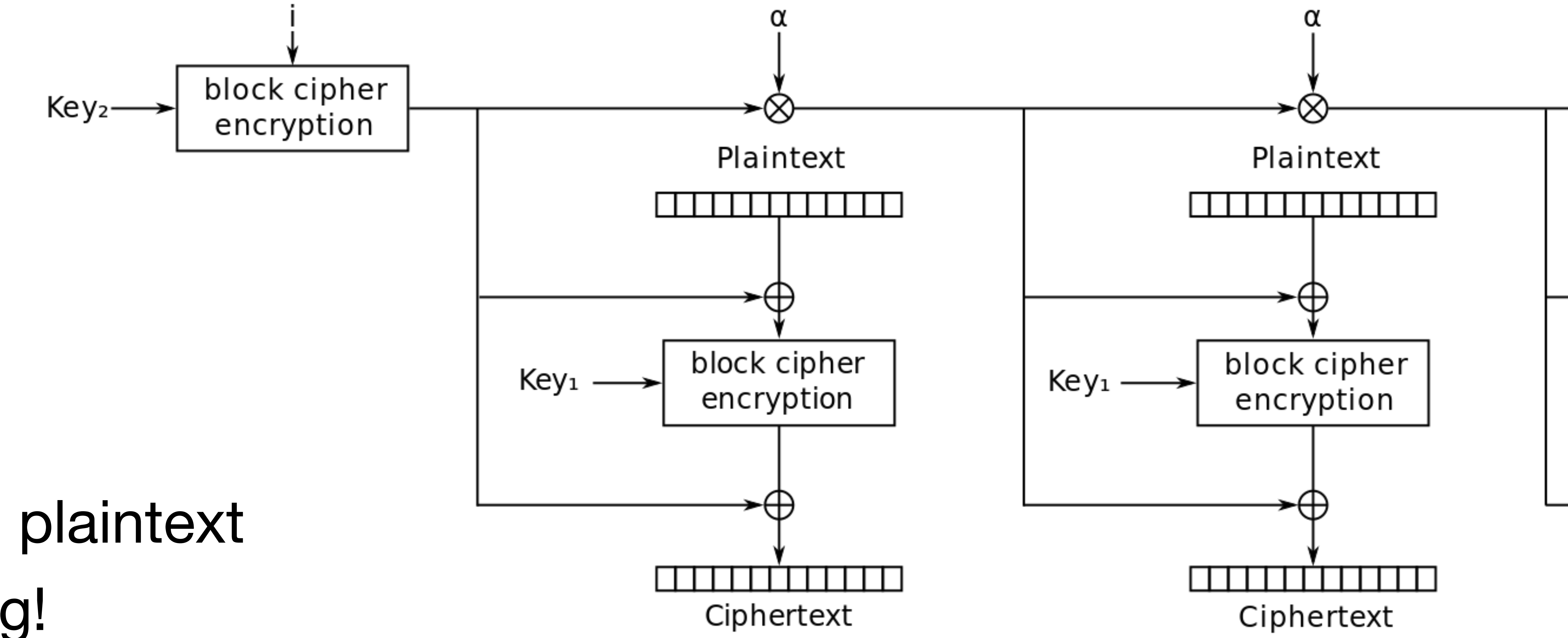
The Roll of the SEP... Things *too important* to allow the OS to handle

- Key management for the encrypted data store
  - The CPU has to ask for access to data!
- Managing the user's passphrase and related information
- User authentication:
  - ***Encrypted*** channel to the fingerprint reader/face recognition camera
- Storing credit cards
  - ApplePay is cheap for merchants ***because it is secure***:
    Designed to have very low probability of fraud!

AES-256-XEX mode

- A ***confidentality-only*** mode developed by Phil Rogaway...
  - Designed for encrypting data within a filesystem block ***i***
    - Known plaintext, when encrypted, can't be replaced to produce known output, only "random" output
  - Within a block: Same cypher text implies different plaintext
  - Between blocks: Same cypher text implies nothing!
  - α is a galios multiplication and is very quick:

    In practice this enables parallel encryption/decryption
- Used by the SEP to encrypt its own memory...
  - Since it has to share main memory with the main processor
- Opens a limited attack surface from the main processor:
  - Main processor can replace 128b blocks with ***random*** corruption



XEX mode encryption

User Passwords...

- Data is encrypted with the user's password
  - When you power on the phone, most data is completely encrypted
- The master key is PBKDF2(password || on-chip-secret)
  - So you need ***both*** to generate the master key
  - Some other data has the key as F(on-chip-secret) for stuff that is always available from boot
- The master keys encrypt a block in the flash that holds all the other keys
  - So if the system can erase this block effectively it can erase the phone by erasing just one block of information
- Apple implemented ***effaceable storage***:
  - After x failures, OS command, whatever...
    Overwrite that master block in the flash securely
  - Destroy the keys == erase everything!

Background: FBI v Apple

- A "terrorist" went on a rampage with a rifle in San Bernardino...
  - Killed several people before being killed in a battle with police
- He left behind a work-owned, passcode-locked iPhone 5 in his other car...
- The FBI ***knew*** there was no valuable information on this phone
  - But never one to refuse a good test case, they tried to compel Apple in court to force Apple to unlock the phone...
- Apple has serious security on the phone
  - Effectively everything is encrypted with PBKDF2(PW||on-chip-secret):
    >128b of randomly set microscopic fuses
    - Requires that ***any*** brute force attack either be done on the phone or take apart the CPU
  - Multiple timeouts:
    - 5 incorrect passwords -> starts to slow down
    - 10 incorrect passwords -> optional (opt-in) erase-the-phone

What the FBI wanted...

- Apple provides a ***modified*** version of the operating system for the Secure Enclave which...
  - Removes the timeout on all password attempts
  - Enables password attempts through the USB connection
  - Enables an ***on-line*** brute force attack..
    but with a 4-digit PIN and 10 tries/second, you do the math...
- Apple ***cryptographically signs the rogue OS version!***
  - A horrific precedent:
    This is ***requiring*** that Apple both create a malicious version of the OS and sign it
    - If the FBI could compel Apple to do this, the NSA could too...
      It would make it ***impossible*** to trust software updates!

Updating the SEP To Prevent This Possibility...

- The SEP will only accept updates ***signed by Apple***
  - But an updated SEP could exfiltrate the secret to enable an offline attack
- The FBI previously asked for this capability against a non-SEP equipped phone
  - "Hey Apple, cryptographically sign a corrupted version of the OS so that we can brute-force a password"
- How to prevent the FBI from asking again?
- Now, an OS update (either to the base OS and/or the SEP) requires the user to be logged in ***and input the password***
  - "To rekey the lock, you must first unlock the lock"
  - The FBI can only even ***attempt*** to ask before they have possession of the phone since once they have the phone they must also have the passcode
  - So when offered the chance to try again with a "Lone Wolf's" iPhone in the Texas church shooting, they haven't bothered
- At this point, Apple has now gone back and allows auto-updates for the base OS
  - (but probably not the SEP)

The Limits of the SEP... The host O/S

- The SEP can keep the host OS from accessing things it shouldn't...
  - Credit cards stored for ApplePay, your fingerprint, etc...
- But it can't keep the host OS from things it is supposed to access
  - All the user data when the user is logged in...
- So do have to rely on the host OS as part of ***my*** TCB
  - Fortunately it is updated continuously when vulnerabilities are found
    - Apple has responded to the discovery of very targeted zero-days in <30 days
  - And Apple has both good sandboxing of user applications and a history of decent vetting
    - So the random apps are ***not*** in the Trusted Base.

The SEP and Apple Pay

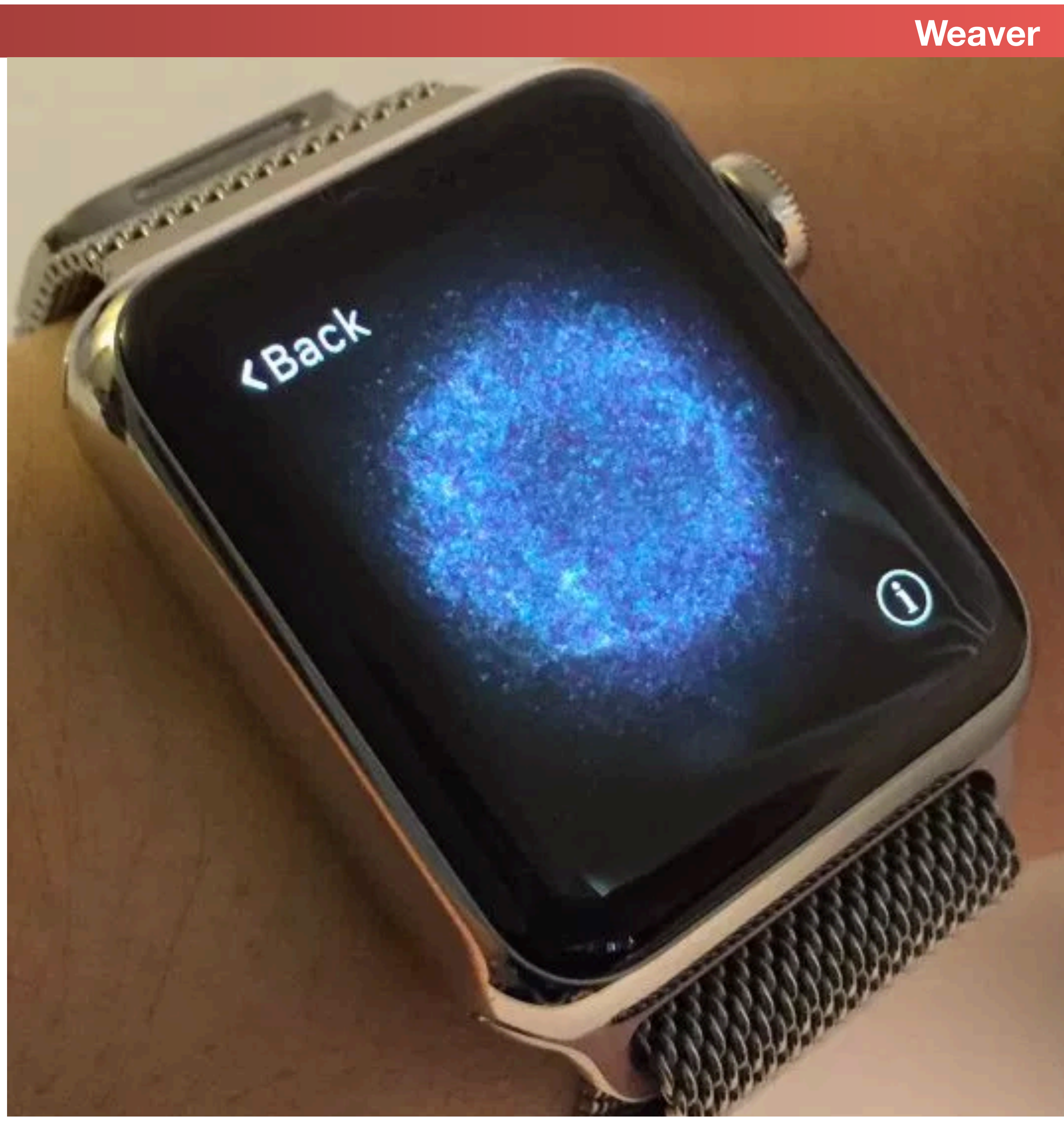
- The SEP is what makes ApplePay possible
  - It handles the authentication to the user with the fingerprint reader/face reader
    - Verifies that it is the user not somebody random
  - It handles the emulation of the credit card
    - A "tokenized" Near Field Communication (NFC) wireless protocol
    - And a tokenized public key protocol for payments through the app
- ***Very hard*** to conduct a fraudulent transaction
  - Designed to enforce user consent at the SEP
- ***Disadvantage***: The fingerprint reader is part of the trust domain
  - Which means you need special permission from Apple to replace the fingerprint reader when replacing a broken screen

I ***love*** ApplePay...

- It is a ***faster*** protocol than the chip-and-signature
  - NFC protocol is designed to do the same operation in less time because the protocol is newer
- It is a ***more secure*** protocol than NFC on the credit card
  - Since it actually enforces user-consent
- It is more ***privacy sensitive*** than standard credit card payments
  - Generates a unique token for each transaction:
    Merchant is not supposed to link your transactions
- Result is its low cost:
  - Very hard to commit fraud -> less cost to transact
- I use it on my watch all the time

Transitive Trust in the Apple Ecosystem...

- The most trusted item is the iPhone SEP
  - Assumed to be rock-solid
  - Fingerprint reader/face reader allows it to be convenient
- The watch trusts the phone
  - The pairing process includes a cryptographic key exchange mediated by close proximity and the camera
  - So Unlock the phone -> Unlock the watch
- My computer trusts my watch
  - Distance-bounded cryptographic protocol
  - So my watch unlocks my computer
- Result? I don't have to keep retyping my password
  - Allows the use of ***strong passwords everywhere*** without driving myself crazy!



Credit Card Fraud

- Under US law we have very good protections against fraud
  - Theoretical $50 limit if we catch it quickly
  - $0 limit in practice
- So cost of credit card fraud for me is the cost of recovery from fraud
  - Because fraud ***will happen***:
  - The mag stripe is all that is needed to duplicate a swipe-card
    - And you can still use swipe-only at gas pumps and other such locations
  - The numbers front and back is all that is needed for card-not-present fraud
    - And how many systems
- What are the recovery costs?
  - Being without the card for a couple of days...
    - Have a second back-up card
  - Having to change all my autopay items...
    - Grrrr....

But What About "Debit" Cards?

- Theoretically the fraud protection is the same...
- But two caveats...
  - It is easier to not pay your credit card company than to claw money back from your bank...
  - Until the situation is resolved:
    - Credit card? It is the credit card company's money that is missing
    - Debit card? It is ***your*** money that is missing
- Result is debit card fraud is more transient disruptions...

So Two Different Policies...

- Credit card: Hakunna Matata!
    - I use it without reservation, just with a spare in case something happens
    - Probably 2-3 compromise events have happened, and its annoying but ah well
        - The most interesting was $1 to Tsunami relief in 2004...
          was a way for the attacker to test that the stolen card was valid
- Debit card: Paranoia-city...
    - It is an ATM-ONLY card (no Visa/Mastercard logo!)
    - It is used ONLY in ATMs belonging to my bank
        - Reduce the risk of "skimmers": rogue ATMs that record cards and keystrokes