Viruses & Worms

CS 161: Computer Security

Prof. Vern Paxson

TAs: Devdatta Akhawe, Mobin Javed & Matthias Vallentin

http://inst.eecs.berkeley.edu/~cs161/

April 19, 2011

Announcements

- Matthias out for at least this coming week :-(
  - Note, his sections are still being held!
- HKN reviewing this Thursday, 12:15PM
- Project #2 out today, due 11:59PM Thu May 5
- Course Summary lecture?
  - Comprehensive overview of the material we've covered
  - For sure works best if you take advantage of the opportunity to ask questions …
    - … including sending them in advance

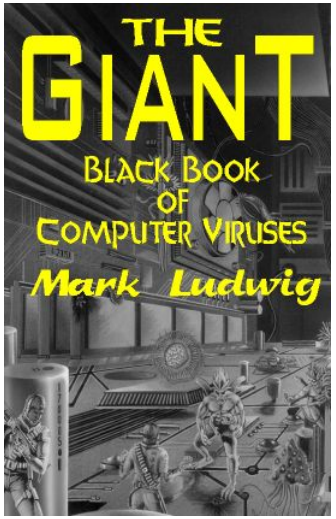
Malware That Propagates

- Virus = code that propagates (replicates) across systems by arranging to have itself eventually executed
  - Generally infects by altering stored code

- 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

The Problem of Viruses

- Virus = code that replicates
  - Instances opportunistically create new addl. instances
  - Goal of replication: install code on additional systems
- Opportunistic = code will eventually execute
  - Generally due to user action
    - Running an app, booting their system, opening an attachment
- Separate notions for a virus: 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

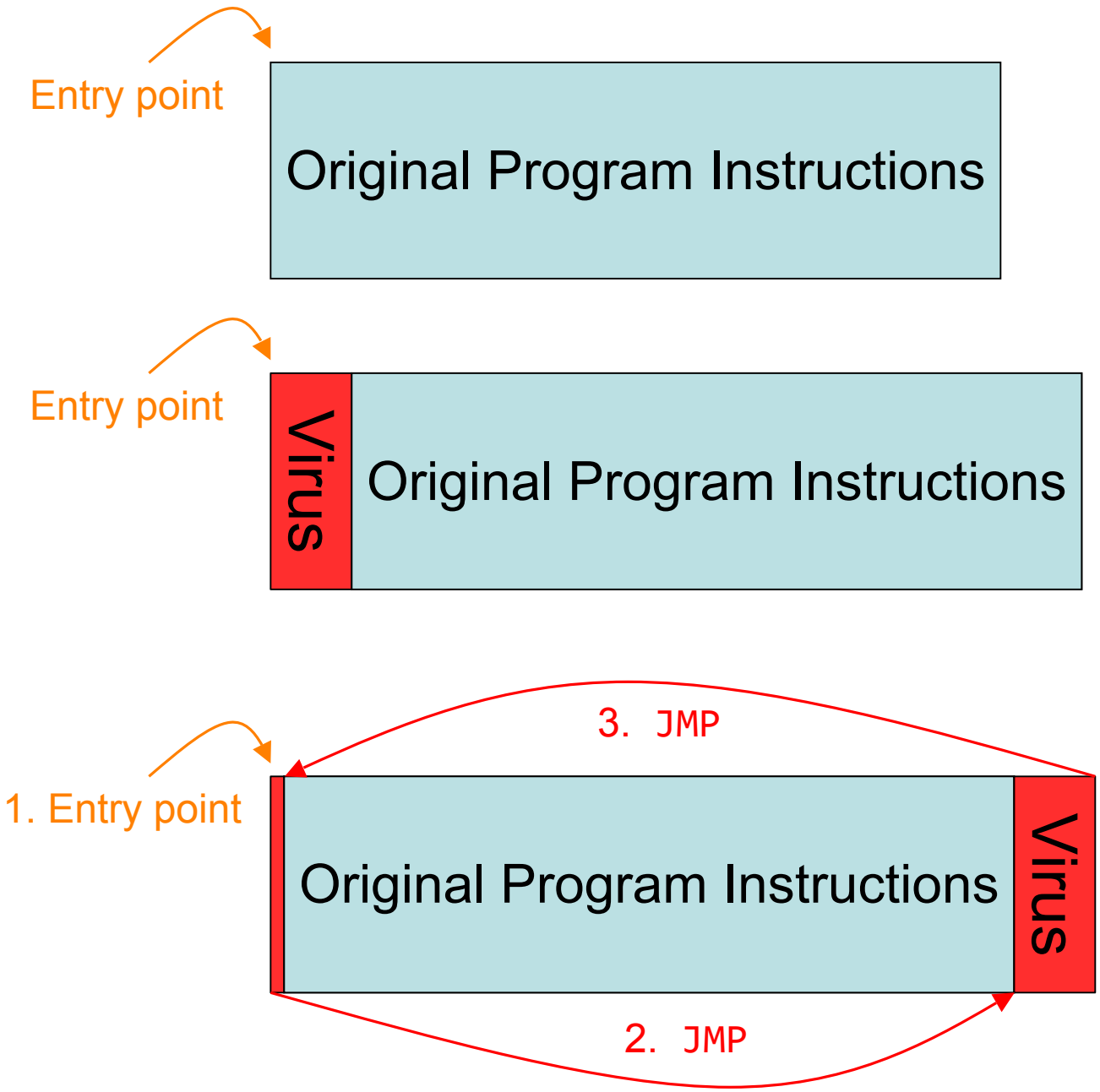


Propagation

- 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: if 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
- …

Many variants are possible, and of course can combine techniques

Payload

- Besides propagating, what else can the virus do when executing?
  - Pretty much *anything*
    - Payload is decoupled from propagation
    - Only subject to permissions under which it runs
- Examples:
  - Brag or exhort (pop up a message)
  - Trash files (just to be nasty)
  - Damage hardware (!)
  - Keylogging
  - Encrypt files
    - "Ransomware"
- Possibly delayed until condition occurs
  - "time bomb" / "logic bomb"

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 **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. More information...

1 VT Community user(s) with a total of 1 reputation credit(s) say(s) this sample is goodware. 6 VT Community user(s) with a total of 8 reputation credit(s) say(s) this sample is malware.

File name: **4.doc**
Submission date: **2011-04-19 07:19:30 (UTC)**
Current status: **finished**
Result: **27 /42 (64.3%)**

VT Community

malware
Safety score: 11.1%

Compact

Print results

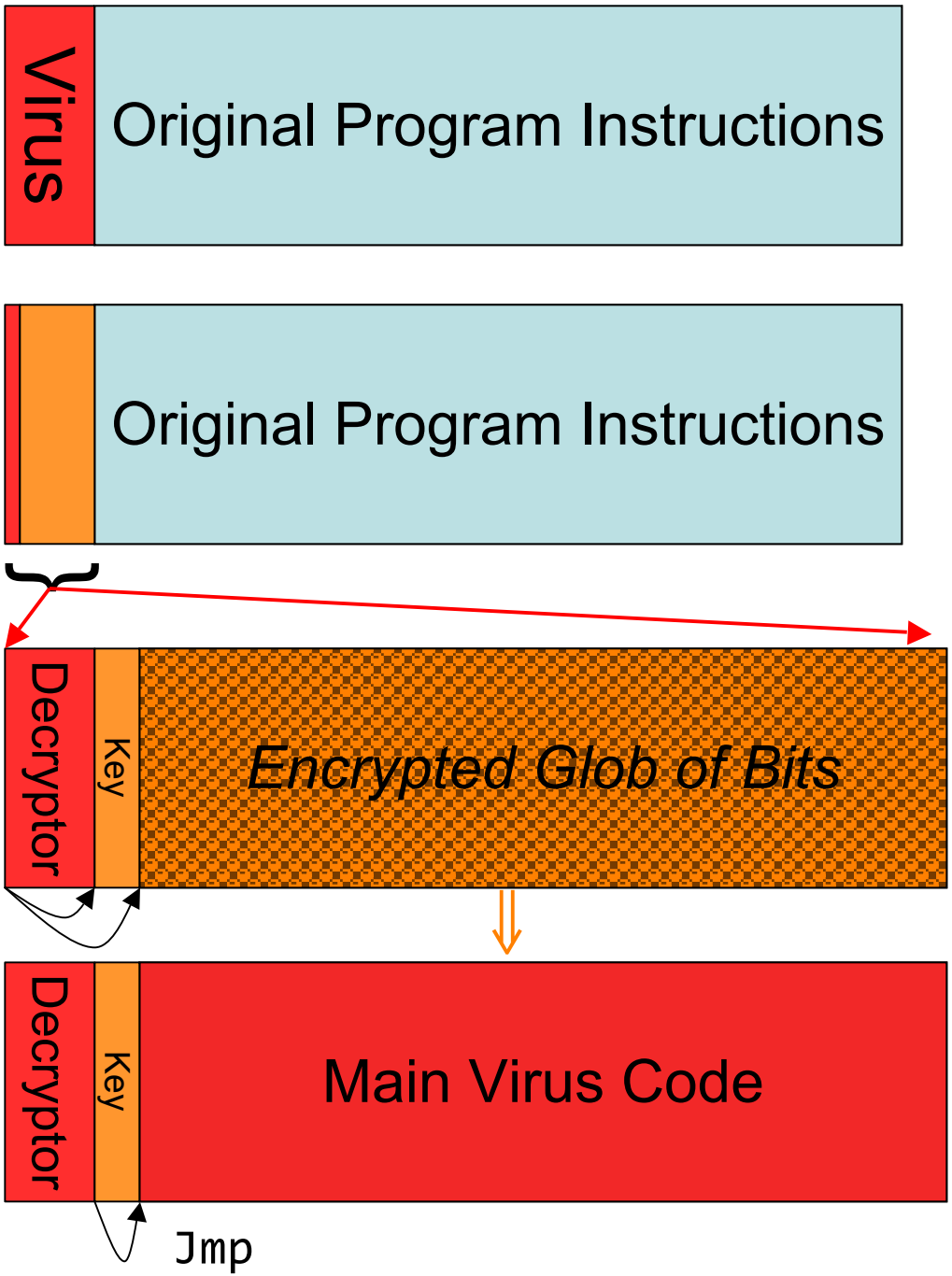
Antivirus	Version	Last Update	Result
AhnLab-V3	2011.04.19.01	2011.04.19	Dropper/Cve-2011-0611
AntiVir	7.11.6.177	2011.04.19	EXP/CVE-2011-0611
Antiy-AVL	2.0.3.7	2011.04.18	Exploit/SWF.CVE-2011-0611
Avast	4.8.1351.0	2011.04.18	SWF:CVE-2011-0609-C
Avast5	5.0.677.0	2011.04.18	SWF:CVE-2011-0609-C
AVG	10.0.0.1190	2011.04.18	-
BitDefender	7.2	2011.04.19	-
CAT-QuickHeal	11.00	2011.04.19	-
ClamAV	0.97.0.0	2011.04.19	-
Commtouch	5.3.2.6	2011.04.19	MSWord/Dropper.B!Camelot
Comodo	8396	2011.04.19	UnclassifiedMalware
DrWeb	5.0.2.03300	2011.04.19	Exploit.Wordbo.12
Emsisoft	5.1.0.5	2011.04.19	Exploit.SWF.CVE-2011-0611!IK

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!
- Writing new viruses by hand takes a lot of effort
- How can you mechanize the creation of new instances of your viruses …
  - … such 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 some data that appears completely unrelated to the original data: 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



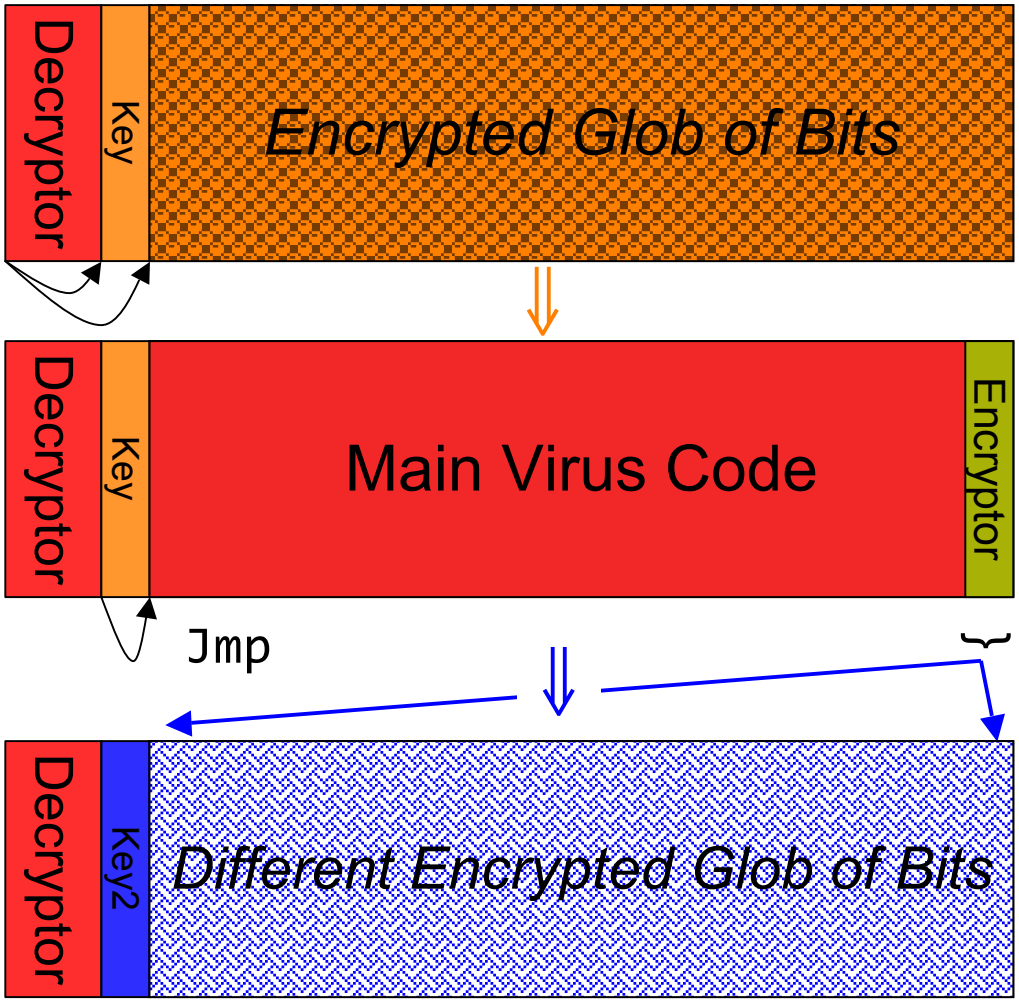
Instead of this …

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

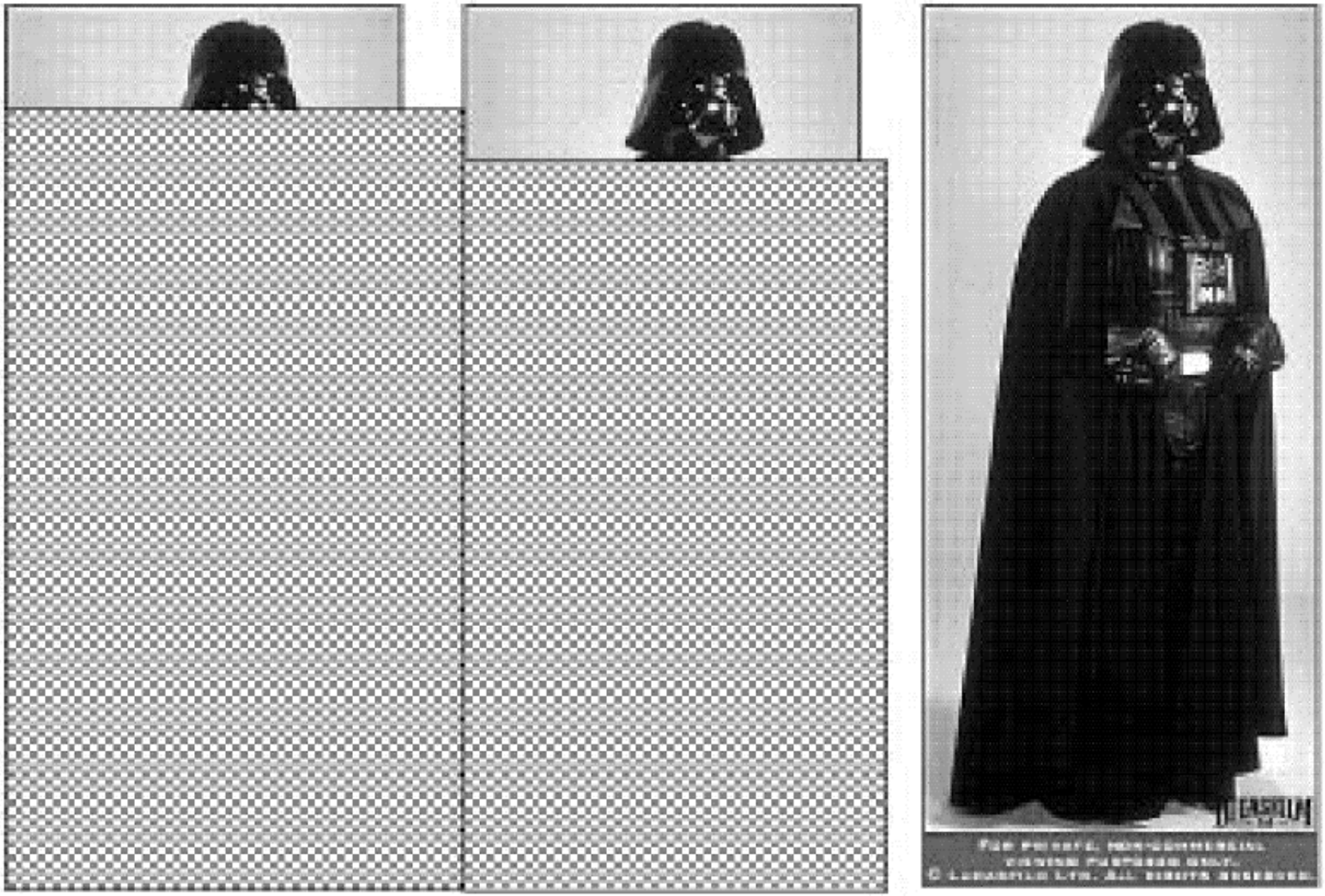
Arms Race: Polymorphic Code

- 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
      - 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

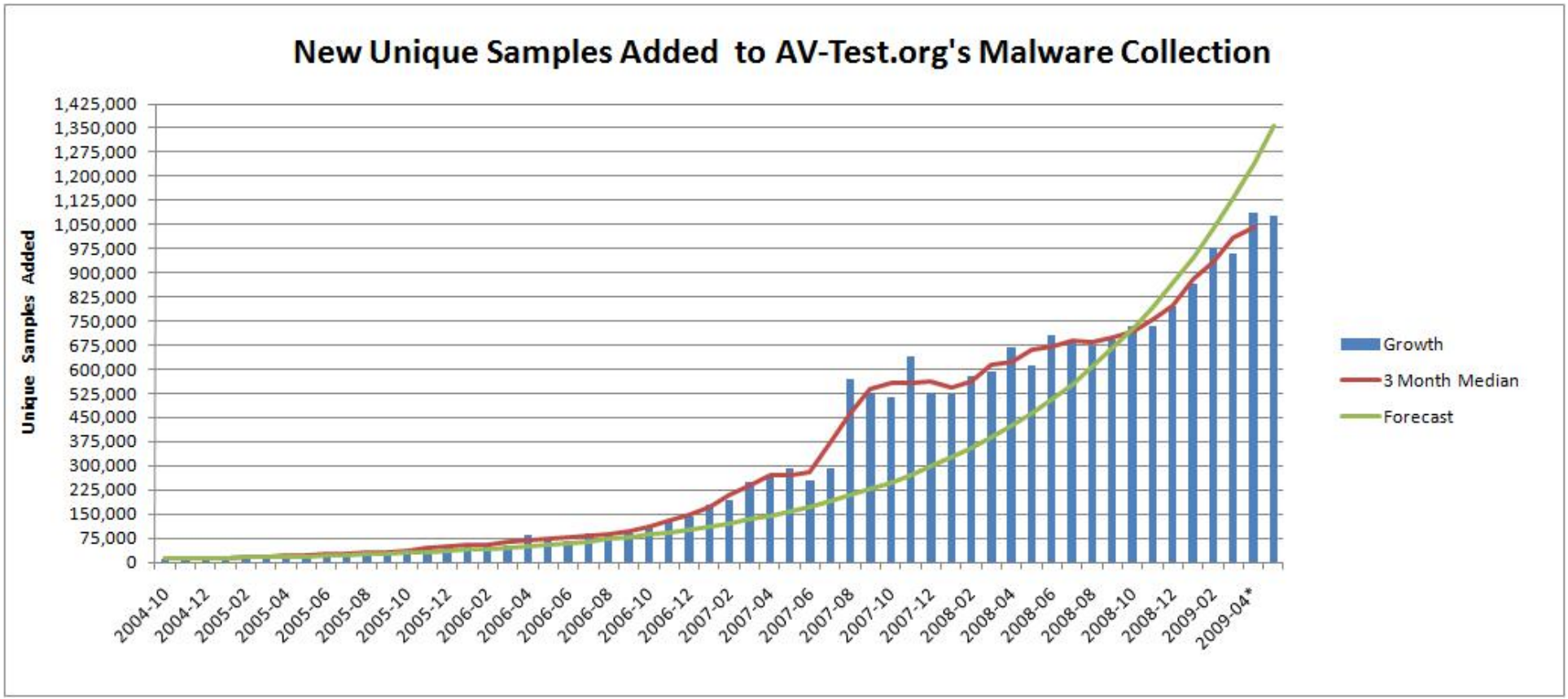
Hunting for Metamorphic, Szor & Ferrie, Symantec Corp., Virus Bulletin Conference, 2001

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 behaviorial signature, (2) AV software on end system analyzes 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*

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 1000s 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*)


New Unique Samples Added to AV-Test.org's Malware Collection
Unique Samples Added
1,425,000
1,350,000
1,275,000
1,200,000
1,125,000
1,050,000
975,000
900,000
825,000
750,000
675,000
600,000
525,000
450,000
375,000
300,000
225,000
150,000
75,000
0
2004-10
2004-12
2005-02
2005-04
2005-06
2005-08
2005-10
2005-12
2006-02
2006-04
2006-06
2006-08
2006-10
2006-12
2007-02
2007-04
2007-06
2007-08
2007-10
2007-12
2008-02
2008-04
2008-06
2008-08
2008-10
2008-12
2009-02
2009-04*
Growth
3 Month Median
Forecast

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

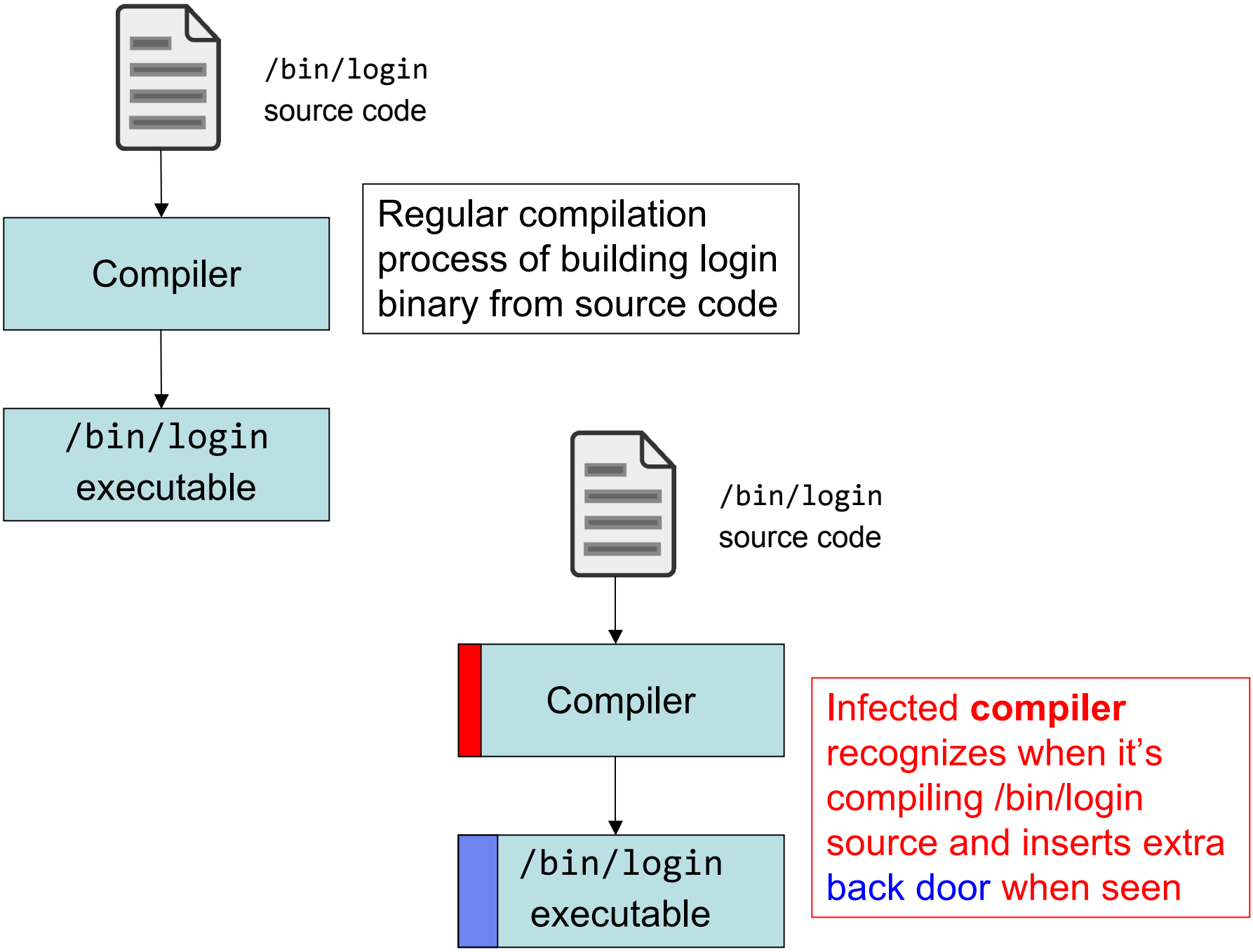
  - i.e., rebuild system from original media + data backups
- If we have complete source code for system, we could rebuild from that instead, right?

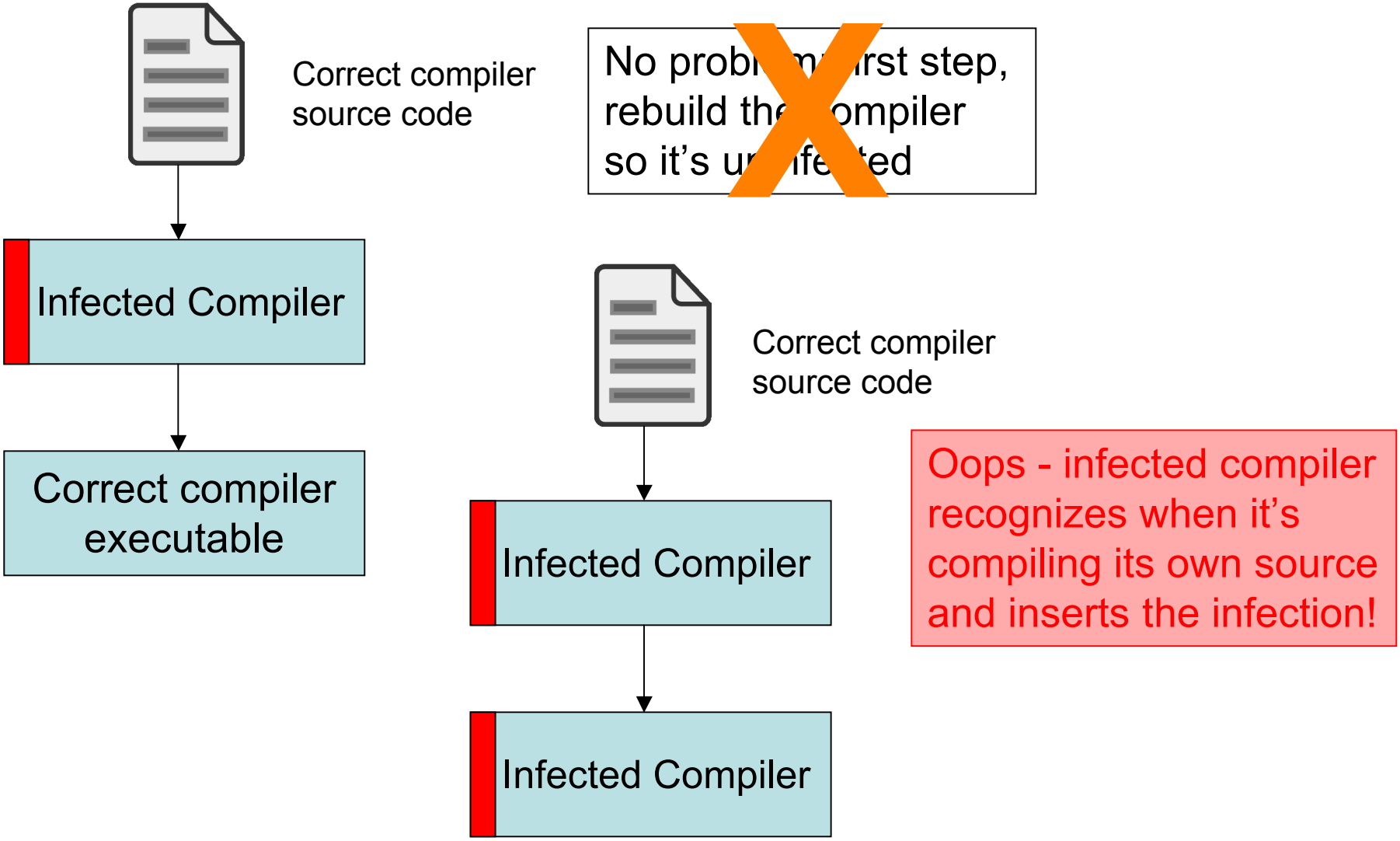
The Perils of Rebuilding From Source

- If we have complete source code for system, we could rebuild from that instead, right?

- 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 …


/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

Worms

The Problem of Worms

- Virus = code that propagates (replicates) across systems by arranging to be eventually executed
  - Generally infects by altering *stored code*
- Worm = code that self-propagates/replicates across systems by arranging to have itself immediately executed
  - Generally infects by altering or initiating *running code*
  - No user intervention required
- Like with viruses, for worms we can separate out propagation from payload
- Propagation includes notions of *targeting* & *exploit*
  - How does the worm **find** new prospective victims?
  - How does worm get code to **automatically run**?

Studying Worms

- *Internet-scale events*
  - Surprising dynamics / emergent behavior
  - Hard problem of attribution (who launched it)
- Modeling propagation mathematically
- Evolution / ecosystem
  - Shifting perspectives on nature of problem
  - *Remanence*
- "Better" worms
- Thinking about defenses
  - Including "white worms"
- Mostly illustrated from a historical perspective …
  - Details/dates/names for the most part not important
    - Other than **Morris Worm**, **Code Red**, and **Slammer**

The Arrival of Internet Worms

- Internet worms date to Nov 2, 1988 - the *Morris Worm*
  - ***Way*** ahead of its time
- Modern Era begins 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 #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

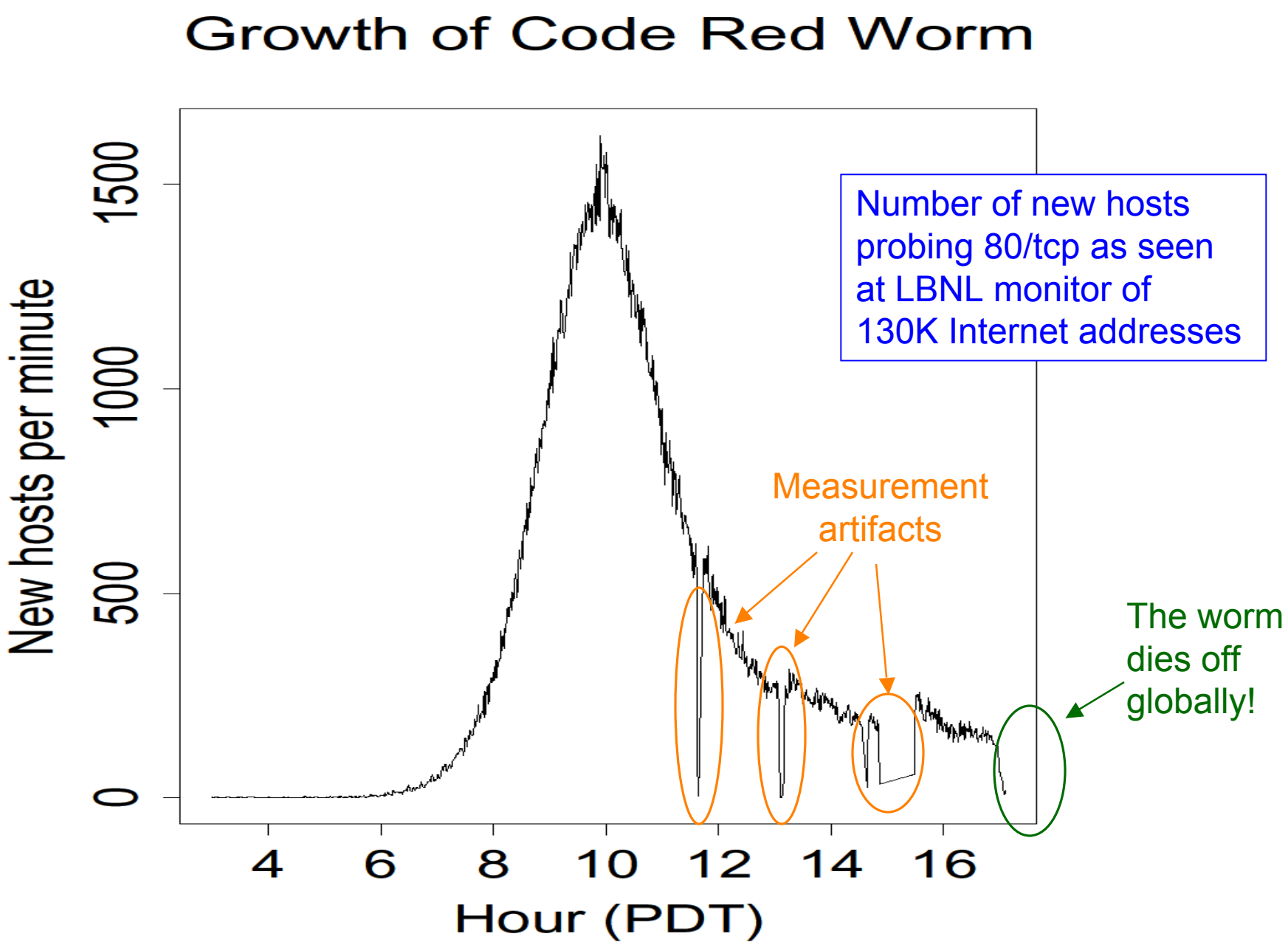
- Payload #2: check day-of-the-month and …
  - … 1st through 20th of each month: spread
  - … 20th through end of each month: attack
    - Flooding attack against 198.137.240.91 …
    - … i.e., *www.whitehouse.gov*
- Spread: via *random scanning* of 32-bit IP address space
  - Generate pseudo-random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