

Slides from

- Michelle Mazurek 414-fall2016
- Dave Levin 414-spring2016

# Malware

With material from Dave Levin,  
Vern Paxson, Dawn Song

**Malware: Malicious code that runs on the victim's system**

# How does malware enter and run?

- Attacks a user- or network-facing **vulnerable service**
  - e.g., using techniques from prior lectures
- **Backdoor**: Added by a malicious developer
- **Social engineering**: Trick user into running/clicking
- **Trojan horse**: Offer a good service, add in the bad
- Attacker with physical access installs & runs it

# What does malware do?

- **Payload**: practically anything (subject to permissions)
  - Brag: “APRIL 1st HA HA HA HA YOU HAVE A VIRUS!”
  - Destroy: files, hardware
  - Crash the machine: over-loading, fork bomb, ...
  - Steal information (“exfiltrate”)
  - Launch external attacks: spam, click fraud, DoS
  - **Ransomware**: e.g., by encrypting files
  - **Rootkits**: Hide from user or software-based detection
    - Often by modifying the kernel
    - **Man-in-the-middle attacks** to sit between UI and reality
- **Propagate**: stealthily or quickly & widely

# Viruses vs Worms

- **Virus**: Run when user initiates something
  - Run program, open attachment, boot machine
  - Typically infects *stored* code
  - Self-propagating: Create new instance elsewhere
  - Goal: **avoid detection, wait for opportunity**
  - Around since at least 70s
- **Worm**: Runs while another program is running
  - No user intervention required
  - Typically infects *running* code
  - Self-propagating: infect running code elsewhere
  - Mode: **spread widely & quickly; outpace** response

**The line between these is thin and blurry; some are both**

# Viruses

# Classified by what they infect

- Document viruses
  - Implemented within a formatted document (Word, PDF, etc.)
  - Enabled by macros, javascript
  - (Why you shouldn't open random attachments)
- Boot sector viruses
  - Boot sector: small disk partition at fixed location
  - If booting from disk, firmware loads it into memory and runs it:
    - Original code loads the OS
    - Virus does other things, *then* load OS
  - Similar: AutoRun on music/video disks
  - (Why you shouldn't plug random USB drives into your computer)
- Etc.

# Viruses have resulted in a technological arms race

The key is *evasion*



Mechanisms for  
evasive  
**propagation**

Mechanisms for  
**detection** and  
prevention

Want to be able to  
claim wide coverage  
for a long time

Want to be able to  
claim the ability to  
detect *many* viruses



# Detecting viruses: Signatures

- Identify bytes corresponding to known virus
- Install **recognizer** to check all files
  - In practice, requires fast scanning
- Drives multi-billion\$ antivirus market
  - Marketing via # signatures recognized
  - Is this a useful metric?

Products &amp; Solutions ▼

Support &amp; Communities ▼

Security Response ▼

Try &amp; Buy ▼

[↑](#) / [Security Response](#) / [Virus Definitions & Security Updates](#)

## Virus Definitions & Security Updates

To stay secure you should be running the most recent version of your licensed product and have the most up-to-date security content. Use this page to make sure your security content is current.

### Select product:

Symantec Endpoint Protection 12.1.3 ▼



Need to update your  
Norton products?

[Go to Norton.com](#)

A valid support contract is required to obtain the latest content. To renew your product license, see the [License Renewal Center](#).

### ■ File-Based Protection (Traditional Antivirus)

**Definitions Created:** 2/10/2014**Definitions Released:** 2/10/2014**Extended Version:** 2/10/2014 rev. 16**Definitions Version:** 160210p**Sequence Number:** 151231**Number of Signatures:** 23,927,535**Details:** [Release History](#)**Download:** [Definitions](#), Content is downloaded by your product via LiveUpdate.

Um.. thanks?

FEATURE

# Antivirus vendors go beyond signature-based antivirus

Robert Westervelt, News Director 



This article can also be found in the Premium Editorial Download **"Information Security magazine: Successful cloud migrations require careful planning."**

[Download it now](#) to read this article plus other related content.

Security experts and executives at security vendors are in agreement that signature-based antivirus isn't able to keep up with the explosion of malware. For example, in 2009, Symantec says it wrote about 15,000 antivirus signatures a day; that number has increased to 25,000 antivirus signatures every day.

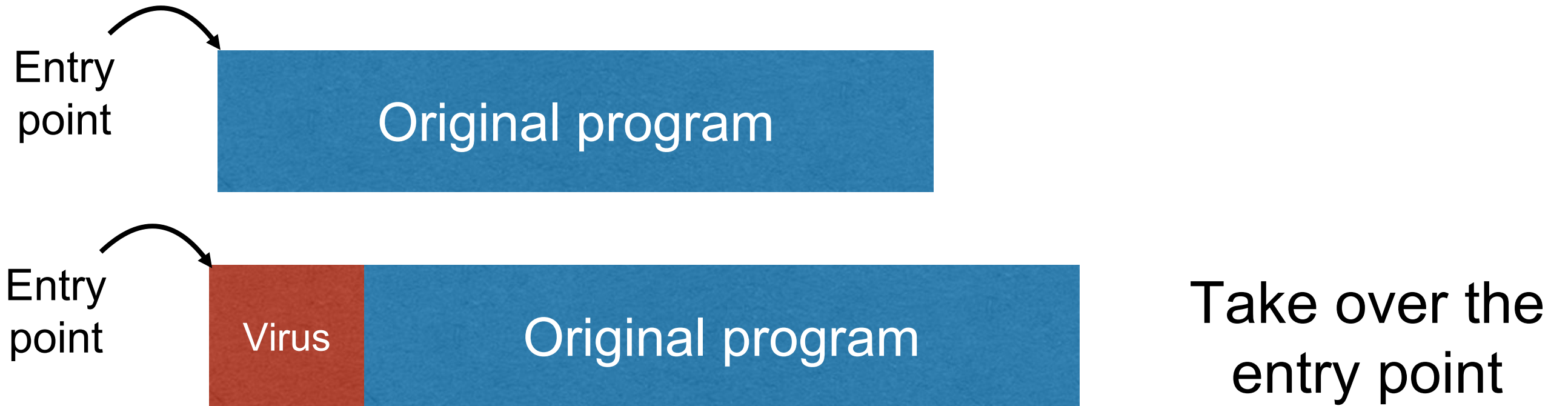
"Signatures have been dying for quite a while," says Mikko H. Hypponen, chief research officer of Finnish-based antivirus vendor, F-Secure. "The sheer number of malware samples we see every day completely overwhelms our ability to keep up with them."

Security vendors have responded by updating their products with additional capabilities, such as file reputation and heuristics-based engines. They're also making upgrades to keep up with the latest technology trends, such as virtualization and cloud computing.

# Virus writer response

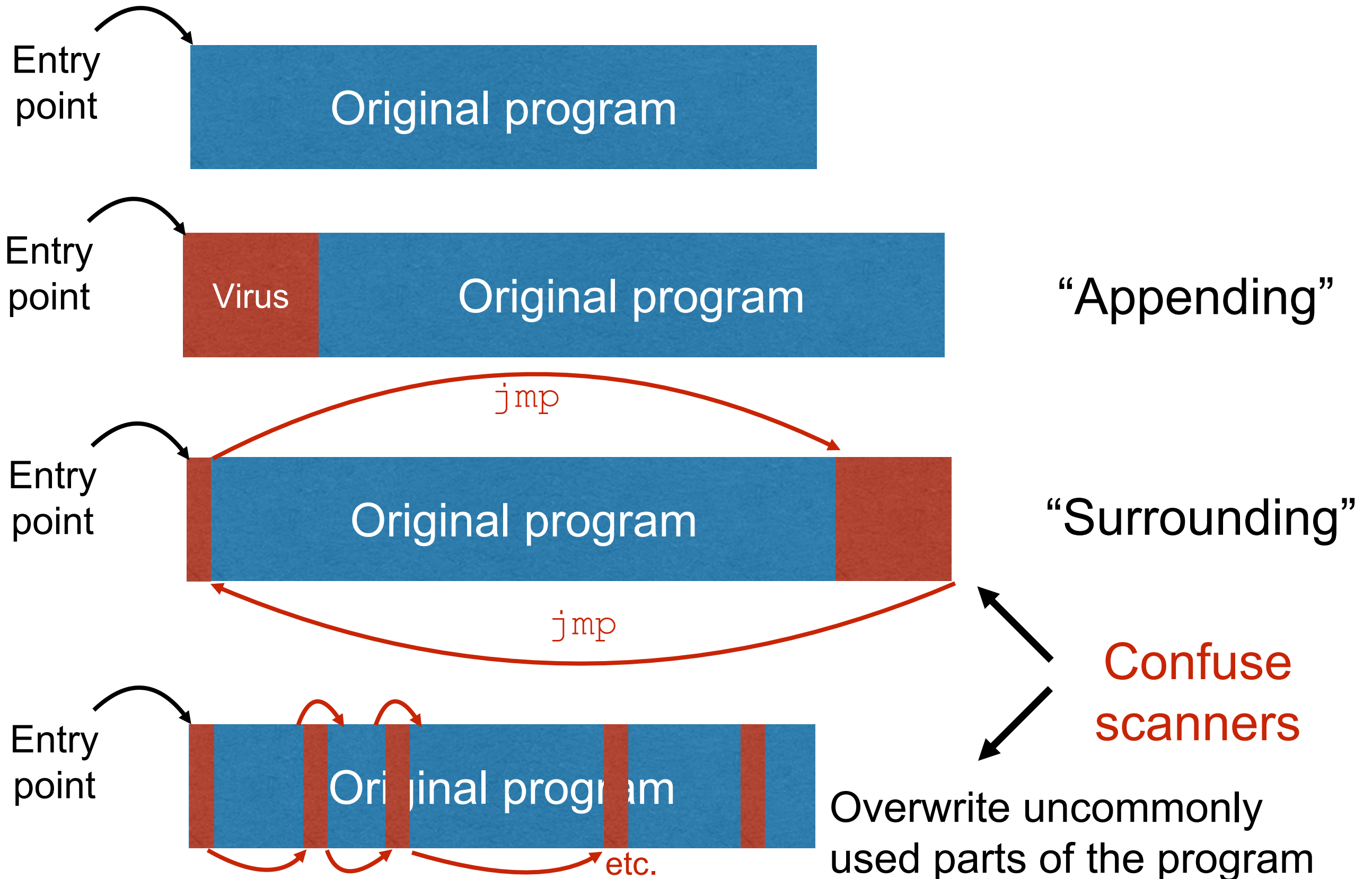
- Make signature **harder to find**
- **Disperse** the virus code in different places in target
- **Mutate** the virus code at each infection
  - polymorphism
  - metamorphism

# How viruses infect other programs



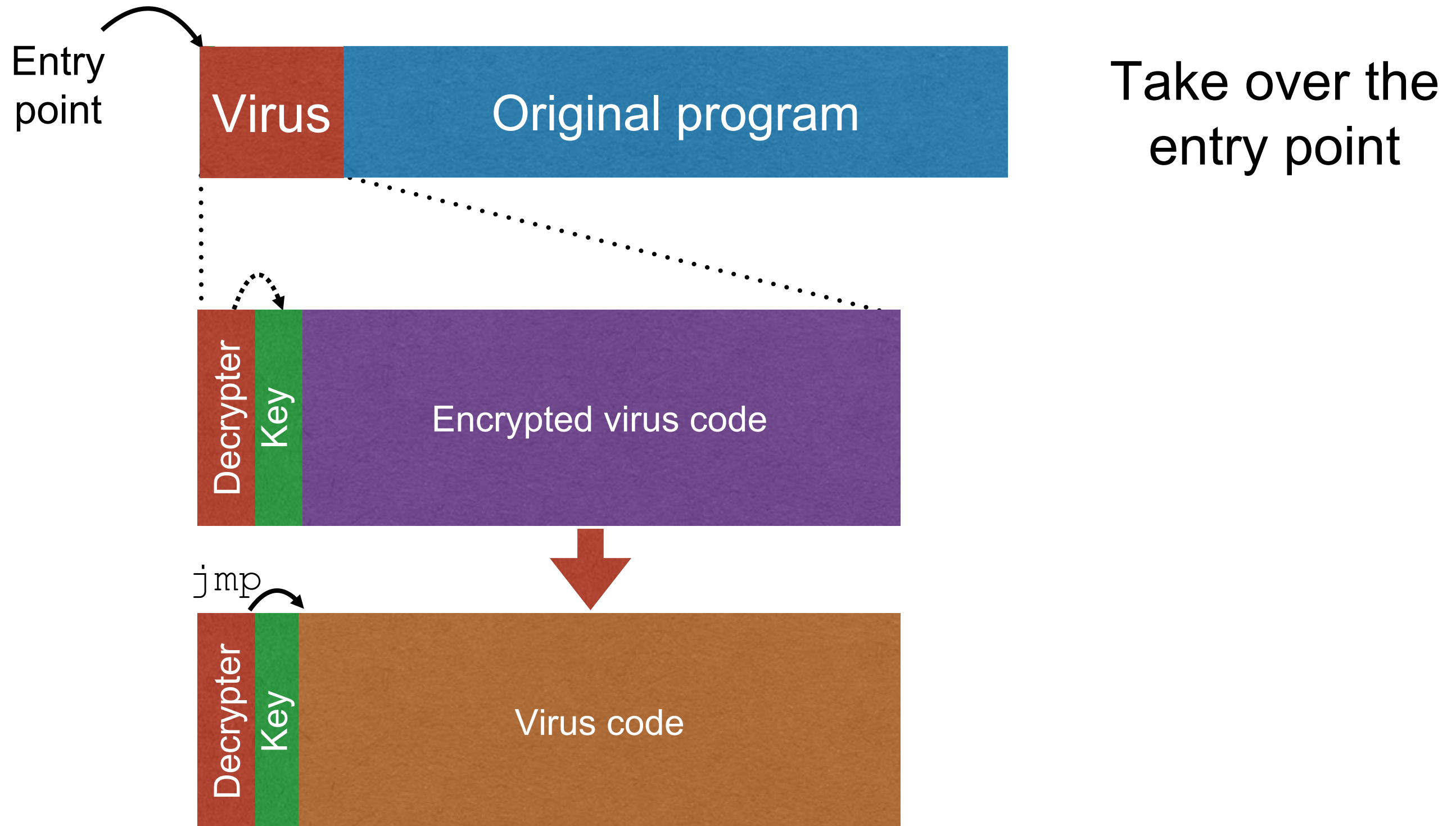


# How viruses infect other programs



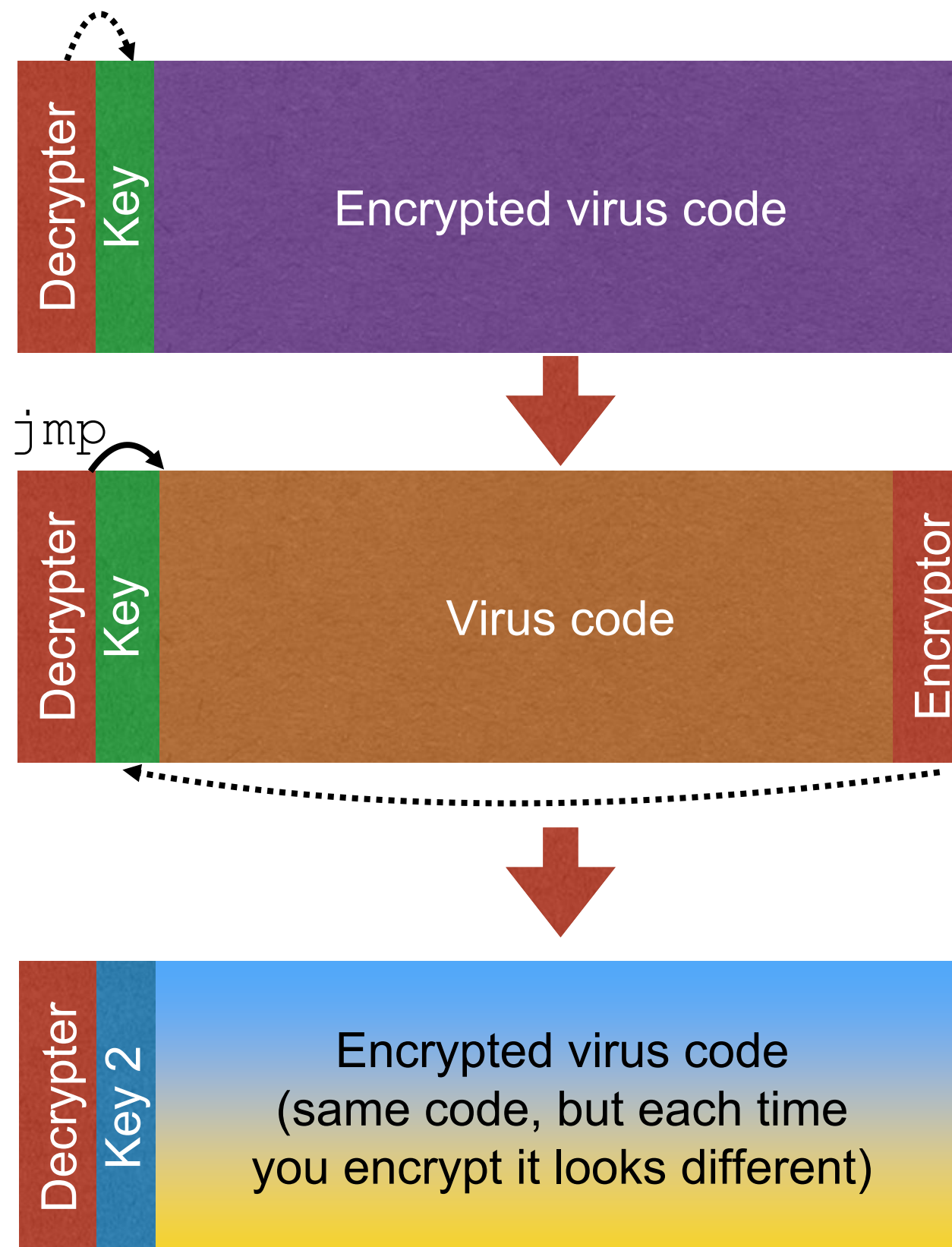
# Polymorphic and metamorphic viruses

# Polymorphic using encryption





# Making it automatic



**When used properly,  
encryption will yield  
a different, random  
output upon each  
invocation**

# Polymorphic viruses: Arms race

**Now you are the antivirus writer: how do you detect?**

- Idea #1: **Narrow signature** to catch the decrypter
  - Often very small: can result in many false positives
  - Attacker can spread this small code around and `jmp`
- Idea #2: **Execute** or statically analyze the suspect code to see if it decrypts.
  - How do you distinguish from common “packers” which do something similar (decompression)?
  - How long do you execute the code??

**Now you are the *virus* writer again: how do you evade?**

# Polymorphic countermeasures

- Change the decrypter
  - **Oligomorphic viruses**: assemble decrypter from several interchangeable alternative pieces
  - **True polymorphic viruses**: can generate an endless number of decrypters
    - Different encryption methods
    - Random generation of confounds
    - Downside: inefficient

# Metamorphic viruses

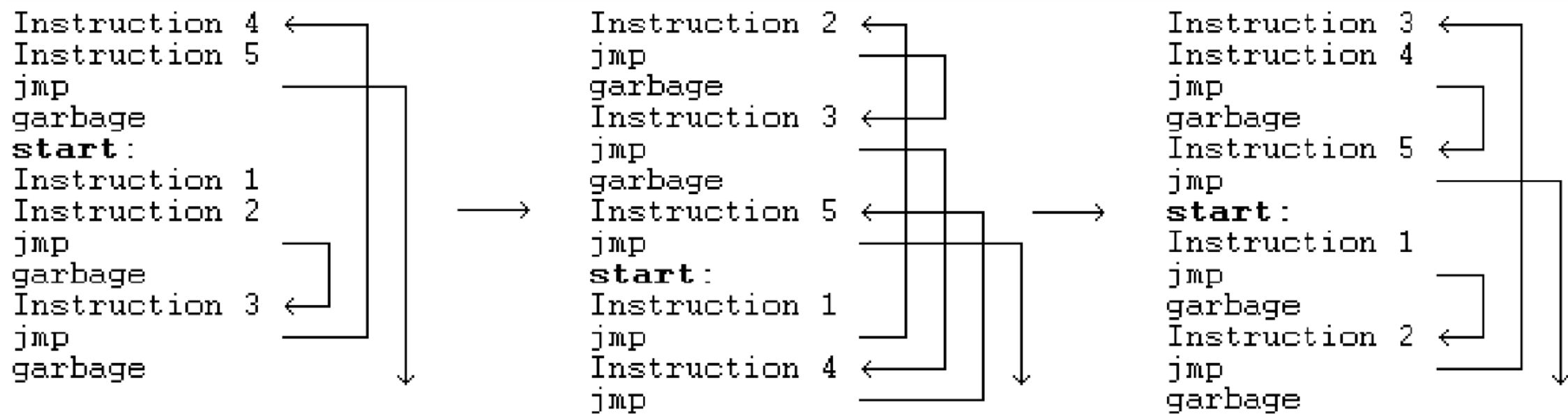
- Every time the virus propagates, generate a *semantically different* version of the code
  - Higher-level semantics remain the same
  - But the way it does it differs
    - Different machine code instructions
    - Different algorithms to achieve the same thing
    - Different use of registers
    - Different constants....
- How would you do this?
  - Include a code rewriter with your virus
  - Add a bunch of complex code to throw others off (then just never run it)

## Symantec HUNTING FOR METAMORPHIC

```
5A          pop     edx
BF04000000  mov     edi,0004h
8BF5        mov     esi,ebp
B80C000000  mov     eax,000Ch
81C288000000 add     edx,0088h
8B1A        mov     ebx,[edx]
899C8618110000 mov    [esi+eax*4+00001118],ebx

58          pop     eax
BB04000000  mov     ebx,0004h
8BD5        mov     edx,ebp
BF0C000000  mov     edi,000Ch
81C088000000 add     eax,0088h
8B30        mov     esi,[eax]
89B4BA18110000 mov    [edx+edi*4+00001118],esi
```

Figure 4: Win95/Regswap using different registers in new generations



ZPerm can directly reorder the instructions in its own code

Figure 7 Zperm.A inserts JMP instruction into its code

a. An early generation:

```
C7060F000055  mov     dword ptr [esi],5500000Fh
C746048BEC5151  mov     dword ptr [esi+0004],5151EC8Bh
```

b. And one of its later generations:

```
BF0F000055      mov     edi,5500000Fh
893E            mov     [esi],edi
5F              pop     edi
52              push    edx
B640            mov     dh,40
BA8BEC5151      mov     edx,5151EC8Bh
53              push    ebx
8BDA            mov     ebx,edx
895E04          mov     [esi+0004],ebx
```

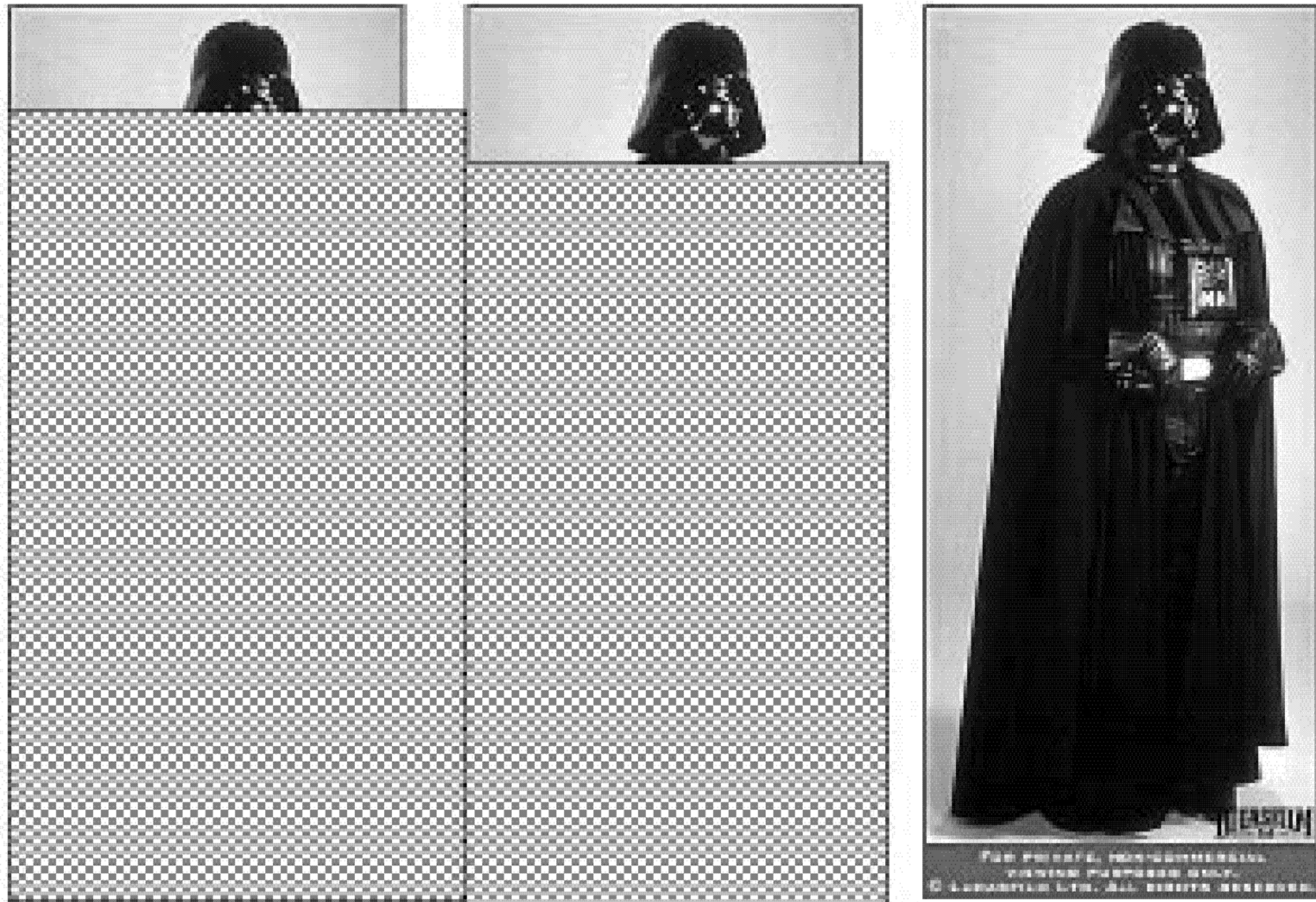
c. And yet another generation with recalculated ("encrypted") "constant" data.

```
BB0F000055      mov     ebx,5500000Fh
891E            mov     [esi],ebx
5B              pop     ebx
51              push    ecx
B9CB00C05F      mov     ecx,5FC000CBh
81C1C0EB91F1    add     ecx,F191EBC0h ; ecx=5151EC8Bh
894E04          mov     [esi+0004],ecx
```

Figure 6: Example of code metamorphosis of Win32/Evol



# Polymorphic



**When can AV software successfully scan?**

Figure 8: A partial or complete snapshot of polymorphic virus during execution cycle



# Metamorphic



**When can AV software successfully scan?**

Figure 10: T-1000 of Terminator 2

Detecting  
metamorphic viruses?

# Scanning isn't enough

- Need to analyze **execution behavior**
- Two broad stages in practice (both take place in a safe environment, like gdb or a virtual machine)
  1. AV company analyzes new virus to find **behavioral signature**
  2. AV system at end host analyzes suspect code to see if it matches the signature

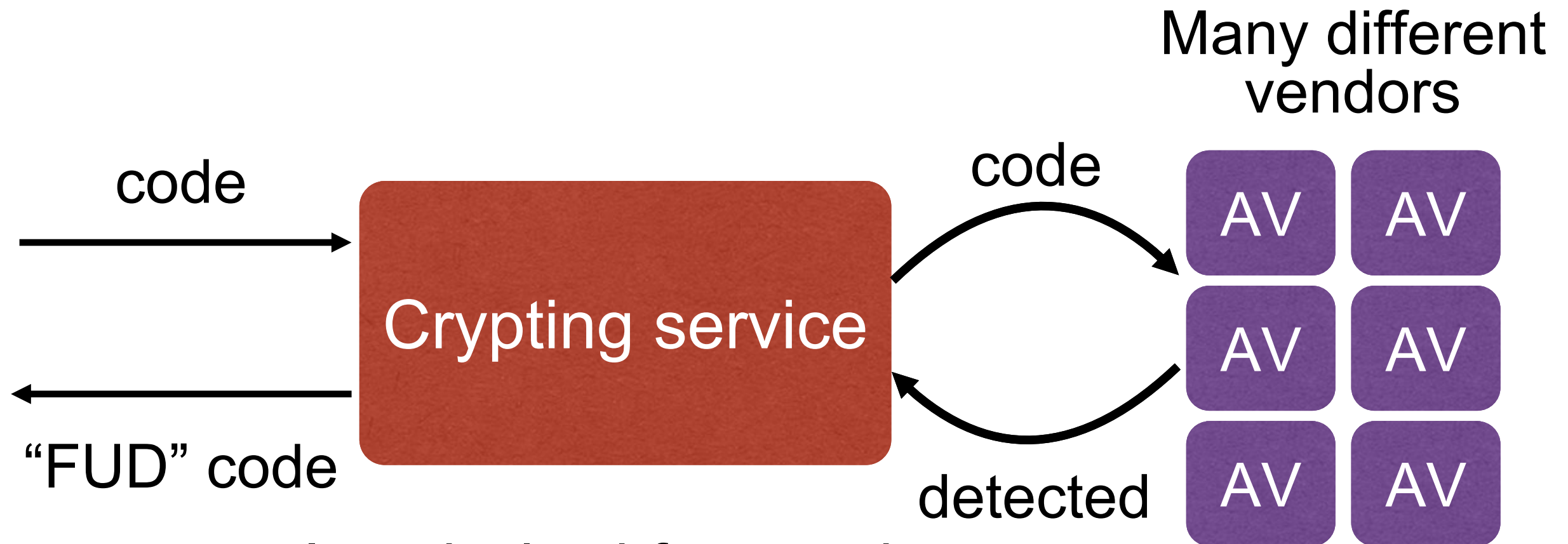
# Detecting metamorphic viruses

- Countermeasures
  - Change slowly (hard to observe pattern)
  - Detect if you are in a safe execution environment (e.g., gdb) and act differently
- Counter-countermeasures
  - **Detect detection** and skip those parts
- Counter-counter-counter.... Arms race

**Attackers have the upper hand:  
AV systems hand out signature *oracles***



# Crypting services



Iteratively obfuscate the  
code (encrypt + jmp + ...)

Until the obfuscated code  
is “fully undetectable”

# Putting it all together sounds hard

- **Creating** a virus can be really difficult
  - Historically error prone
- But **using** them is easy: any **scriptkiddy** can use metasploit
  - Good news: so can any white hat pen tester

```
root@bt: /opt/framework3/msf3
File Edit View Terminal Help
root@bt:/opt/framework3/msf3# msfcli windows/smb/ms08_067_netapi RHOST=192.168.1.100 P
[*] Please wait while we load the module tree...

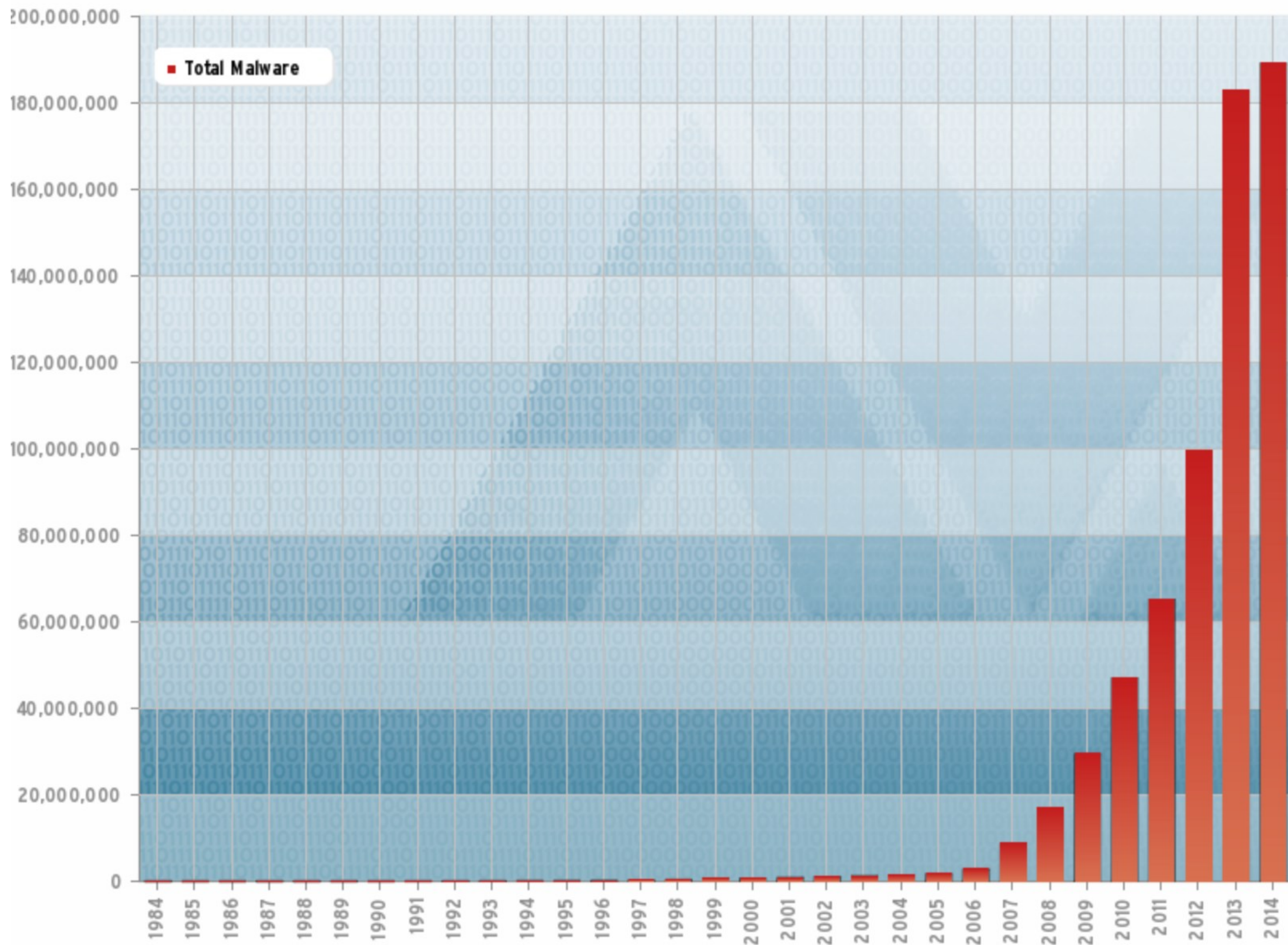
Compatible payloads
=====

Name                                     Description
----
generic/debug_trap                      Generate a debug trap in the target process
generic/shell_bind_tcp                  Listen for a connection and spawn a command shell
generic/shell_reverse_tcp               Connect back to attacker and spawn a command shell
generic/tight_loop                      Generate a tight loop in the target process
windows/adduser                         Create a new user and add them to local administration group
windows/dllinject/bind_ipv6_tcp         Listen for a connection over IPv6, Inject a Dll via a reflective loader
windows/dllinject/bind_nonx_tcp         Listen for a connection (No NX), Inject a Dll via a reflective loader
windows/dllinject/bind_tcp              Listen for a connection, Inject a Dll via a reflective loader
windows/dllinject/reverse_ipv6_tcp      Connect back to the attacker over IPv6, Inject a Dll via a reflective loader
windows/dllinject/reverse_nonx_tcp      Connect back to the attacker (No NX), Inject a Dll via a reflective loader
windows/dllinject/reverse_ord_tcp       Connect back to the attacker, Inject a Dll via a reflective loader
windows/dllinject/reverse_tcp           Connect back to the attacker, Inject a Dll via a reflective loader
windows/dllinject/reverse_tcp_allports  Try to connect back to the attacker, on all possible ports (1-65535, slowly), I
nject a Dll via a reflective loader
windows/dllinject/reverse_tcp_dns       Connect back to the attacker, Inject a Dll via a reflective loader
windows/download_exec                   Download an EXE from an HTTP URL and execute it
```

# So how much malware is out there?

- Polymorphic and metamorphic viruses can make it easy to *miscount* viruses
- Take numbers with a grain of salt
  - Large numbers are in the AV vendors' best interest
- Previously, most malware was showy
  - Now primary goal is frequently to not get noticed







# How do we clean up an infection?

- Depends what the virus did, but..
- May require restoring / repairing files
  - A service that antivirus companies sell
- What if the virus ran as root?
  - May need to rebuild the entire system
- So what, just recompile it?
  - What if the malware left a backdoor in your compiler?
    - Compile the malware back into the compiler
  - May need to use original media and data backups

# Virus case studies

# Brain

## First IBM PC virus (1987)

- Propagation
  - Copies itself into the boot sector
  - Tells the OS that all of the boot sector is “faulty” (so that it won’t list contents to the user)
    - Thus also one of the first examples of a **stealth** virus
  - Intercepts disk read requests for 5.25” floppy drives
    - Sees if the 5th and 6th bytes of the boot sector are 0x1234
    - If so, then it’s already infected, otherwise, infect it
- Payload
  - Nothing really; goal was just to spread (to show off?)
  - However, it served as the template for future viruses

PC Tools Deluxe M.22

Disk View/Edit Service

Path=A:

Absolute sector 0000000, System BOOT

Displacement	Hex codes
0000(0000)	FA E9 4A 01 34 12 00 07 14 00 01 00 00 00 00 20
0016(0010)	20 20 20 20 20 20 57 65 6C 63 6F 6D 65 20 74 6F
0032(0020)	20 74 68 65 20 44 75 6E 67 65 6F 6E 20 20 20 20
0048(0030)	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0064(0040)	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0080(0050)	20 20 63 23 20 31 33 38 36 20 42 61 73 65 74 20
0096(0060)	26 20 41 60 6A 61 64 20 20 70 76 74 29 20 4C 74
0112(0070)	64 2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0128(0080)	20 42 52 41 49 4E 20 43 4F 40 50 55 54 45 52 20
0144(0090)	53 45 52 56 49 43 45 53 2E 2E 37 33 30 20 4E 49
0160(00A0)	5A 41 40 20 42 4C 4F 43 48 20 41 4C 4C 41 40 41
0176(00B0)	20 49 51 42 41 4C 20 54 4F 57 4E 20 20 20 20 20
0192(00C0)	20 20 20 20 20 20 20 20 20 20 20 4C 41 48 4F 52
0208(00D0)	45 20 50 41 48 49 53 54 41 4E 2E 2E 50 48 4F 4E
0224(00E0)	45 20 3A 34 33 30 37 39 31 2C 34 34 33 32 34 30
0240(00F0)	2C 32 38 30 35 33 30 2E 20 20 20 20 20 20 20 20

ASCII value  
-83044 of 0  
Welcome to  
the Dungeon

(c) 1988 Basit  
& Anjad (pvt) Lt  
d.

BRAIN COMPUTER  
SERVICES..730 MI  
ZAM BLOCC ALLAMA  
IQBAL TOWN

LAHORE  
E-PAKISTAN..PHON  
E :438791,443249  
,288530.

Home=begin of file/disk End=end of file/disk

ESC=Exit PgDn=forward PgUp=back F2=chg sector num F3=edit F4=get name

# Rootkits

- Recall: a rootkit is malicious code that takes steps to go undiscovered
  - By intercepting system calls, patching the kernel, etc.
  - Often effectively done by a man in the middle attack
- **Rootkit revealer**: analyzes the disk offline and through the online system calls, and compares
- Mark Russinovich ran a rootkit revealer and found a rootkit in 2005... **installed by a CD he had bought.**

# Sony XCP rootkit

**Detected 2005**

- Goal: keep users from copying copyrighted material
- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - Intercepted read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)
- How it messed up
  - Morally: violated trust
  - Technically: Hid **all files** that started with "\$sys\$"
  - Seriously?: Uninstaller did not actually uninstall; introduced additional vulnerability instead

# Worms

# Controlling millions of hosts: *Why?*

- Distributed Denial of Service (DDoS)
  - Generate network traffic from many sources ...
  - ... to a single destination to overload the target network
  - Consume enough resources to stop legitimate users
- Steal sensitive information from millions of others
  - Even a small fraction of unprotected people  $\Rightarrow$  \$
- Botnets
  - maintain command-control
  - sell hosts as resources
  - send spam, mine bitcoin, turn on webcams, install keyloggers

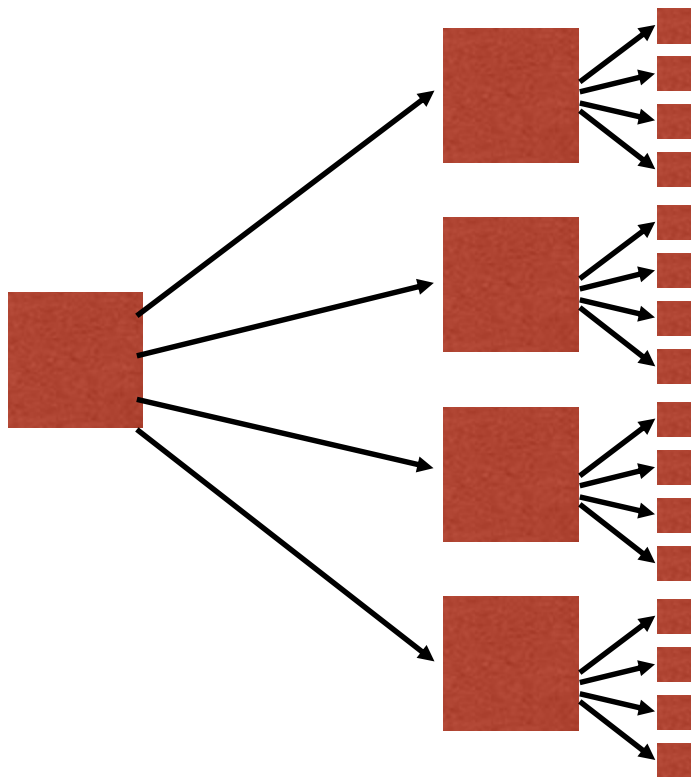


# Controlling millions of hosts: *How?*

- **Worm**: *self*-propagates by arranging to have itself *immediately* executed
  - At which point it creates a new, additional instance of itself
- Typically infects by altering *running* code
  - No user intervention required
- Like viruses, propagation and payload are orthogonal

# Self-propagation

- Goal: spread as quickly as possible
- The key is *parallelization*
  - Without being triggered by human interaction!



## Propagation

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

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

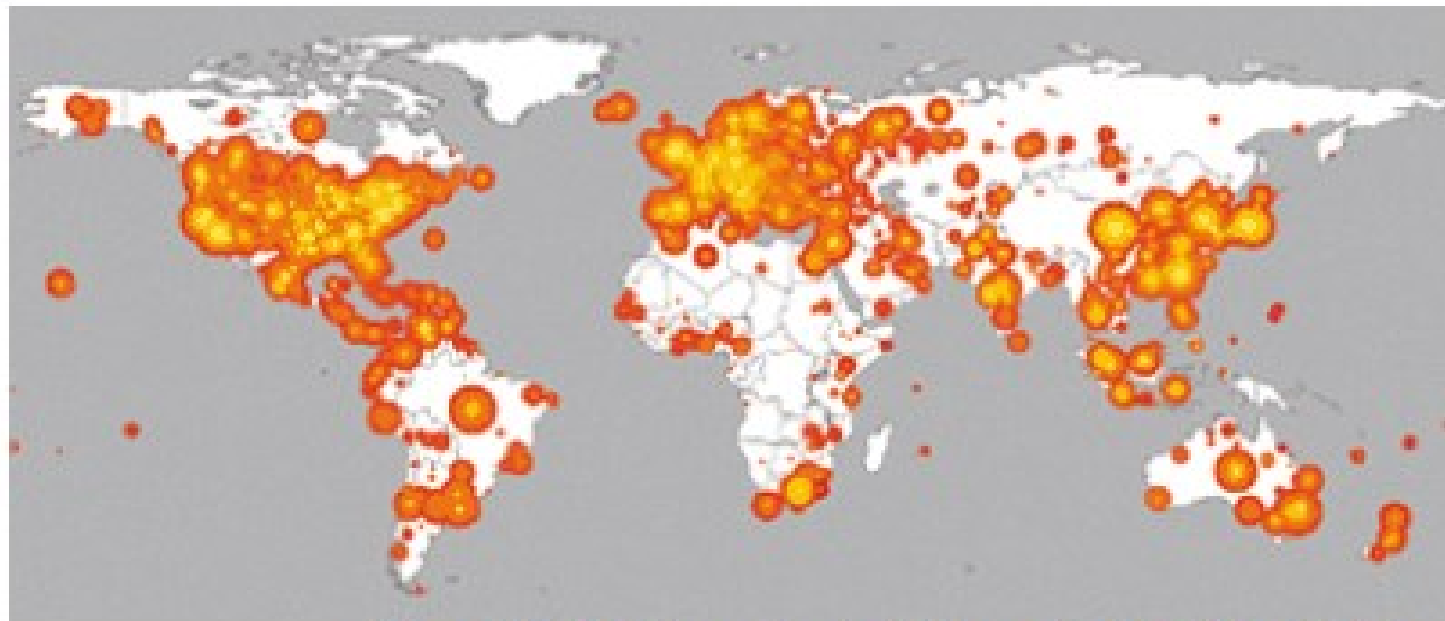
# Morris worm — 1988

- Variety of attacks
  - Buffer overflow attack against fingerd on VAXes
  - Crack passwords
  - ...
- More aggressive than intended
  - 6-10% of all internet hosts infected
- Didn't check OS: caused Suns running BSD to crash
- End result: \$10-100M damages, probation, comm. svc.

**Robert Morris is now a professor at MIT**

# Code Red — 2001

- Propagation: Exploited an overflow in MS-IIS server
  - 300,000 machines infected in 14 hours
  - At peak, more than 2000 new infections/minute
- Payload 1: website defacement
  - “HELLO! Welcome to <http://www.worm.com>! Hacked By Chinese!”
- Payload 2: time bomb
  - Day of month 1-20: **Spread**
  - Day of month 20+: **Attack** (flood 198.137.240.91 = whitehouse.gov)



# CodeRed Propagation

- Spread by randomly scanning the entire 32-bit IP address space
  - Pick a pseudorandom 32-bit number = IP addr
  - Send exploit packet to that address
  - Repeat
- This is a very common worm technique
- Each instance used the ***same random seed***
  - What does this mean in practice?

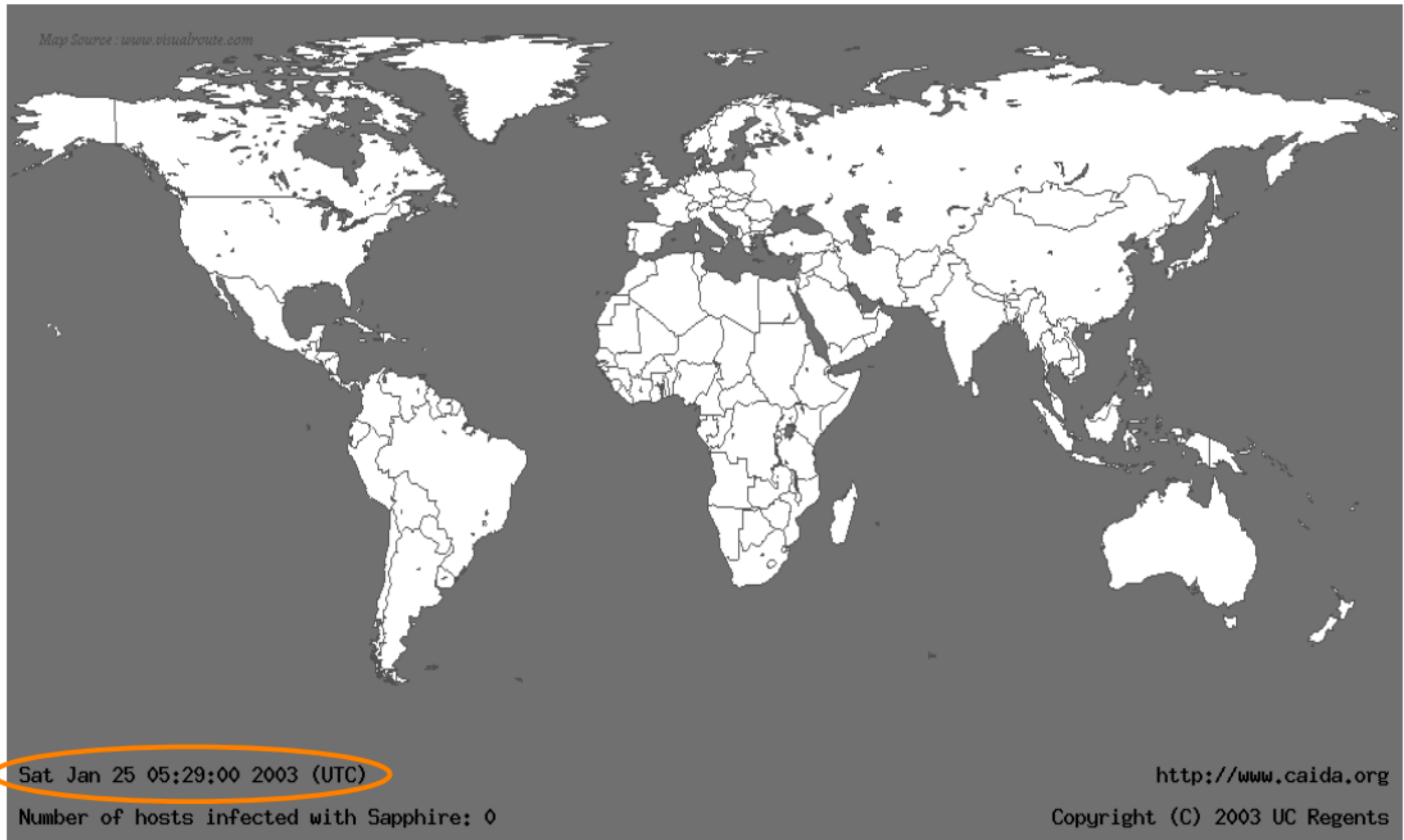
# More CodeRed

- If found c:\notworm then do nothing
- Whitehouse.gov **changed** its IP address
  - Made the attack portion useless
- Revision one week later: random number generator was **seeded** properly
  - No attack function, installs backdoor instead
  - By then many but not all hosts patched

# SQL Slammer (2003)

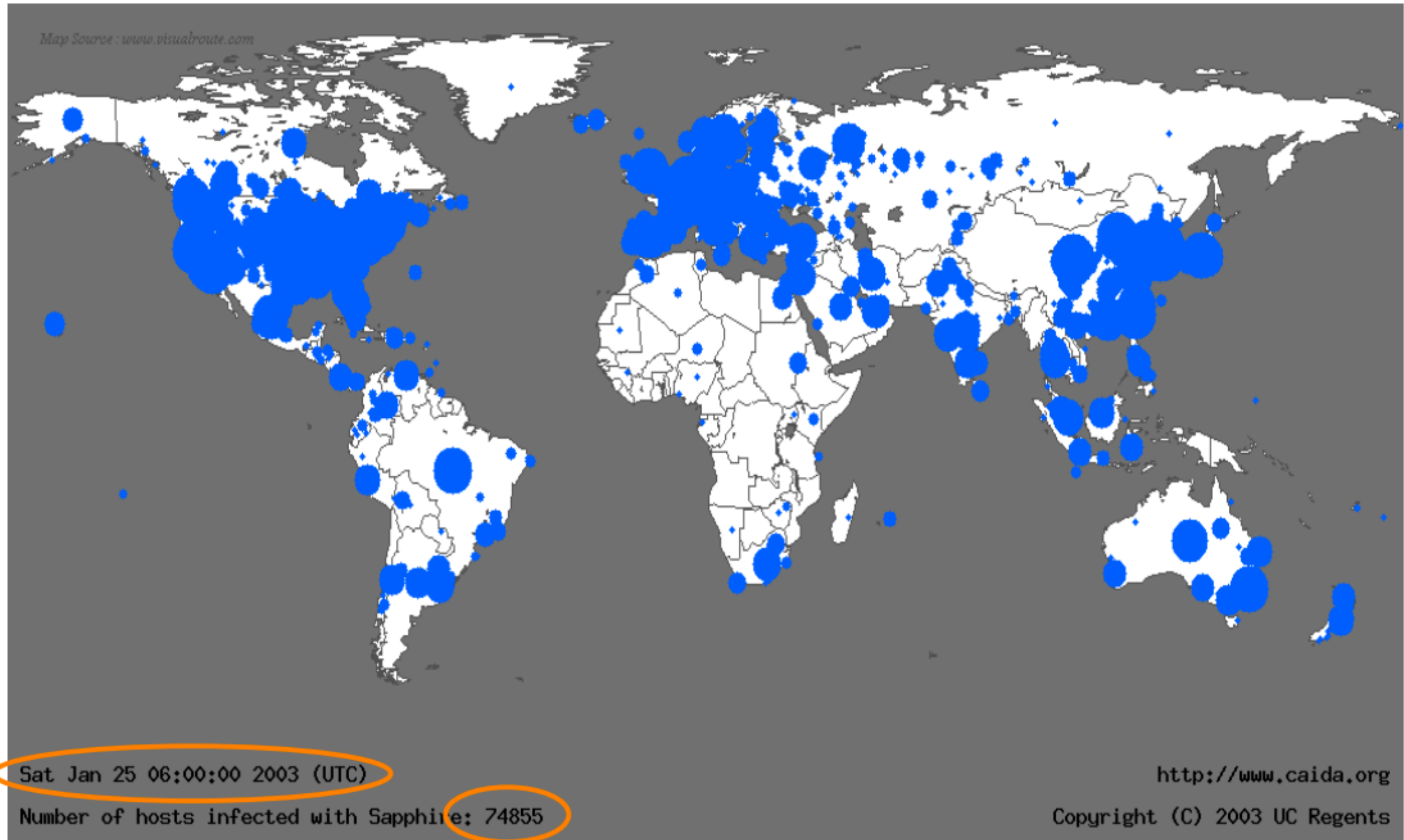
- Exploited overflow in MS SQL Server
  - Patch had been available for > 6 months
- Connectionless UDP rather than TCP
  - Entire worm fit in a single packet!
- When scanning, the worm could “fire and forget”
  - Stateless!
- Infected 75k machines in 10 minutes
  - At its peak, doubled every 8.5 seconds

# Life just before Slammer



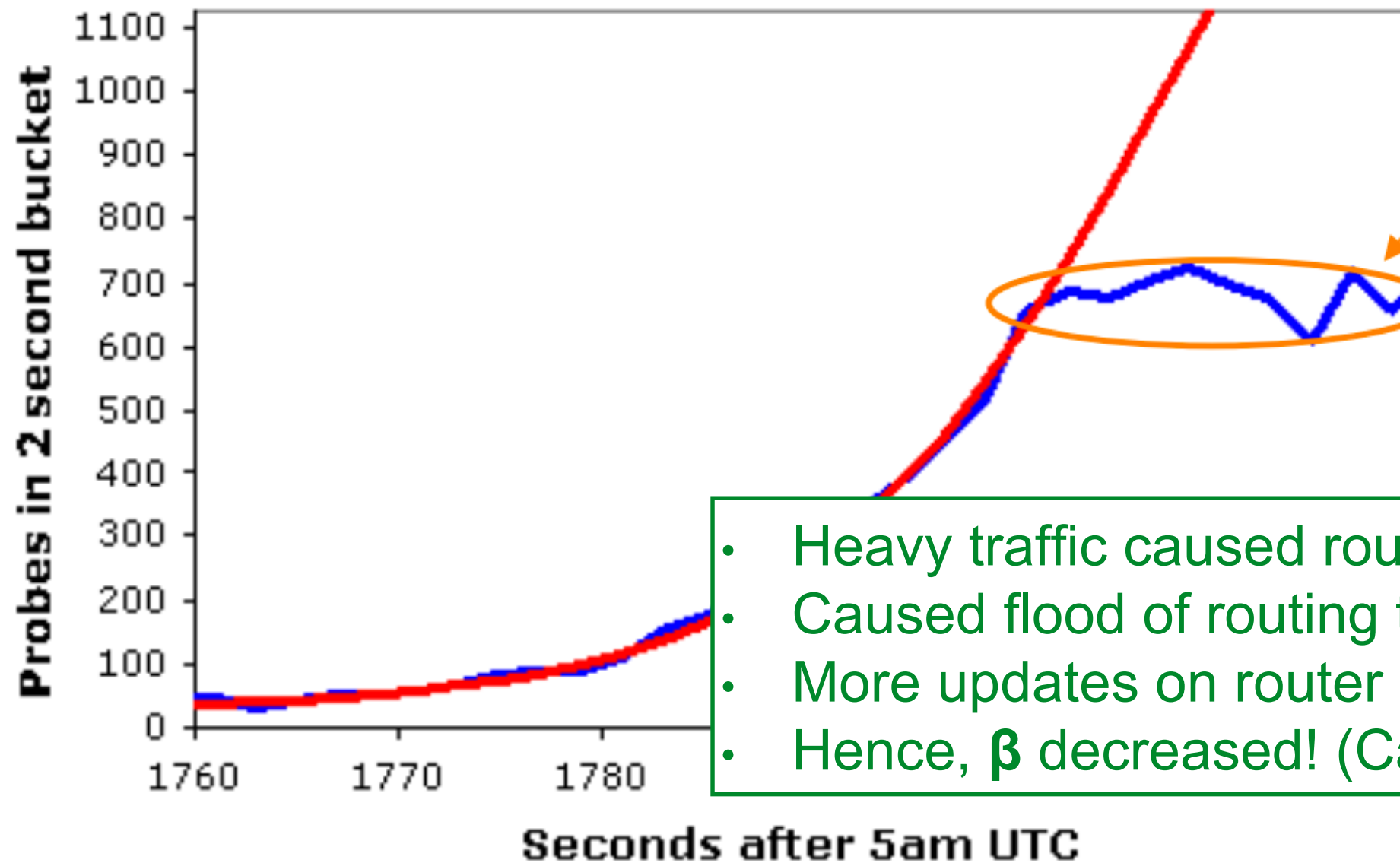


# Life just after Slammer



# Slammer's growth

DSshield Probe Data



What could have caused growth to deviate from the model?

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

- Heavy traffic caused routers to collapse
- Caused flood of routing table updates
- More updates on router restart
- Hence,  $\beta$  decreased! (Carrying capacity)

— DSshield Data —  $K=6.7/m$ ,  $T=1808.7s$ , Peak=2050, Const. 28

# Detecting modern malware

- Botnets
  - Connection to known C&C server
    - Counter: Cycle domain and use dynamic DNS
    - Re-counter: Block connections to new domains
  - “Custom” TCP and UDP
  - Generating direct email (vs. traversing mail server)
  - Anomaly detection
- Detection, not prevention**  
**All subject to arms race!**

# Modern Malware: Stuxnet

# Stuxnet: Propagation

June 2010

- **Virus:** initially spread by infected USB stick
  - Once inside network, acted as a **worm**, spreading quickly
- Exploited **four zero-day exploits**
  - Zero-day: Known to only the attacker until the attack
  - Typically, one zero-day is enough to profit
  - Four was unprecedented
    - Immense cost and sophistication on behalf of the attacker
- Rootkit: Installed *signed* device drivers
  - Thereby avoiding user alert when installing
  - Signed with **certificates stolen** from two Taiwanese CAs

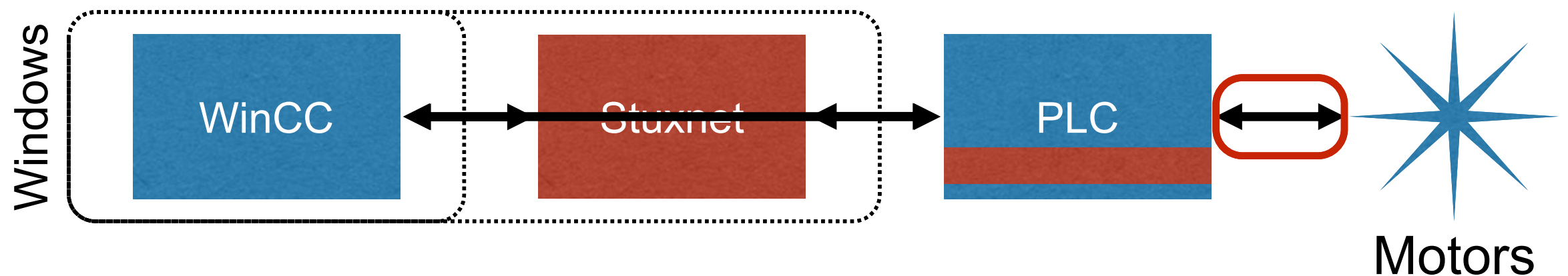
# Stuxnet: Payload

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



# Stuxnet: Payload

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



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

# Stuxnet: Fallout

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

# Malware summary

- Technological arms race between those who wish to detect and those who wish to evade detection
- Started off innocuously
- Became professional, commoditized
  - Economics, cyber warfare, corporate espionage
- Advanced detection: based on behavior, anomalies
  - Must react to attacker responses