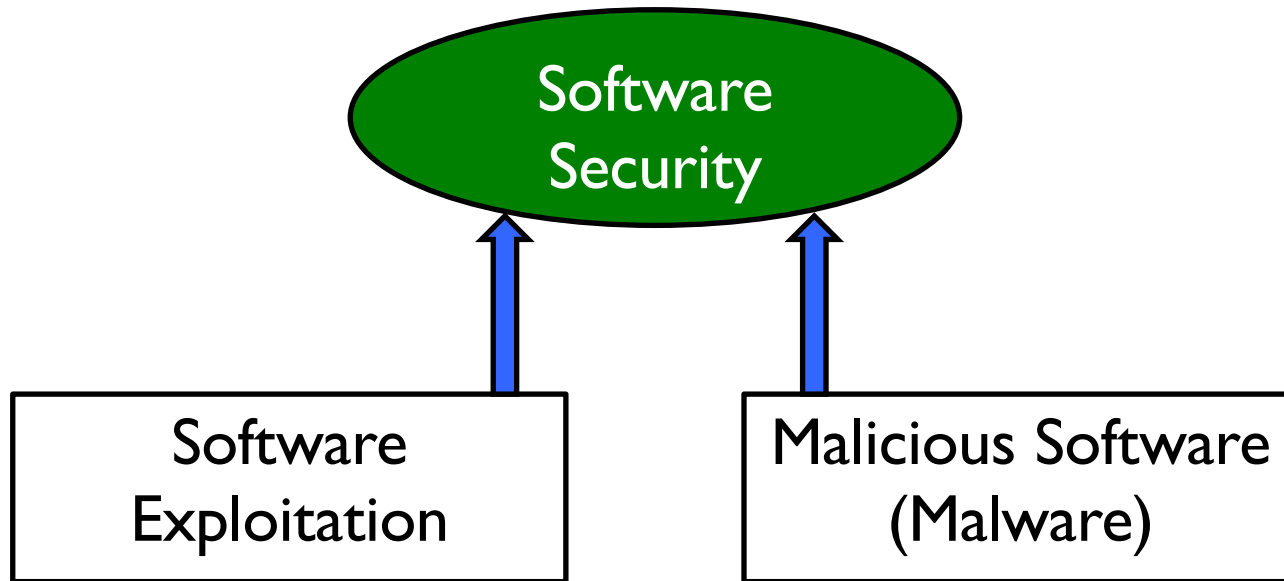


# Malware: Malicious Software

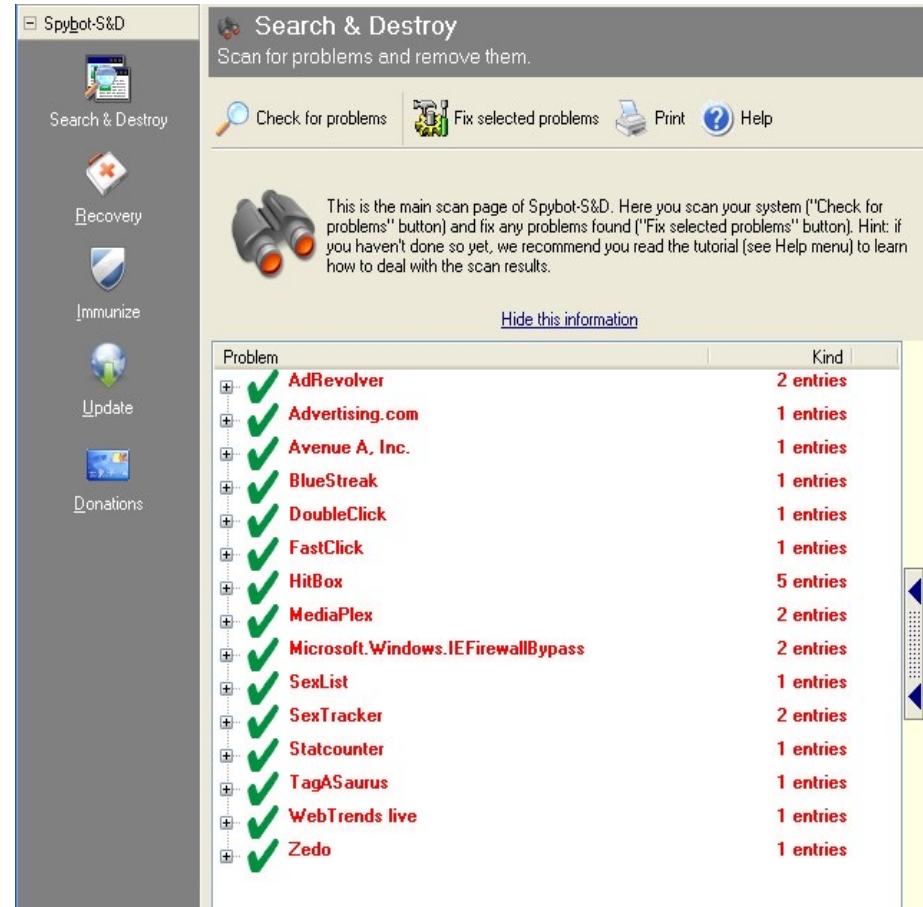
# Software Security



# What is a malware ?

## ■ Malicious software

- Virus
- Backdoor
- Trojan horse
- Rootkit
- Ransomware
- Adware
- Worm
- Bot



# 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?

- Virtually anything
  - Brag: “APRIL 1st HA HA HA HA YOU HAVE A VIRUS!”
  - Destroy:
    - Delete/mangle files
    - Damage hardware (more later this lecture)
  - Crash the machine, e.g., by over-consuming resources
    - **Fork bombing** or “rabbits”: `while(1) { fork(); }`
  - Steal information (“exfiltrate”) Launch external attacks
  - **Spam, click fraud, denial of service attacks**
  - **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
  - Use your computer as relay
    - To launch **DDoS (Distributed Denial of Service)** attacks against a victim machine
    - Sending spamming emails

# When does it run?

- Some delay based on a trigger
  - **Time bomb**: triggered at/after a certain time
    - On the 1st through the 19th of any month...
  - **Logic bomb**: triggered when a set of conditions hold
    - If I haven't appeared in two consecutive payrolls...
  - Can also include a **backdoor** to serve as ransom
    - "I won't let it delete your files if you pay me by Thursday..."
- Some attach themselves to other pieces of code
  - **Viruses**: run when the user initiates something
    - Run a program, open an attachment, boot the machine
  - **Worms**: run while another program is running
    - No user intervention required

# Self-propagating malware

- **Virus**: propagates by arranging to have itself *eventually* executed
  - At which point it creates a new, additional instance of itself
  - Typically infects by altering *stored* code
  - User intervention required
- **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

The line between these is thin and blurry  
Some malware uses both styles

# Technical Challenges

- Virus: Detection
  - Antivirus software wants to detect
  - Virus writers want to avoid detection for as long as possible
  - **Evade** human response
- Worms: Spreading
  - The goal is to hit as many machines and as quickly as possible
  - **Outpace** human response



# What is a Virus ?

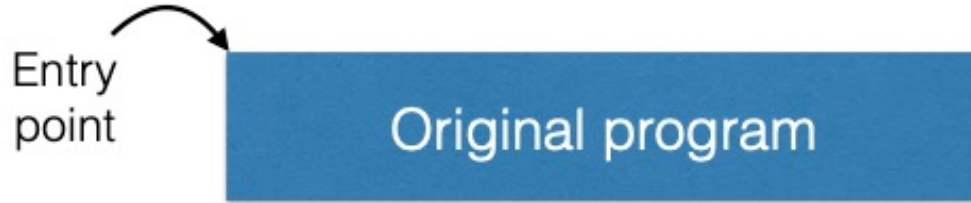
- *a program that can infect other programs by modifying them to include a, possibly evolved, version of itself*

- Fred Cohen 1983

# Viruses

- They are opportunistic: they will eventually be run due to user action
- Two orthogonal aspects define a virus:
  - How does it propagate?
  - What else does it do(what is the payload)?
- General infection strategy:
  - Alter some existing code include the virus
  - Share it, and expect users to re-share
- Viruses have been around since at least the 70s

# How viruses infect other programs



# How viruses infect other programs



# Viruses are classified by what they infect

- Document viruses
  - implemented within a formatted document
  - Word documents
  - PDF(Acrobat permits javascript)
  - This is why you shouldn't open random attachments
- Boot sector viruses
  - Boot sector: small disk partition at a fixed location
  - if the disk is used to boot, then the firmware loads the boot sector code into memory and runs it
  - What's supposed to happen: this code loads the OS
  - Similar: AutoRun on music/video disks
  - This is why you shouldn't plug random USB drives into your computer
- Memory-resident viruses
  - Resident code stays in memory because it is used so often

# How viruses propagate

- First, the virus looks for an **opportunity to run**
- Increase chances by attaching malicious code to something a user is likely to run
  - autorun.exe on storage devices
  - Email attachments
- When a virus run, it looks for an opportunity to infect other systems
  - User plugs in a USB thumb drive: try to overwrite autorun.exe
  - User is sending an email: alter the attachment

# Detecting Viruses

- Method 1: Signature-based detection
  - Look for bytes corresponding to injected virus code
  - Protect other systems by installing a recognizer for a known virus
  - In practice, requires fast scanning algorithms
- This basic approach has driven the multi-billion dollar antivirus market
- Recognized signatures is a means of marketing and competition
  - But what does that say about how important they are?

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: 151234

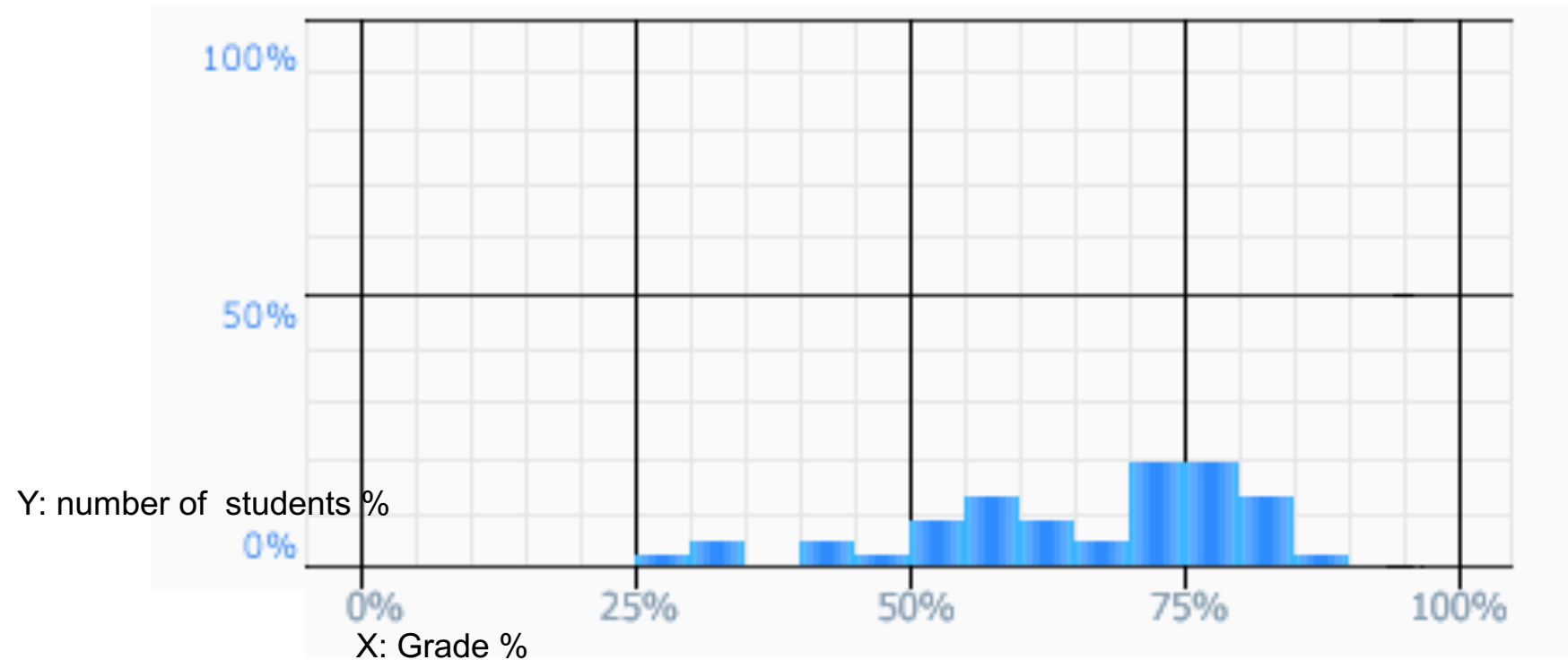
Number of Signatures: 23,927,535

Details: [Release History](#)

Download: [Definitions](#) , Content is downloaded by your product via LiveUpdate.



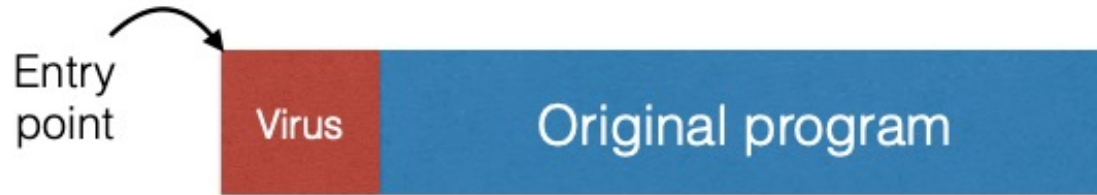
# Midterm Statistic



Average: 65, Median: 73

# If you are a virus writer

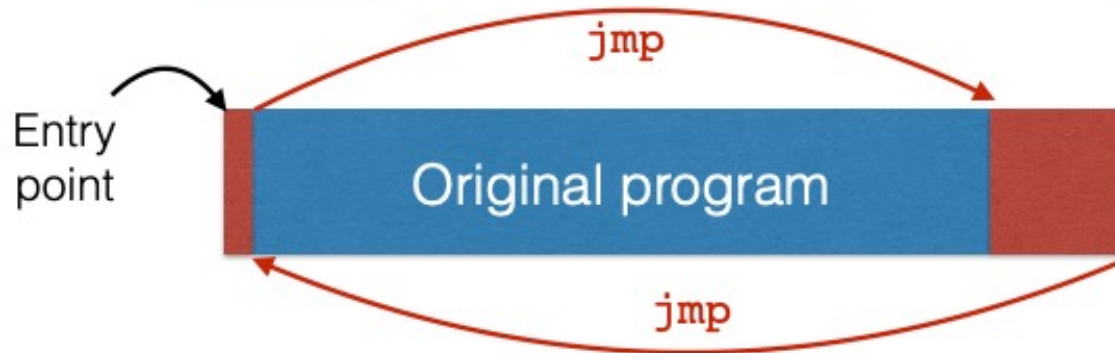
- Your goal is for your virus to spread far and wide
- How do you avoid detection by antivirus software?
  - 1. Give them a harder signature to find



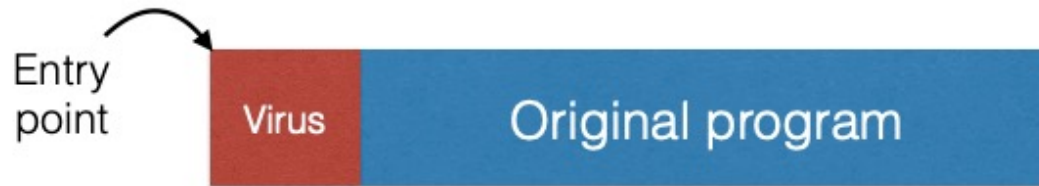
“Appending”



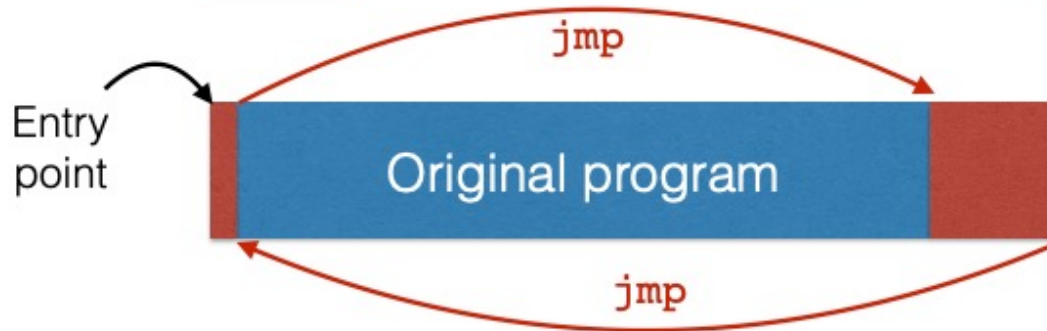
“Appending”



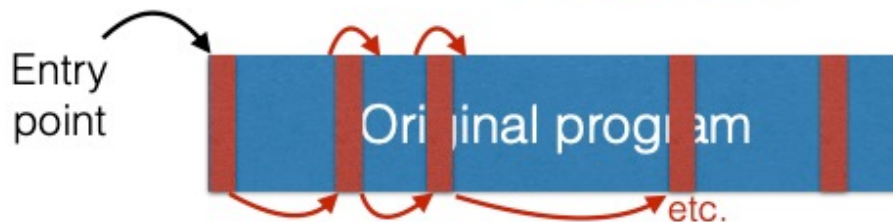
“Surrounding”



“Appending”



“Surrounding”



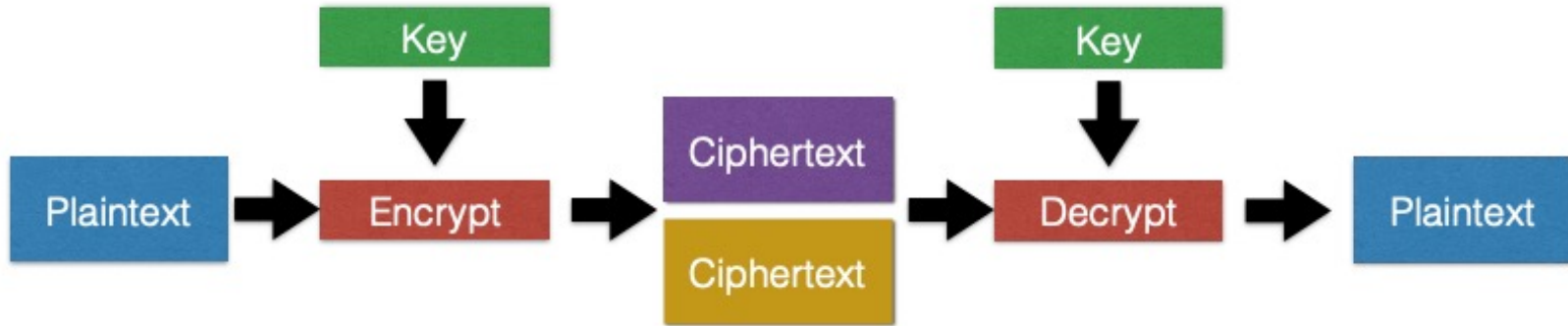
Confuse  
scanners

Overwrite uncommonly  
used parts of the program

# If you are a virus writer

- Your goal is for your virus to spread far and wide
- How do you avoid detection by antivirus software?
  - 1. Give them a harder signature to find
  - 2. Change your code so they can't pin down a signature
    - Goal: every time you inject your code, it looks different

# Encryption



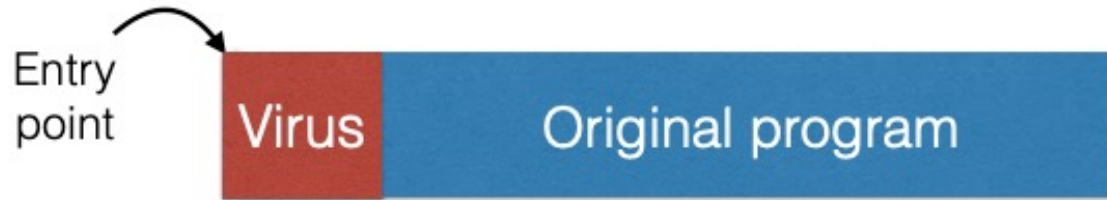
Symmetric key: both keys are the same

Asymmetric key: different keys

Important property: the ciphertext is ***nondeterministic***  
i.e., “Encrypt” has a different output each time

but decrypting always returns the plaintext

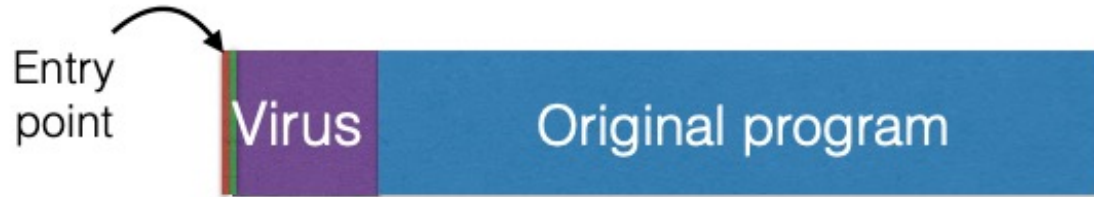
# Polymorphic viruses



Take over the  
entry point

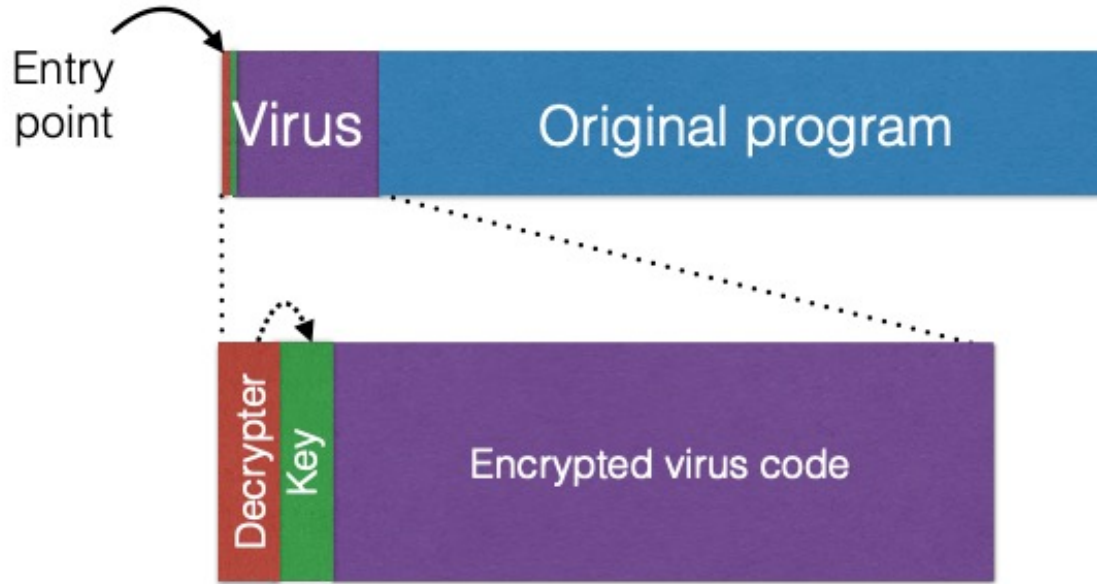


# Polymorphic viruses



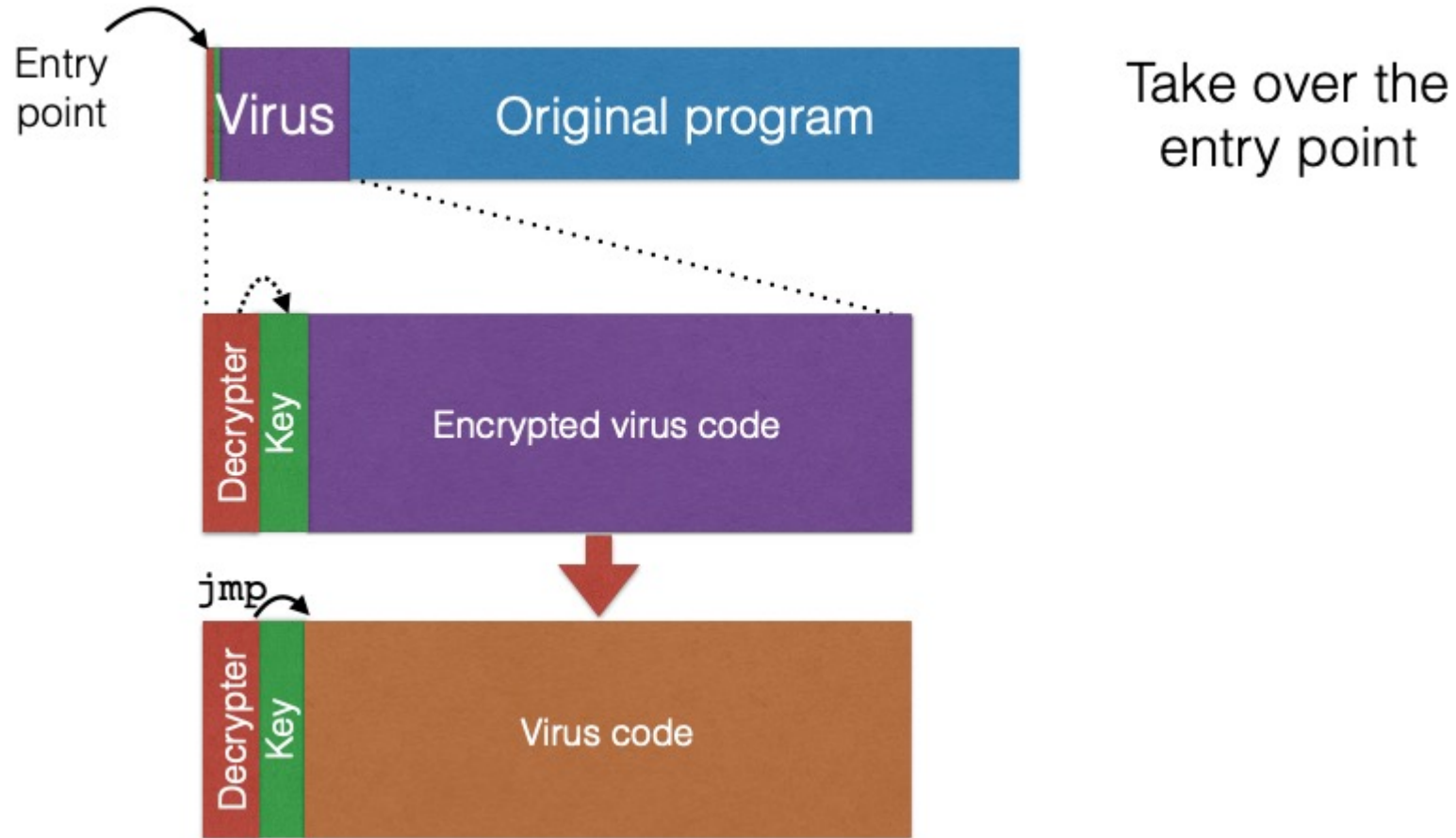
Take over the  
entry point

# Polymorphic viruses

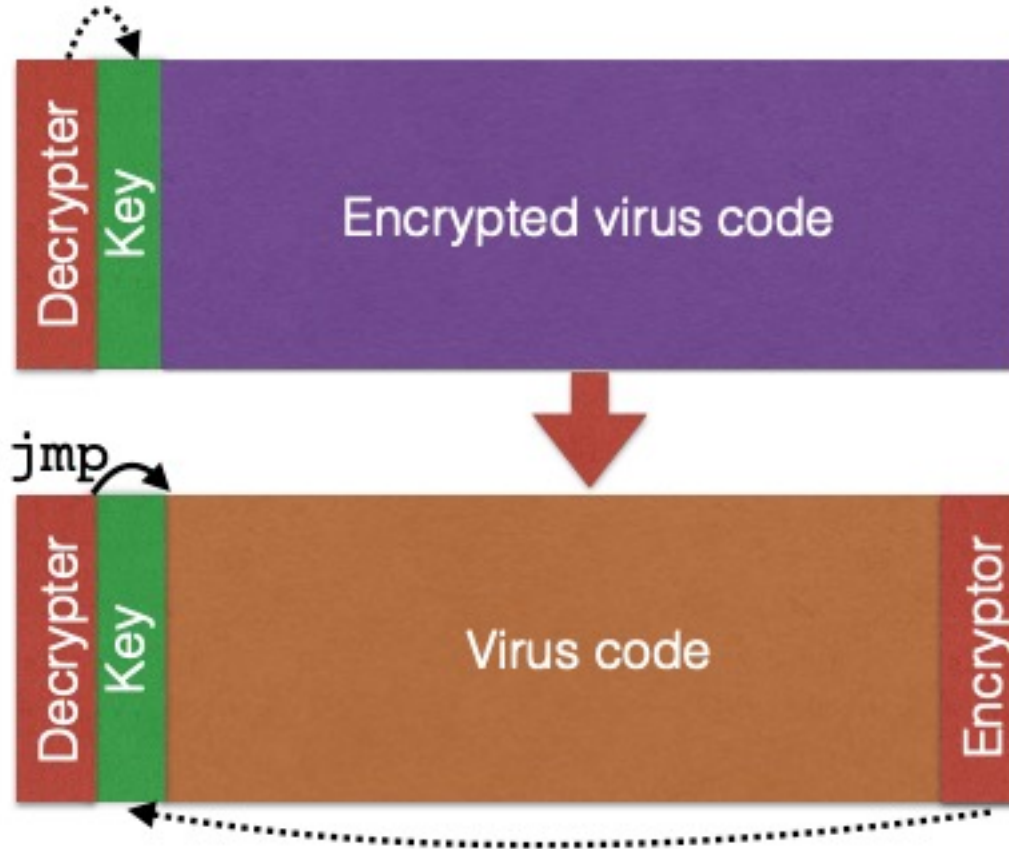


Take over the entry point

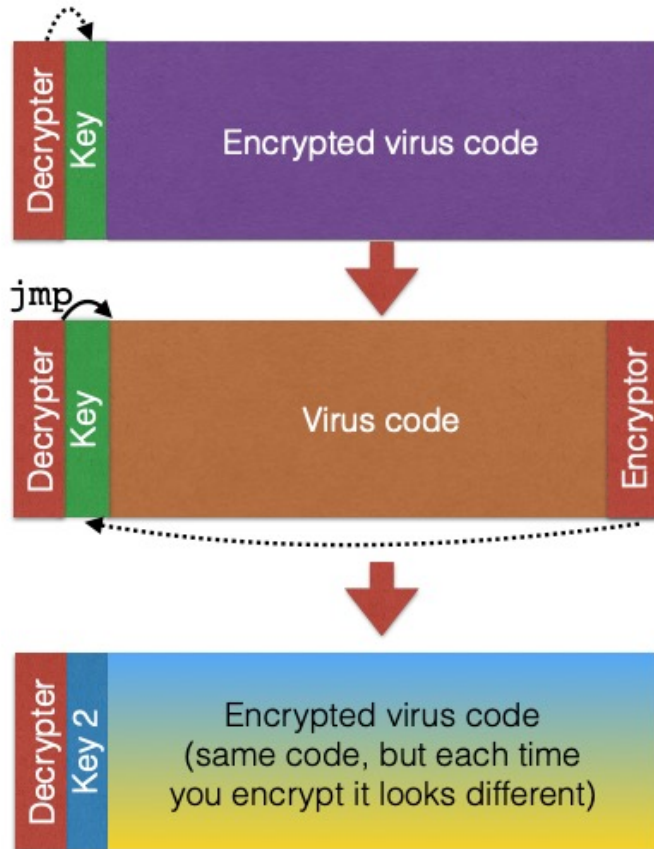
# Polymorphic viruses



# Polymorphic viruses



# Polymorphic viruses



# Polymorphic viruses

- 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?
    - How long do you execute the code?

# Polymorphic countermeasures

- **Oligomorphic viruses:** change to one of a fixed set of decrypters
  - Variations are limited by the virus code
  - Still finite range of detectable patterns
- **True polymorphic viruses:** can generate an endless number of decrypters
  - Theoretically infinite number of decrypters
  - e.g., brute force key break
  - Downside: inefficient

# Metamorphic code

- 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
    - Translate code into semantic different version
  - Add a bunch of complex code to throw others off



# 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)
  - anti-virus company analyzes new virus to find **behavioral signature**
  - anti-virus system at the end host analyzes suspect code to see if it **matches** the signature

# Detecting metamorphic viruses

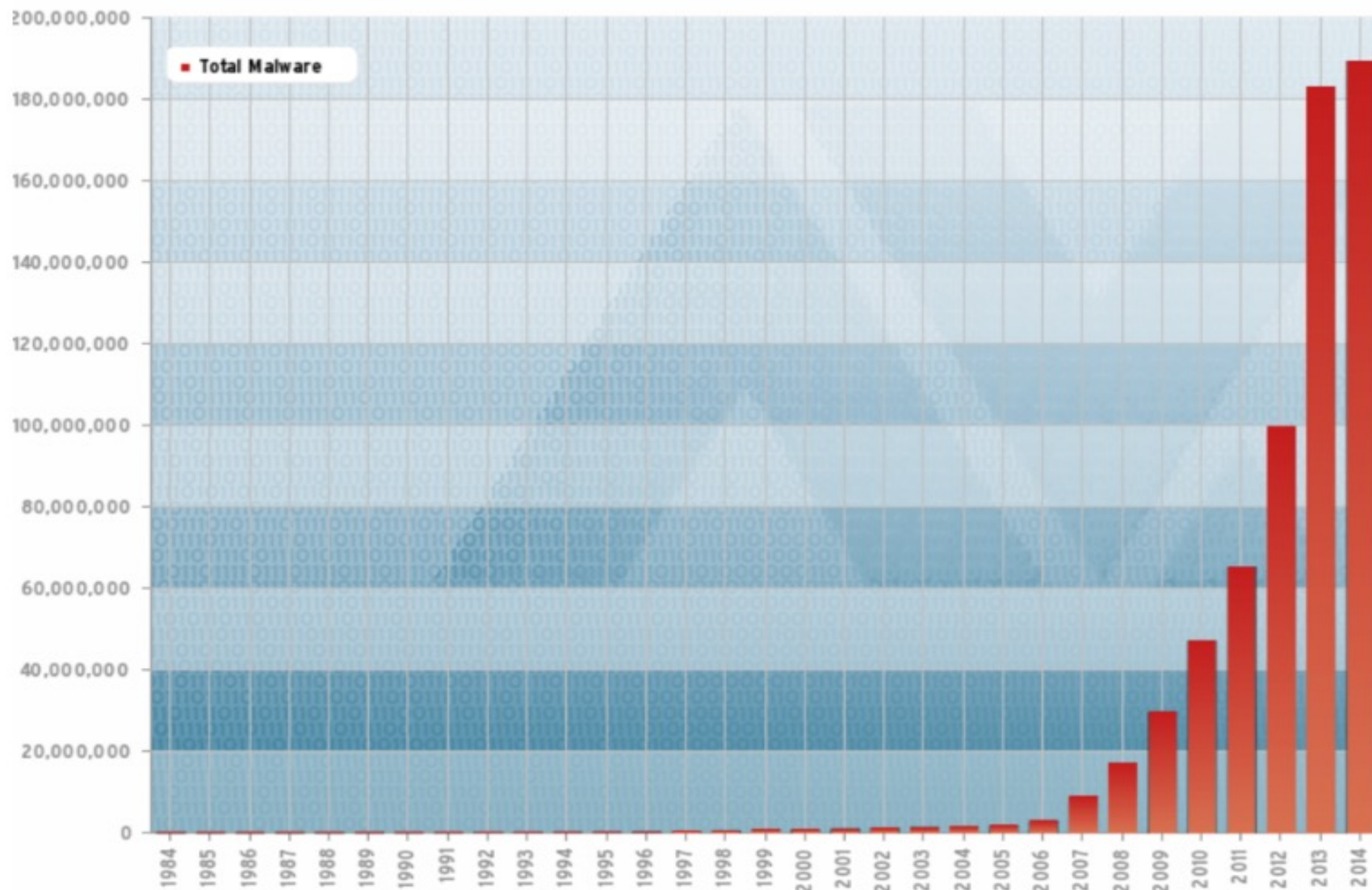
- Countermeasures
  - Have your virus change slowly(hard to create a proper behavioral signature)
    - Apply with longer timer for behaviors
  - Detect if you are in a safe execution environment(e.g.,gdb) and act differently
    - Checking parent process name, ptrace...
- Counter-countermeasures
  - **Detect detection** and skip those parts

# Putting it all together

- Creating a virus can be really difficult
  - Historically error prone
- But using them is easy: any scriptkiddy can use Metasploit
  - A penetration testing software
  - Good news: so can any white hat pen test

# How much malware is out there?

- Polymorphic and metamorphic viruses can make it easy to miscount viruses
  - Theoretically infinite viruses
- Take numbers with grain of salt
  - Large numbers are in the anti-virus vendors' best interest



# How do we clean up an infection?

- Depends what the virus did
- 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

# Definition of Other Malware Types

# What is a Trojan (or Trojan horse)

*A trojan describes the class of malware that appears to perform a desirable function but in fact performs undisclosed malicious functions that allow unauthorized access to the victim computer*



Wikipedia



# What is a worm

*A computer worm is a self-replicating computer program. It uses a network to send copies of itself to other nodes and do so **without any user intervention**.*



Image courtesy of: Tech Tips.com

# What is a bot

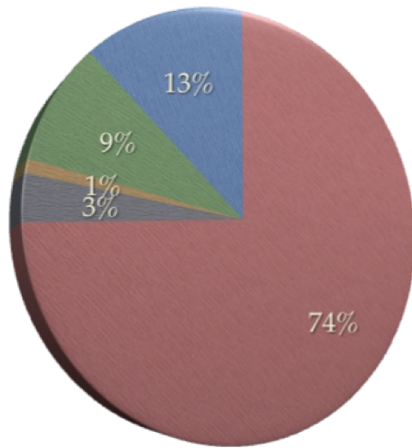
- A **bot** (or a zombie) is a computer that a remote attacker has accessed and set up to forward transmissions (including spam and viruses) to other computers on the Internet.
- A **botnet** is a collection of bot machines that are under the control of a human operator commonly known as a *bot master*.



# Adware, Spyware, Scareware

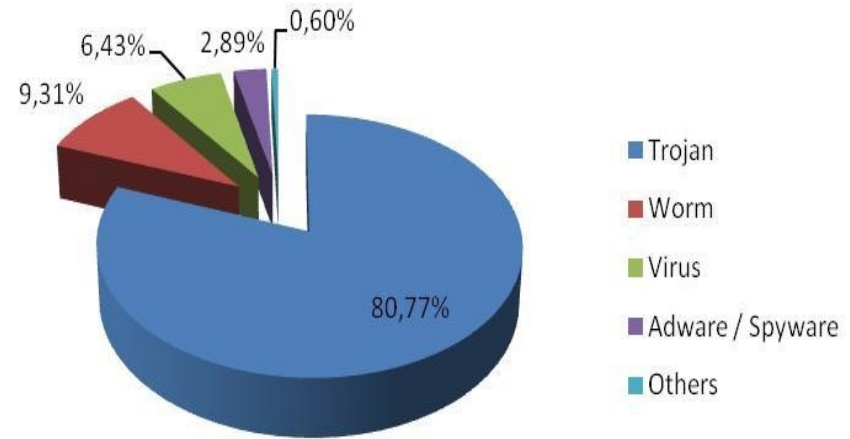
- **Adware** is any software package that automatically renders advertisements in order to generate revenue for its author.
- **Spyware** is software that aids in gathering information about a person or organization without their knowledge and that may send such information to another entity without the consumer's consent, or that asserts control over a computer without the consumer's knowledge.
- **Ransomware** is a type of malicious software designed to block access to a computer system until a sum of money is paid.

# Malware numbers by categories



Panda Q1 report 2009

- Trojan
- Worm
- Other
- Adware
- Spyware



Panda Q1 report 2012

# Other Infection methods

# What to Infect

- Executable
- Interpreted file
- Rootkit
  - Userland rootkit
  - Kernel rootkit
  - Bootkit
  - Hypervisor rootkit



# Infecting executables - Unix autostart

- **/etc/init.d**: a directory containing initialization and termination scripts for changing init states
- **/etc/rc.local**: runlevel control (rc), called by init
- **.login**: every time you log in
- **.xsession**: starts a GUI session
- **crontab**
  - crontab -e
  - /etc/crontab

The software utility **cron** is a time-based job scheduler in Unix-like Operating Systems.

# Infecting interpreted files: macro virus

- Use the builtin script engine
- Example of call back used (Word)
  - AutoExec()
  - AutoClose()
  - AutoOpen()
  - AutoNew()



# What is rootkit

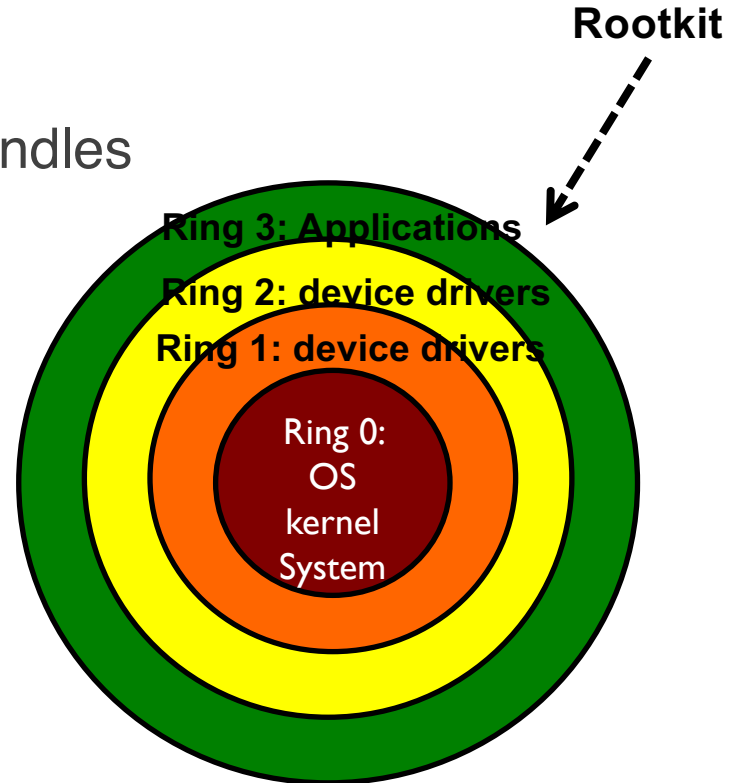
- A rootkit is a component that uses stealth to maintain a persistent and undetectable presence on the machine

- Symantec

# Rootkit (1): Userland rootkit

- Hide their processes
- Hide registry keys
- Hide files, windows and handles

For example a userland rootkit who wants to hide registry information from a windows application which uses libraries such as user32.dll, kernel32.dll, or Advapi32.dll.




# Azazel userland rootkit source code:

<https://github.com/chokepoint/azazel>

Rely on LD\_PRELOAD technique

Intercept **fopen** function call

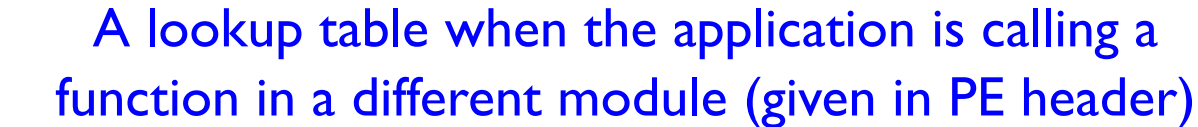


```
FILE *fopen (const char *filename, const char *mode) {
    DEBUG("fopen hooked %s.\n", filename);
    if (is_owner())
        syscall_list[SYS_FOPEN].syscall_func(filename, mode);

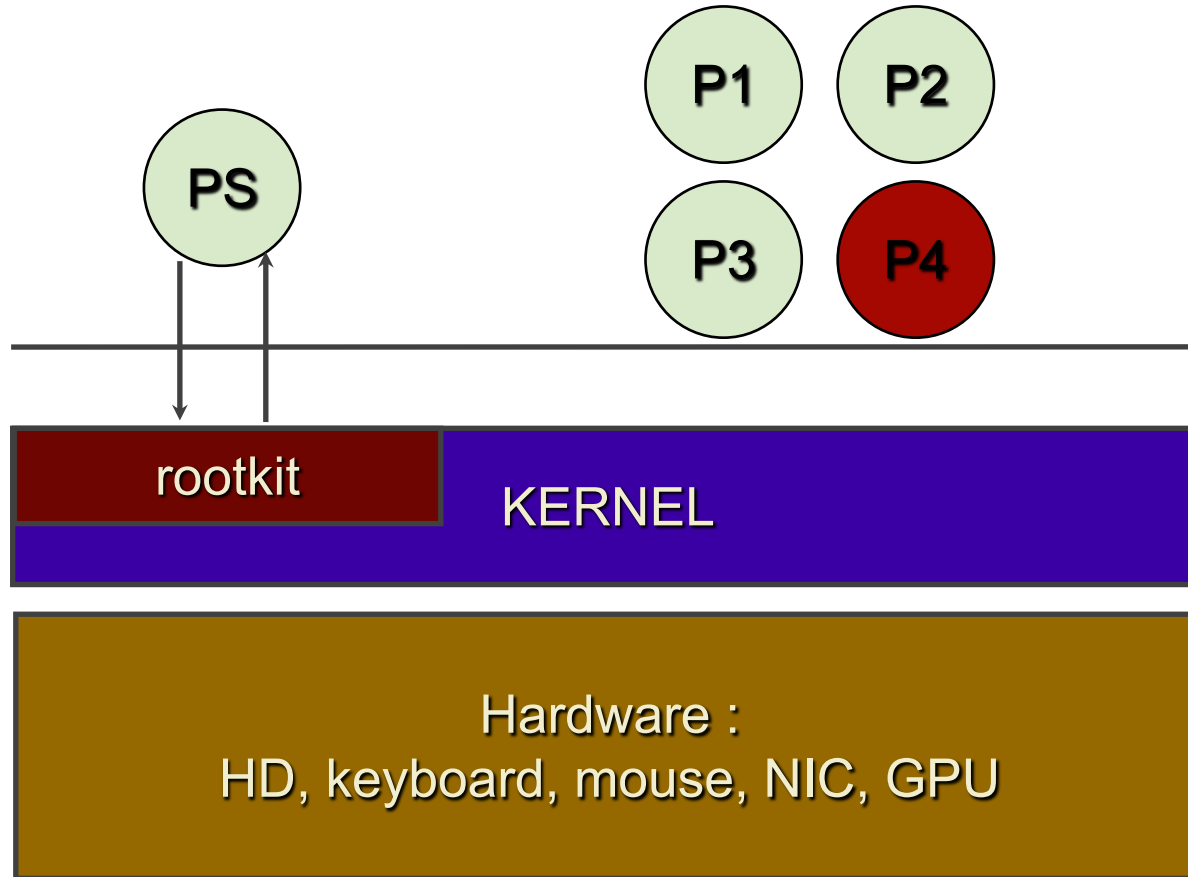
    if (is_procnet(filename))
        return hide_ports(filename);

    if (is_invisible(filename)) {
        errno = ENOENT;
        return NULL;
    }
    return syscall_list[SYS_FOPEN].syscall_func(filename, mode);
}
```

## 1



# Rootkit (2): Kernel rootkit

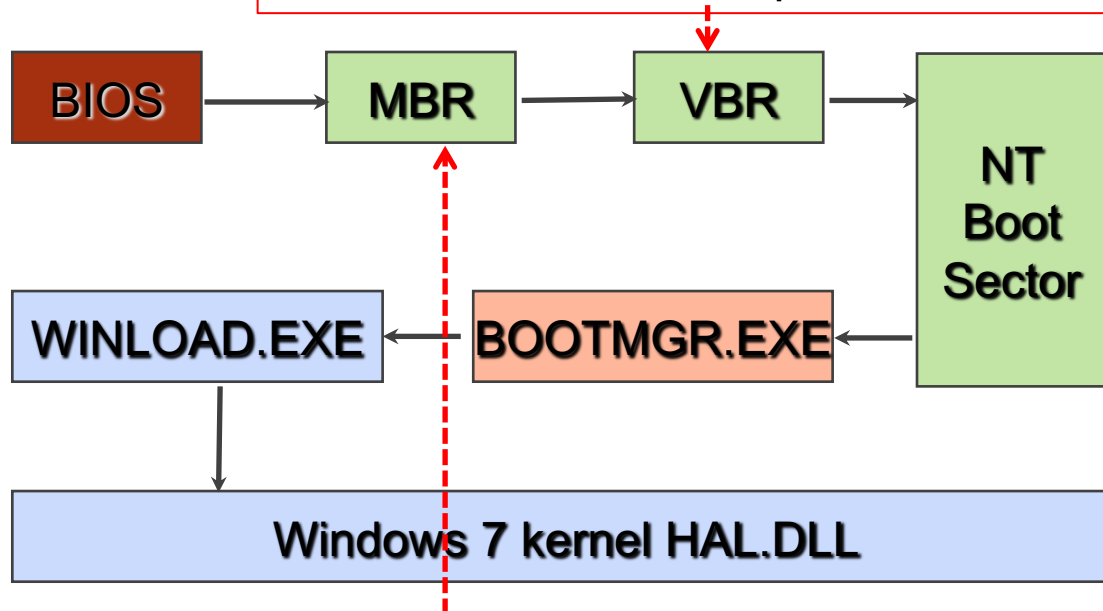


# Rootkit (2): Kernel rootkit

- Kernel rootkits run with the most highest operating system privileges (**Ring 0**).
- They **add or replace pieces of code of the operating system** to modify kernel behavior.
- This class of rootkit has **unrestricted security access**.
- Kernel rootkits can be **especially difficult to detect and remove** because they operate at the same security level as the operating system itself.

# Rootkit (3): Bootkit -- Infecting the boot process

VBR/VBS: Volume boot record/sector (first sector of an individual partition on a device)

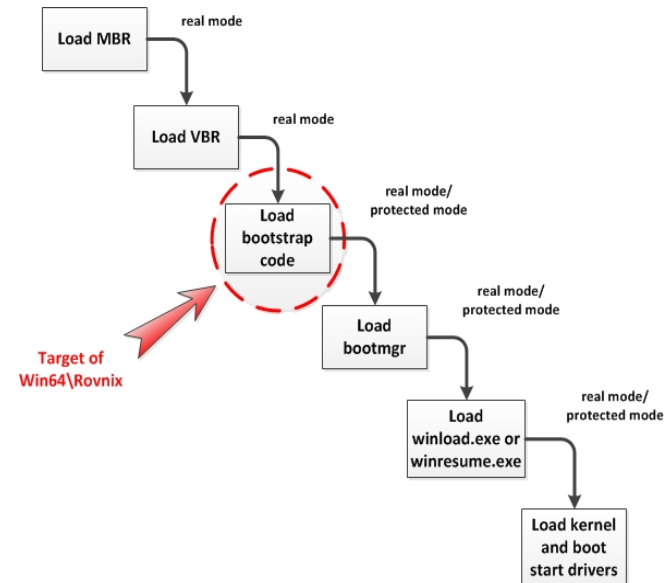
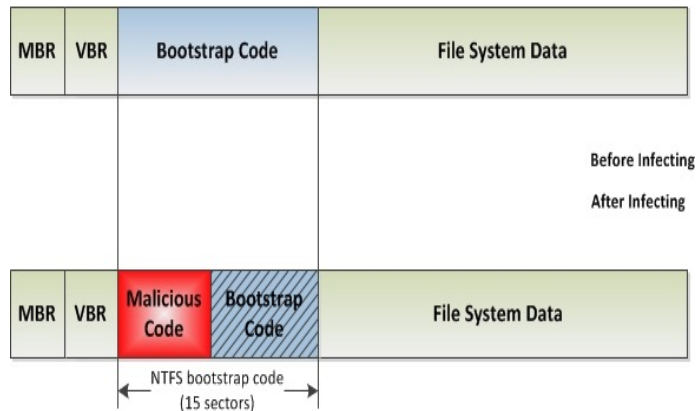


A master boot record (MBR) is a special type of boot sector at the very beginning of partitioned computer mass storage devices like fixed disks or removable drives intended for use with IBM PC-compatible systems and beyond.

# Rootkit (3): Advantage of bootkits

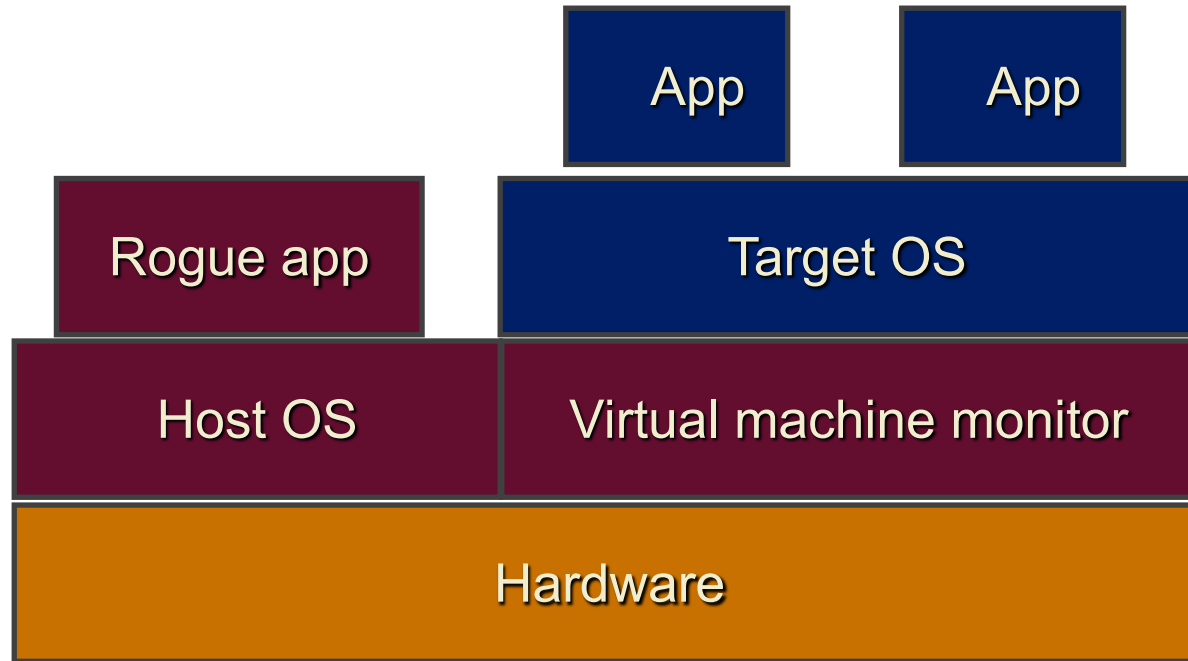
- Bootkits can be used to avoid all protections of an OS, because OS consider that the system was in a trusted state at the moment the OS boot loader took control.

## Rovnix bootkit





# Hypervisor rootkit

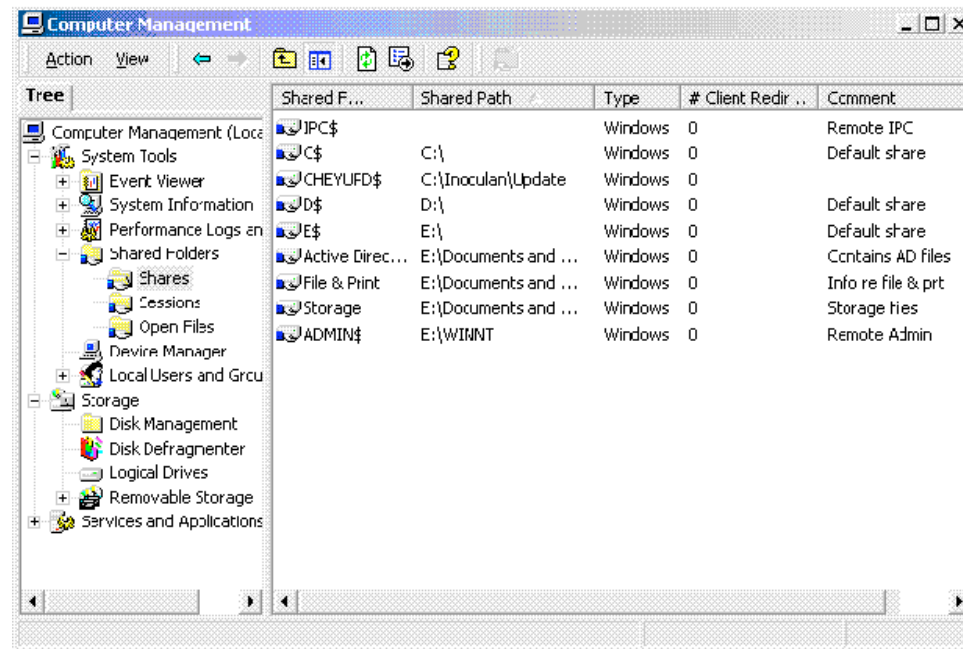


# Sony XCP rootkit

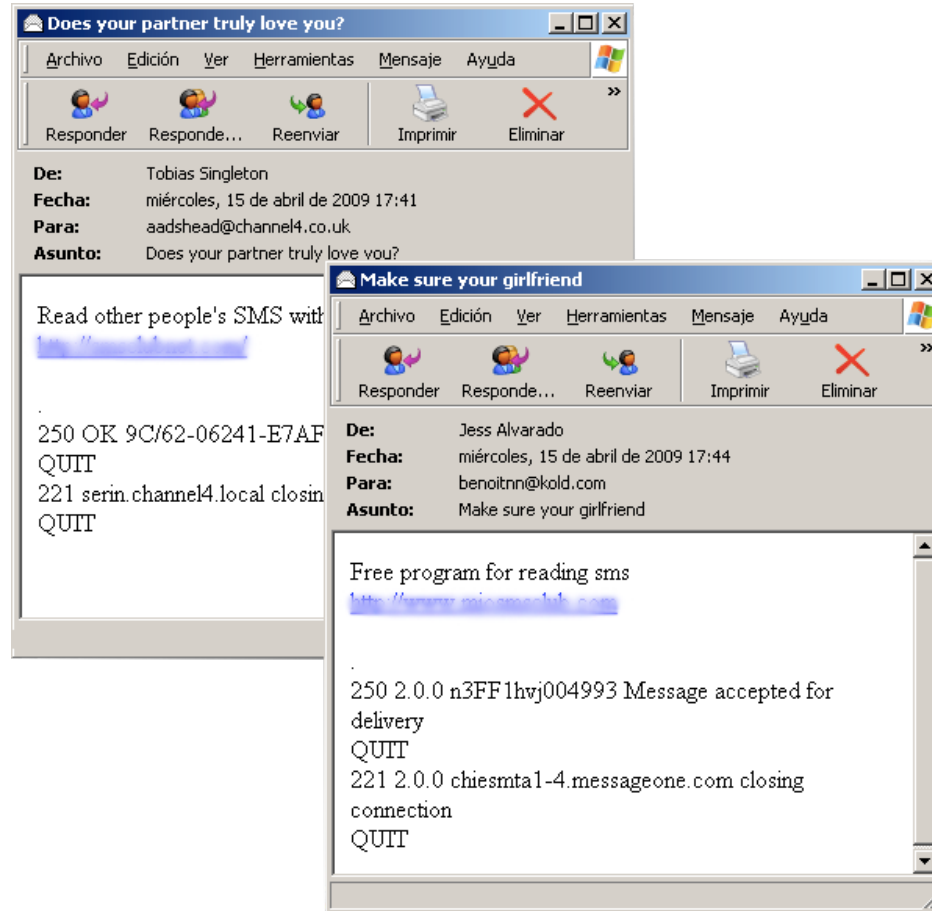
- Goal: keep users from copying copyrighted material
- How it worked:
  - Loaded thanks to autorun.exe on the CD
  - *Intercepted* read requests for its music files
  - If anyone but Sony's music player is accessing them, then garble the data
  - Hid itself from the user (to avoid deletion)
- How it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

# Propagation Vector

# Shared folder

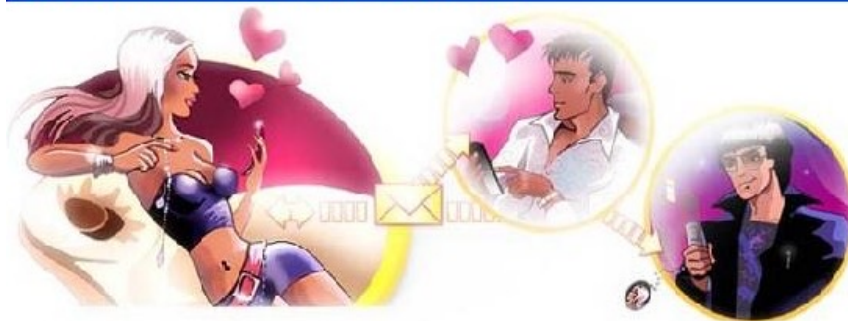


# Email propagation



# Email again

Are you interested in reading other people's sms?



Get Your Free 30-Day Trial!

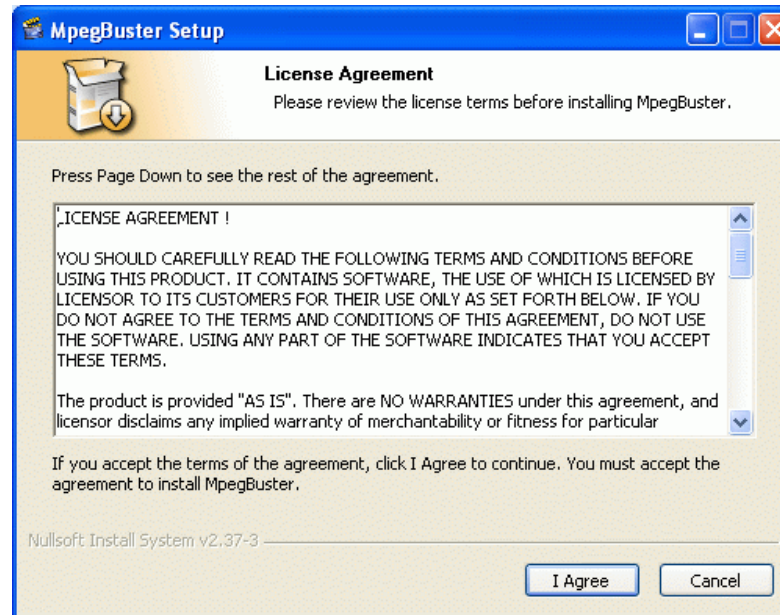
Do you want to test your partner or just to read somebody's SMS?  
This program is exactly what you need then! It's so easy! You don't  
need to install it at the mobile phone of your partner.  
Just download the program and you will be able to read all SMS  
when you are online. Be aware of everything! This is an  
extremely new service!

[http://\[Removed\].com/freetrial.exe](http://[Removed].com/freetrial.exe)

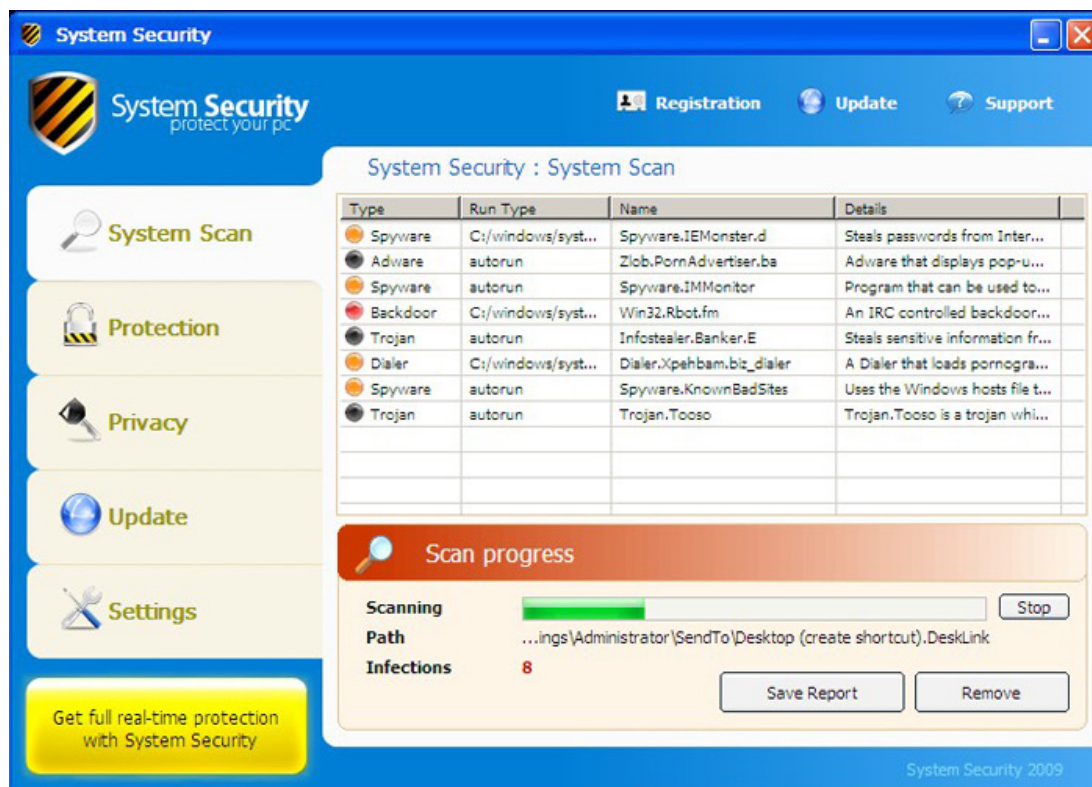
Download Free Trial

© SMS Spy. All rights reserved

# Fake codec

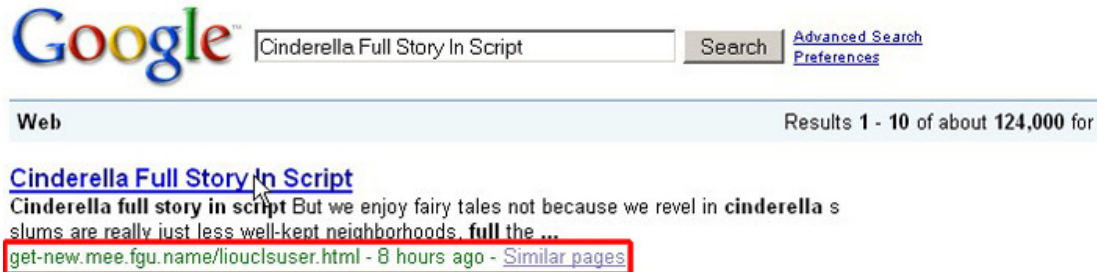


# Fake antivirus

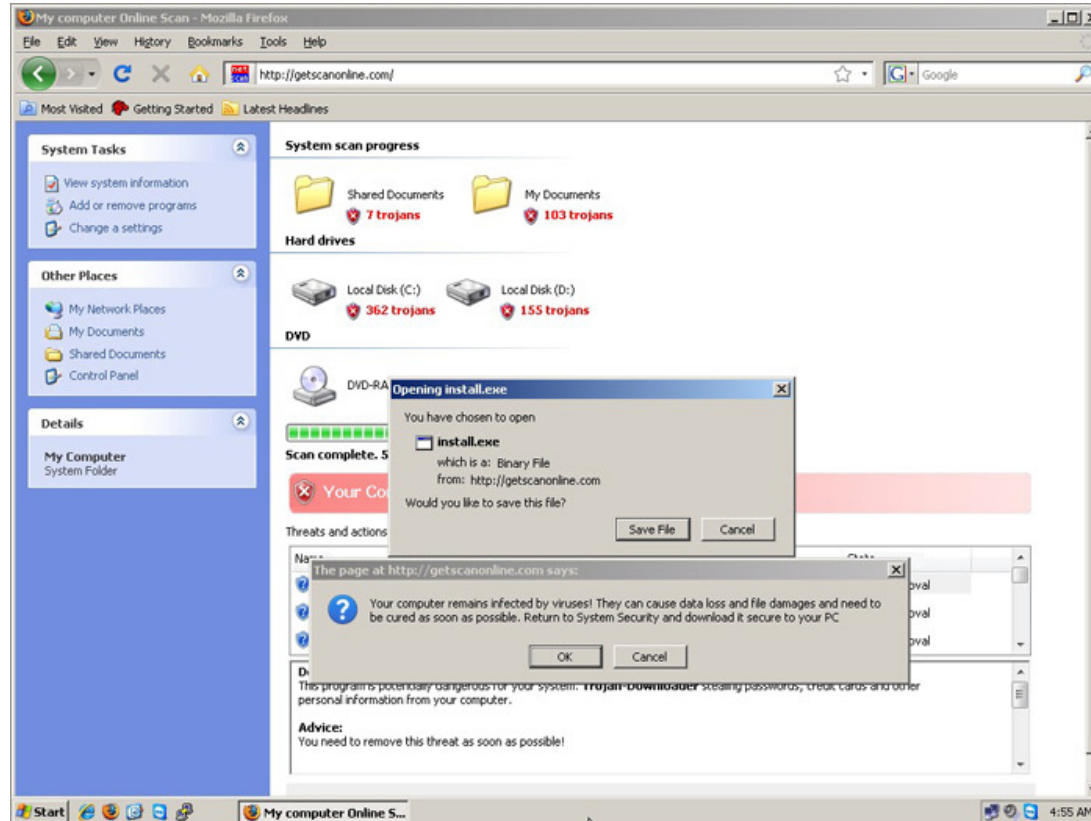




# Hijack you browser

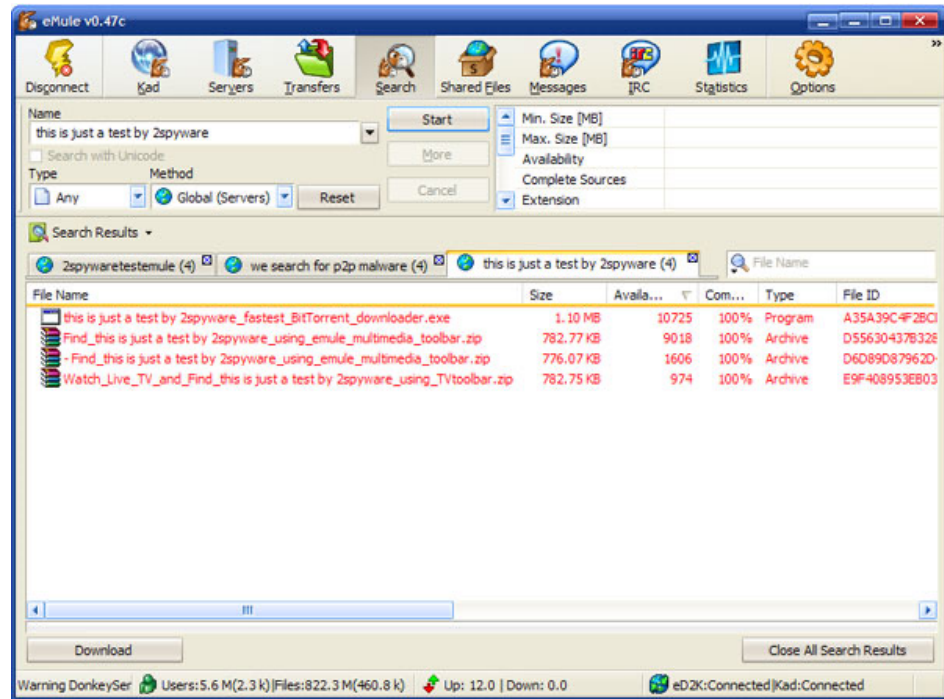


# Fake page !



# P2P Files

- Popular query
- 35.5% are malwares



# Backdoor

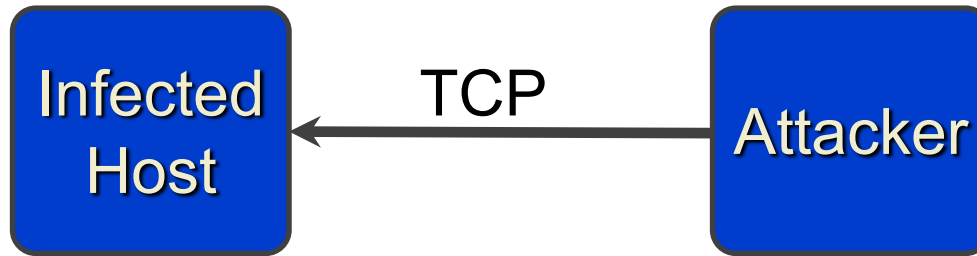
# What is a backdoor

- A **backdoor** in a computer system (or cryptosystem or algorithm) is a method of bypassing normal authentication, securing remote access to a computer, obtaining access to plaintext, and so on, while attempting to remain undetected.

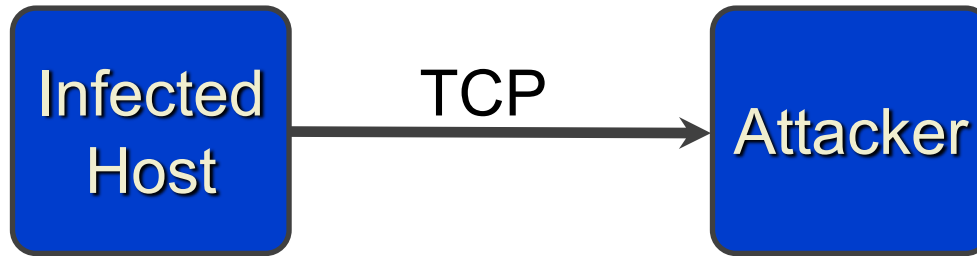


Wikipedia

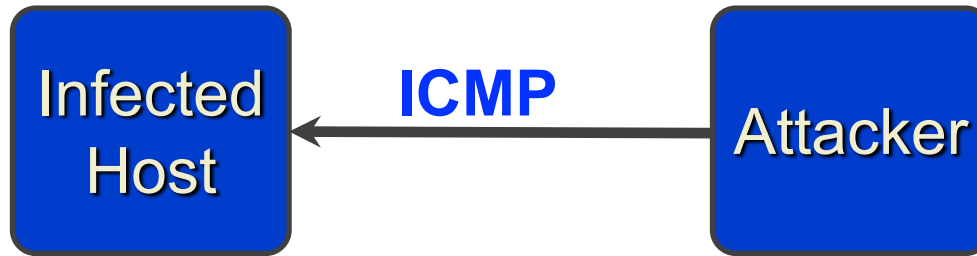
# Basic



# Reverse

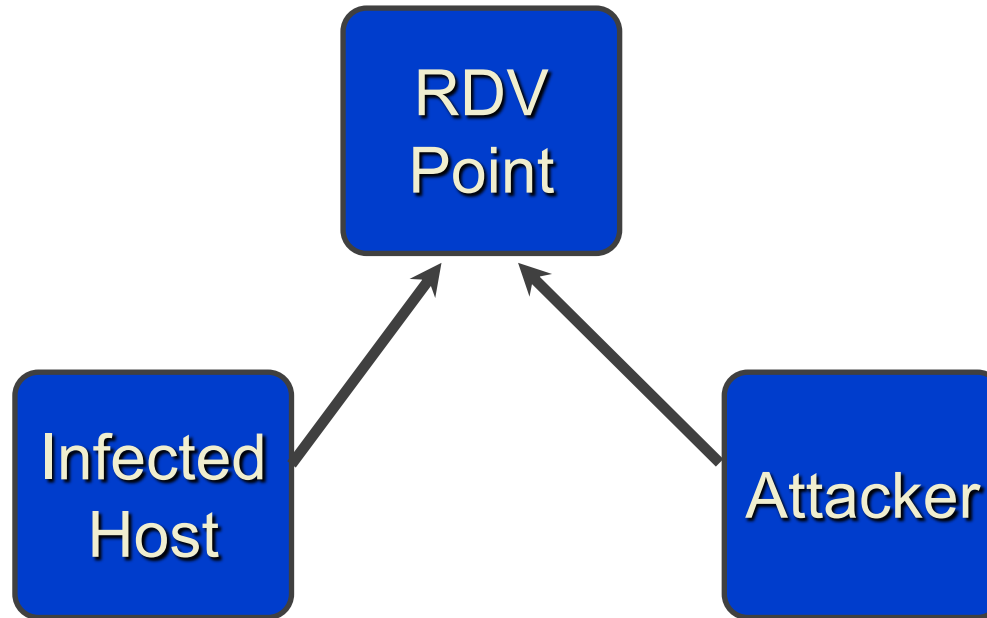


# Covert





# Rendezvous backdoor



# Adware



# Ransomware

## ■ April 2019: REvil

Your network has been infected!



Your documents, photos,  
databases and other important files  
encrypted



To decrypt your files you need to  
buy our special software - General-  
Decryptor



Follow the instructions below. But  
remember that you do not have  
much time

General-Decryptor price  
the price is for all PCs of your infected network

You have 8 days, 20:59:12

\* If you do not pay on time, the price will be doubled

\* Time ends on Mar 28, 16:30:11

Current price

214151 xMR  
≈ 50,000,000 USD

After time ends

428302 xMR  
≈ 100,000,000 USD

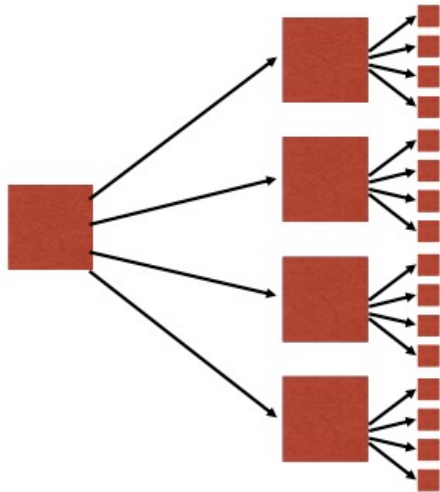
# Worms

# Worm

- A worm is **self-replicating** software designed to spread through the network
  - Typically, exploit security flaws in widely used services
  - Can cause enormous damage
    - Launch DDOS attacks, install botnets
    - Access sensitive information
    - Cause confusion by corrupting the sensitive information
- Worm vs Virus vs Trojan horse
  - A virus is code embedded in a file or program
  - Viruses and Trojan horses rely on human intervention to spread
  - Worms are self-contained and may spread autonomously

# Worm 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?

# Cost of worm attacks

- **Morris worm, 1988**
  - Variety of attacks: buffer overflow, crack passwords
  - Infected approximately 6,000 machines, 10% of computers connected to the Internet
  - cost ~ \$10 million in downtime and cleanup
- **Code Red worm, July 16 2001**
  - Infected more than 500,000 servers
  - Caused ~ \$2.6 Billion in damages
- **Love Bug worm: \$8.75 billion**
  - Statistics: Computer Economics Inc., Carlsbad, California

# Some historical worms of note

Worm	Date	Distinction
Morris	11/88	Used multiple vulnerabilities, propagate to “nearby” sys
ADM	5/98	Random scanning of IP address space
Ramen	1/01	Exploited three vulnerabilities
Lion	3/01	Stealthy, rootkit worm
Cheese	6/01	Vigilante worm that secured vulnerable systems
Code Red	7/01	First sig Windows worm; Completely memory resident
Walk	8/01	Recompiled source code locally
Nimda	9/01	Windows worm: client-to-server, c-to-c, s-to-s, ...
Scalper	6/02	11 days after announcement of vulnerability; peer-to-peer network of compromised systems
Slammer	1/03	Used a single UDP packet for explosive growth



# Worm propagation speed

- Code Red, July 2001
  - Affects Microsoft Index Server 2.0,
    - Windows 2000 Indexing service on Windows NT 4.0.
    - Windows 2000 that run IIS 4.0 and 5.0 Web servers
  - Exploits known **buffer overflow** in Idq.dll
  - **Vulnerable population (360,000 servers) infected in 14 hours**
- SQL Slammer, January 2003
  - Affects Microsoft SQL 2000
  - Exploits known **buffer overflow** vulnerability
    - Server Resolution service vulnerability reported June 2002
    - Patches released in July 2002 Bulletin MS02-39
  - **Vulnerable population infected in less than 10 minutes**

# Code Red

- Initial version released July 13, 2001
  - Sends its code as an HTTP request
  - HTTP request exploits **buffer overflow**
  - Malicious code is not stored in a file
    - **Placed in memory** and then run
- When executed,
  - Worm checks for the file C:\Notworm
    - If file exists, the worm thread goes into infinite sleep state
  - Payload: Time bomb
    - If the date is before the 20th of the month, the next 99 threads attempt to exploit more computers by targeting random IP addresses
    - If the date is after the 20th, attack(flood whitehouse.gov)

# Code Red of July 13 and July 19

- Initial release of July 13
- 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
- Revision released July 19, 2001.
  - White House responds to threat of flooding attack by changing the ip address of [www.whitehouse.gov](http://www.whitehouse.gov)
  - Causes Code Red to die for date  $\geq 20^{\text{th}}$  of the month.
  - ... except for hosts with inaccurate clocks!
    - It just takes one of these to restart the worm on August 1<sup>st</sup>

# Code Red 2

- Released August 4, 2001
- Comment in code: “Code Red 2.”
  - **But in fact, completely different code base.**
- Payload: a root backdoor, resilient to reboots
- Bug: crashes NT, only works on Windows 2000.
- Localized scanning: prefers nearby addresses
- By then, many but not all hosts patched
- Safety valve: programmed to die Oct 1, 2001

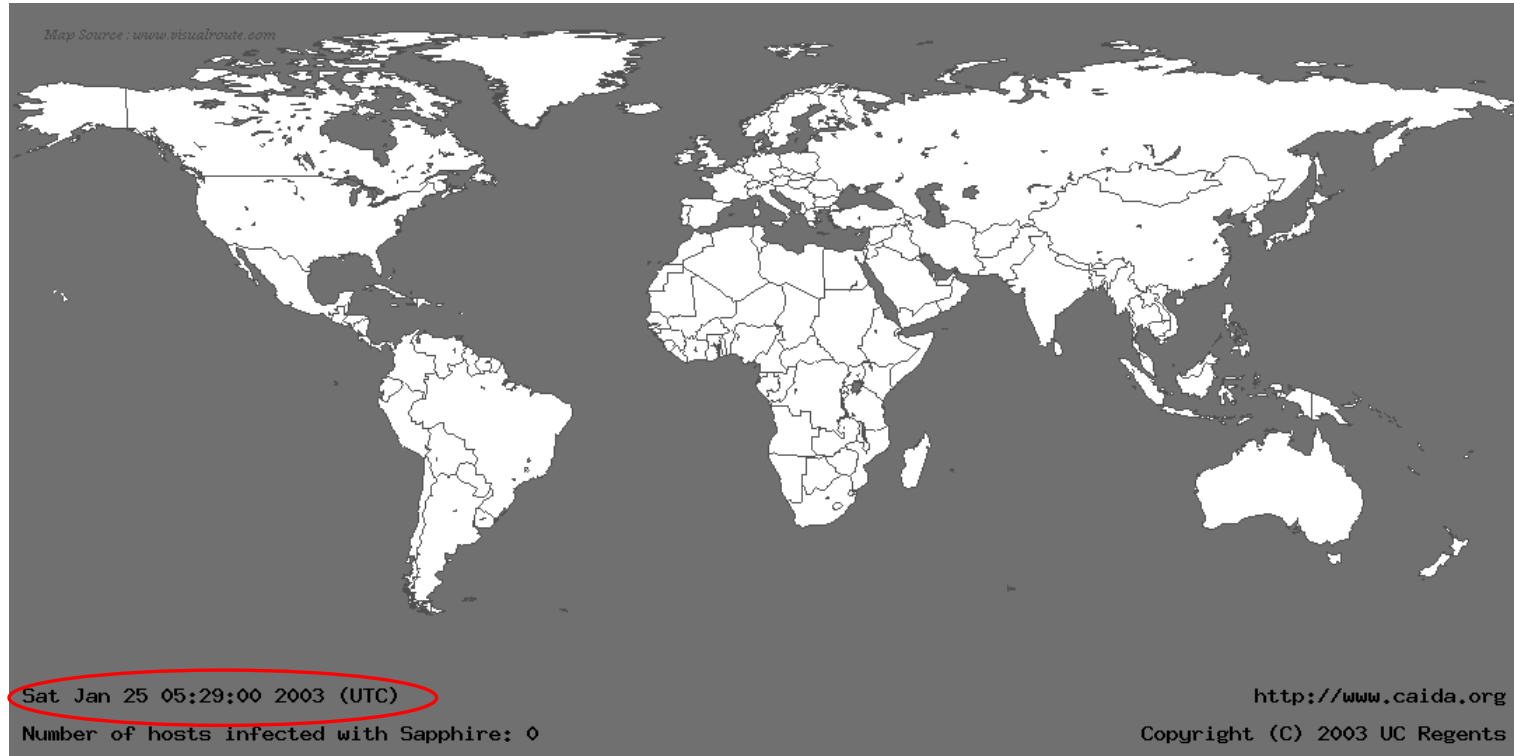
# Striving for Greater Virulence: Nimda

- Released September 18, 2001.
- Multi-mode spreading:
  - attack IIS servers via infected clients
  - email itself to address book as a virus
  - copy itself across network
  - modifying Web pages on infected servers w/ client exploit
  - **scanning for Code Red II backdoors**
- Worms form an ecosystem
- Leaped across firewalls

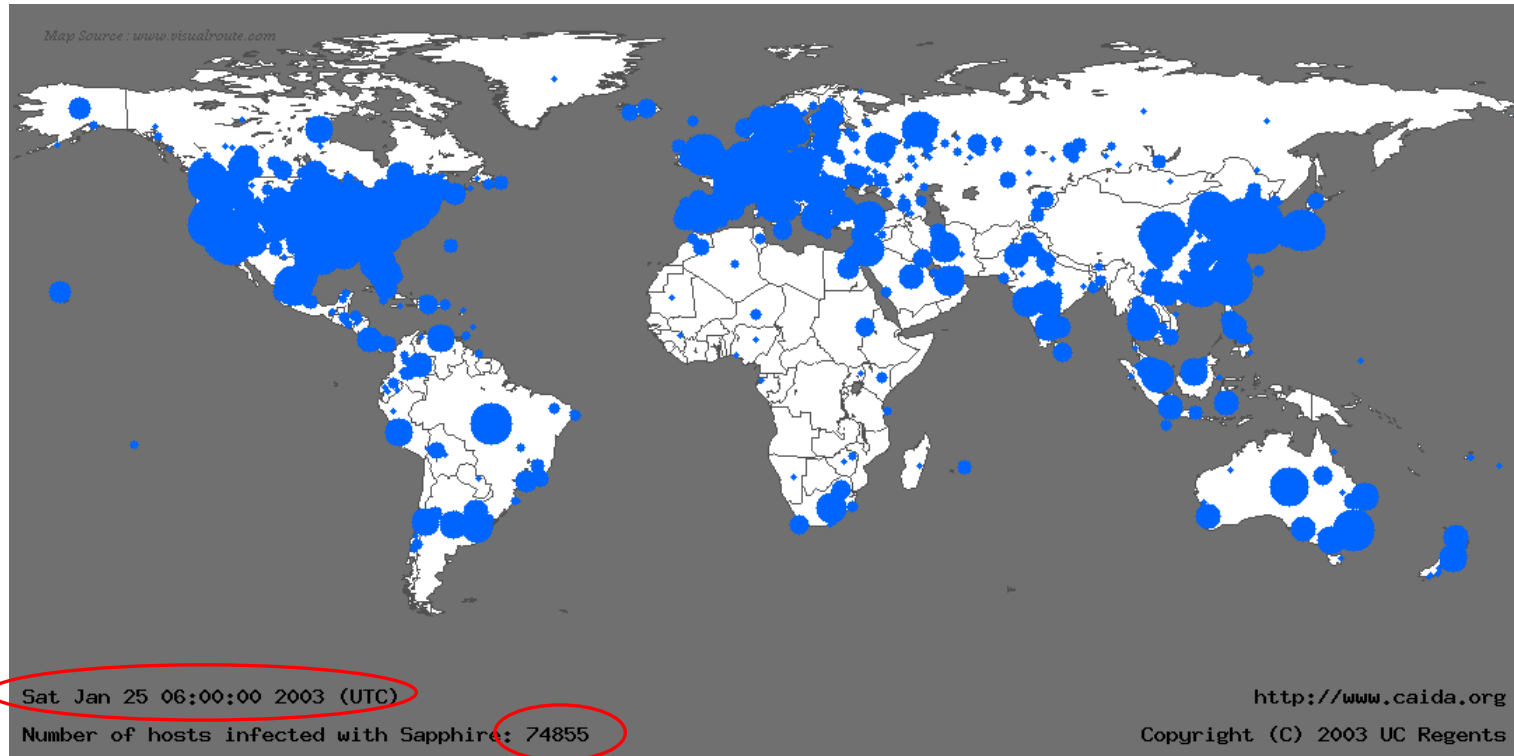
# SQL Slammer

- 01/25/2003
- Vulnerability disclosed : 25 June 2002
- Exploit overflow in MS SQL server
- Patch had been available for > 6 months
- Connectionless UDP rather than TCP
  - Entire worm fit in a single packet: 380 bytes
- Better scanning algorithm
  - worm could “fire and forget” - stateless
- Infected 75k machines in 10 minutes
  - At its peak, double every 8.5 seconds

# Life Just Before Slammer



# Life Just After Slammer





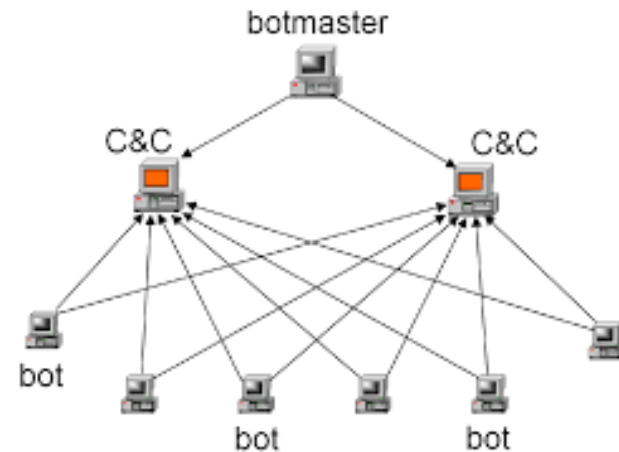
# Consequences

- ATM systems not available
- Phone network overloaded (no 911!)
- 5 DNS (Domain Name Service) root servers down
- Planes delayed

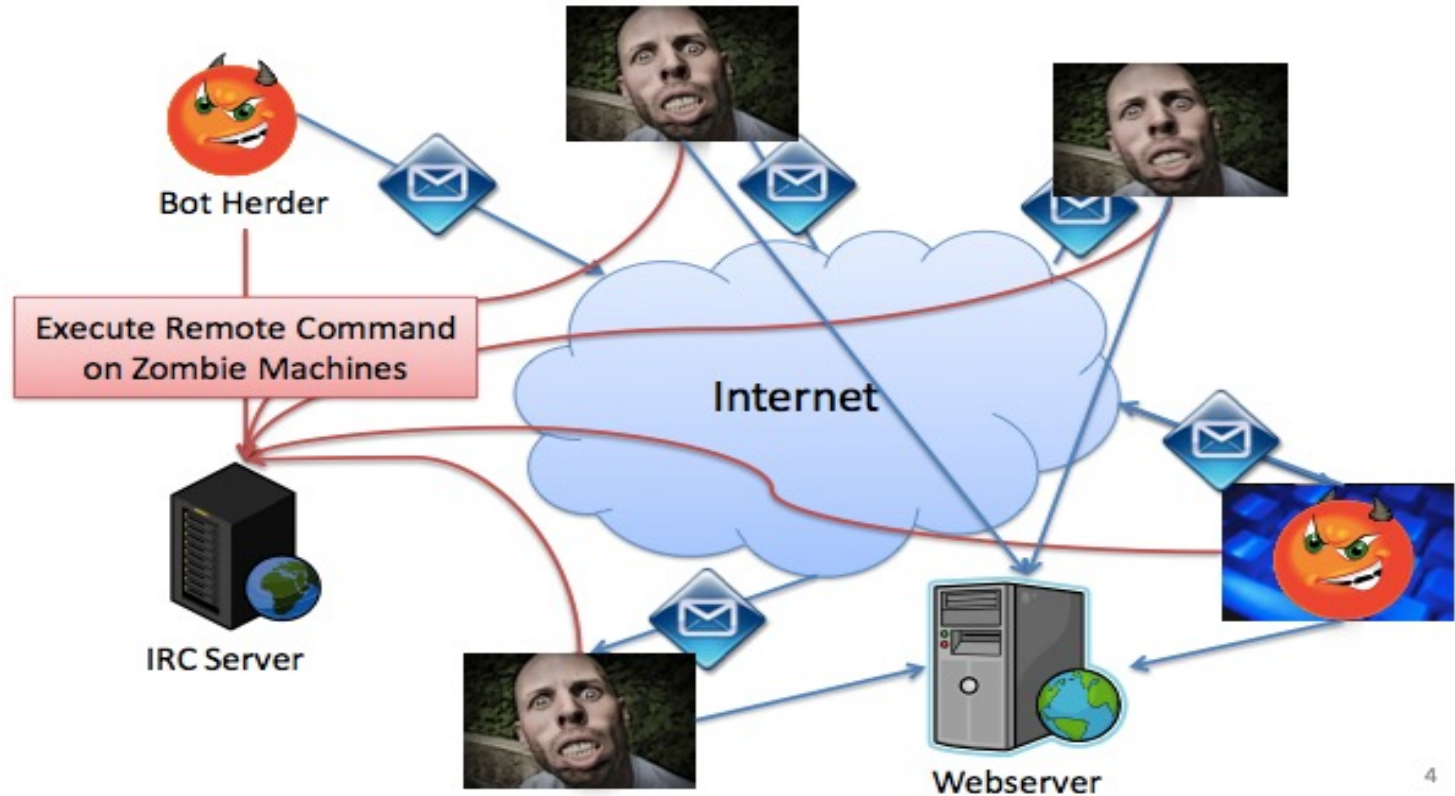
# Botnets

# Botnets

- **C&C (Command & Control)** channel between bots and botmaster
- Centralized botnet
  - Agobot
- P2P botnet
  - Storm
- Interesting facts
  - Twitter-based Botnet Command Channel



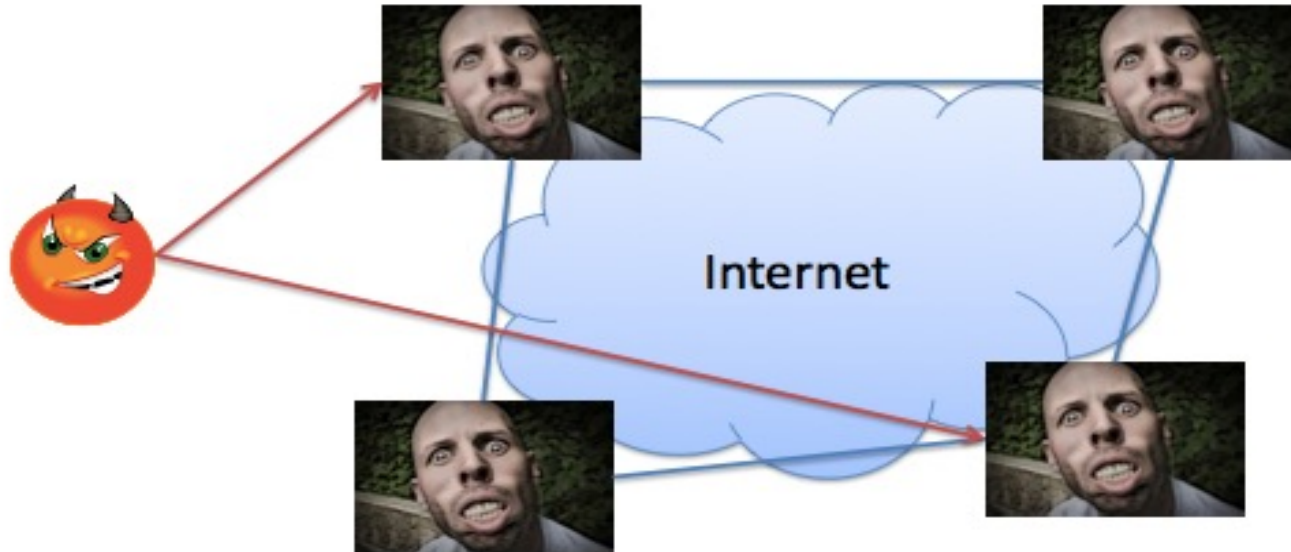
# Botnet: the old way – IRC or webserver based



# Agobot

- Public source code release: 2002
- IRC based command and control
- Features:
  - DoS attack library
  - Limited polymorphic obfuscations
  - Harvests PayPal passwords, AOL keys, etc.
  - Defends compromised systems
    - Killing anti-virus, testing for VMWare, altering anti-virus DNS entry
  - Anti-disassembly mechanisms
    - Testing for debugger presence

# P2P-based botnet



# Storm botnet

- Appeared in 2006, gained prominence in Jan 2007
- First major botnet to employ P2P command and control architecture
- Features
  - Recruits new bots using a variety of attack vectors
    - Email messages with exe
    - Email messages with link to infected sites
    - E-card spam
  - Use computing power of compromised machines
    - Sends and relays SPAM
    - Hosts the exploits and binaries
    - Conducts DDoS attacks
  - First to spam with embedded mp3 (non-malicious)

# Stuxnet: modern malware

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

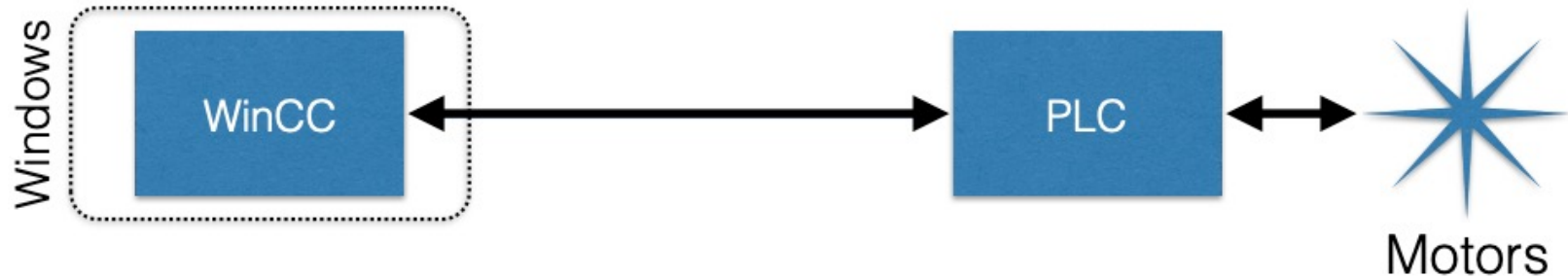


# Stuxnet

- Payload
  - Do nothing
  - Unless attached to particular models of frequency converter drives that operate at 807-1210Hz
  - In which case, slowly increase the frequency 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

- 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



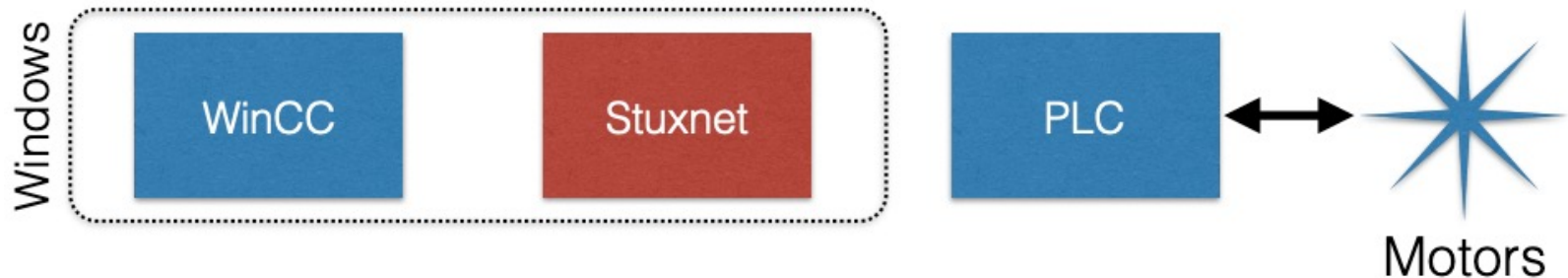
# Stuxnet

- 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



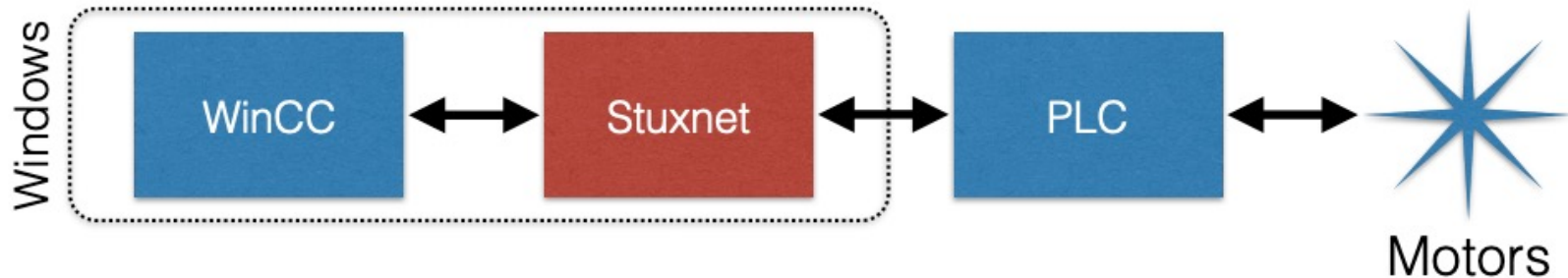
# Stuxnet

- 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



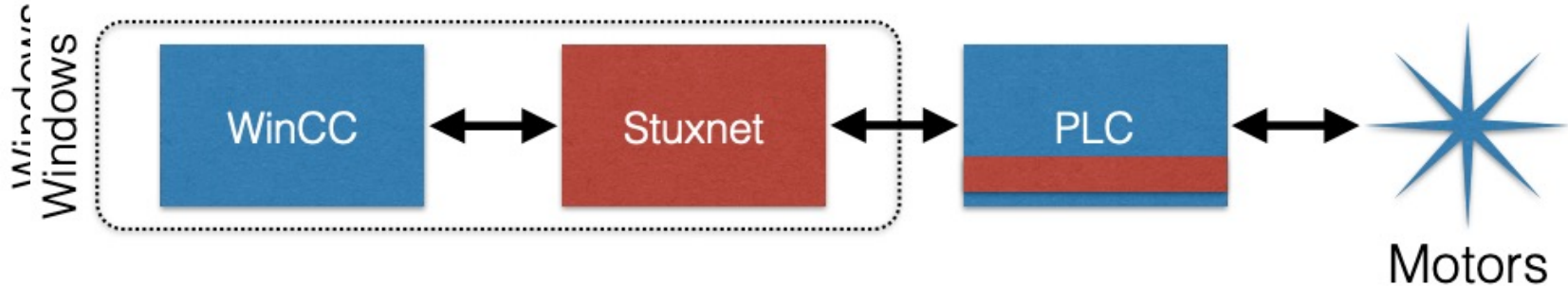
# Stuxnet

- 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



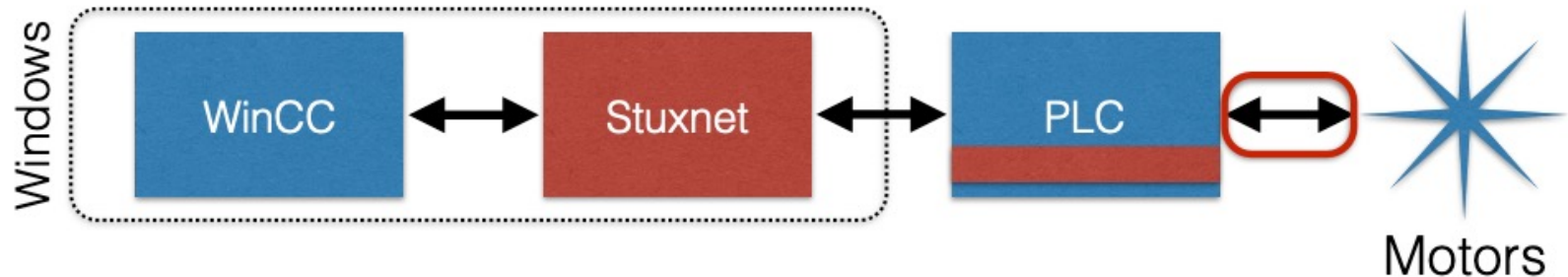
# Stuxnet

- 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



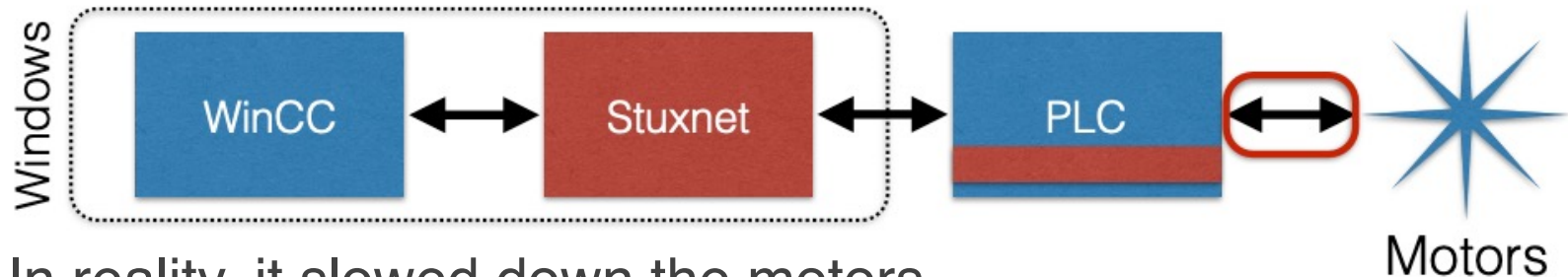
# Stuxnet

- 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



# Stuxnet

- 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 slowed down the motors
- Destroy or decrease the productivity



# 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 response