Malware
Part 1: Introduction

Part 2: Malware bestiary

Part 3: Malware Analysis
Part 1: Introduction

- What it is
- Why it is so common
- Who are the authors
- Very short history
- Current situation

Part 2: Malware bestiary

Part 3: Malware Analysis
Introduction

- Malicious Code (Malware)
  - Software that fulfills the malicious intent of the author
  - Intentionally written to cause adverse effects on the system (it is NOT a vulnerable program)
  - Comes in many flavors but all share a common characteristic: perform some unwanted activity on your system

- Term often used equivalent with virus (due to media coverage)
  - However, a virus is just a particular type of malware
  - Classic viruses account for only 3% of malware in the wild
Why

- The old “hacker” school
  - Fun
  - Show that something is possible
  - Prove the author leetness

- The virus golden age
  - Vandalism

- The new “criminal” age
  - Steal information
  - Control the machine
    - DDos
    - Spam
  - Make money
Guess Who?

>> Yesterday
Guess Who?

>> Yesterday

>> Today
The Rise of the Underground Economy

- Moving from “Hack for Fun” to “Hack for Profit”
  - Traditional fraud mechanism combined with technological knowledge led to a well-established underground economy

- Understanding the technical side is not enough
  - Monitor underground channels
  - Understanding the economic structure (players, payments...)
  - Find and disrupt weak links

- In this scenario, we do not need perfect solutions, we need solutions that make an attack vector economically not viable
Short History

- **1981** – First widespread outbreak of a virus
  - Propagates on floppy disks for the Apple II platform
- **1983** – The first documented experimental virus
  - Len Eidelmen coined the term *virus* in connection with self-replicating computer programs
  - Fred Cohen’s pioneering work on the topic
- **1987** – File infectors
  - “Christmas Tree” worm hit IBM Mainframes (500,000 replications/hour)
- **1987** – The Vienna virus is the first one to intentionally destroy data
- **1988** – First Worm: Internet worm
  - Internet worm created by Robert Morris
  - CERT is created
- **1989** – IBM releases Viruscan for MS-Dos
Short History

- 1991 – First polymorphic virus
- 1992 – First virus to look for (and damage) an anti-virus
- 1995 – First macro virus
- 1998 – First Java virus
- 1999 – Melissa
  - Word macro virus + first large scale email propagation
- 1999 – Distributed denial of service (DDOS)
- 1999 – Kernel Rootkits
  - Knark (modification of system call table)
Short History

- 2000 – Spyware:
- 2000 – First virus for mobile phones
- 2001 – Code Red
  - first large-scale, exploit-based worm
- 2003 – SQL Slammer worm
  - extremely fast propagation
- 2003 – Sobig makes the first attempt to create a botnet
- 2007 – Storm Worm+Botnet
- 2008 – Rogue Antivirus
- 2008 – Koobface targets Social Networks
- 2009 – Conficker is the first pandemic malware infection
  - Microsoft is still offering 250K $ to find the authors
Reasons for Malware Prevalence

- Mixing **data and code**
  - violates important design property of secure systems
  - unfortunately very frequent

- **Homogeneous computing base**
  - Windows is just a very tempting target

- **Unprecedented connectivity**
  - easy to attack from safety of home

- **Clueless user base**
  - many targets available

- **Malicious code has become profitable**
  - compromised computers can be sold and used to make money
Are we Losing the Game?

- The overall picture:
  - Malware industry is worth 100 billion dollars per year
  - 60%-90% of websites are vulnerable to some kind of attacks
  - 80% of the e-mail traffic out there is spam
  - 50%-80% of computers connected to Internet are infected with spyware

- The economical side:
  - A 26 year-old made 20 million $ with spam before being caught
  - Phishing sites hosted by the Russian business network made an estimated 150 million $ with stolen bank credentials in 2006
  - Top rogue anti-virus "affiliates" make over 300,000 $ a month
The Size of the Problem

![Graph showing the increase in the number of new malicious programs from 2003 to 2009. The graph indicates a significant rise in 2008 onwards.](image-url)
Part 1: Introduction

Part 2: Malware bestiary
   - Malware classification
   - The main flavors
   - Infection mechanisms
   - Propagation strategies
   - Defense

Part 3: Malware Analysis
<table>
<thead>
<tr>
<th></th>
<th>Self-Spreading</th>
<th>Non-Spreading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Spreading</td>
<td>• Computer Virus</td>
<td>• Worms</td>
</tr>
<tr>
<td>Non-Spreading</td>
<td>• Trojan Horse</td>
<td>• Rootkit</td>
</tr>
<tr>
<td>Need an host program</td>
<td>• Keylogger</td>
<td>• Spyware</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dialers</td>
</tr>
<tr>
<td>Self-contained program</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
One Malware – Many Names

- Malware writers would like to name themselves
  - this is of course not possible
  - Netsky --> Skynet discussion

- The first identifier of a new sample is the one that typically assign the name
  - however, people (and anti-virus companies) work in parallel…

- Each anti-virus company has its own notation
  - causes confusion and often multiple names
  - attempts to create a unique naming and identification notation
Example: Virus Total

<table>
<thead>
<tr>
<th>Antivirus</th>
<th>Version</th>
<th>Last Update</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>AhnLab-V3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AntiVir</td>
<td>-</td>
<td>-</td>
<td>TR/Proxy.Cimuz.1</td>
</tr>
<tr>
<td>Authentium</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Avast</td>
<td>-</td>
<td>-</td>
<td>Win32:Zhelatin-MI</td>
</tr>
<tr>
<td>AVG</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BitDefender</td>
<td>-</td>
<td>-</td>
<td>Trojan.Peed.LW</td>
</tr>
<tr>
<td>CAT-QuickHeal</td>
<td>-</td>
<td>-</td>
<td>(Suspicious) - DNAScan</td>
</tr>
<tr>
<td>ClamAV</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DrWeb</td>
<td>-</td>
<td>-</td>
<td>Trojan.Packed.75</td>
</tr>
<tr>
<td>eSafe</td>
<td>-</td>
<td>-</td>
<td>Suspicious Trojan/Worm</td>
</tr>
<tr>
<td>eTrust-Vet</td>
<td>-</td>
<td>-</td>
<td>Win32/Difisim.AG</td>
</tr>
<tr>
<td>Ewido</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F-Prot</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F-Secure</td>
<td>-</td>
<td>-</td>
<td>Email-Worm.Win32.Zhelatin.cl</td>
</tr>
</tbody>
</table>
Virus

- A program that can infect other programs by modifying them to include a, possibly evolved, version of itself
  
  -- Fred Cohen (1983)

- Distinguishing features:
  - Reproduce its own code
  - Attach itself to other files (a virus cannot “survive” by itself)
  - Get executed when the infected executable file is executed
Where is it?
Where is it?

File infection

- overwrite virus
  - substitute the original program
- parasitic virus
  - append virus code before or after the program
  - modify program entry point
- cavity virus
  - install inside of the file it is infecting
  - usually in empty gaps inside the program binary
Many modern applications support macro languages
- Microsoft Word, Excel, Outlook
- macro languages are powerful
- Embedded macros automatically executed on load
Where is it?

- **Kernel**
- **Hypervisor**
- **Hardware**

**Boot infection**
- master boot record (MBR) of hard disk (first sector on disk)
- boot sector of partitions
- rather old, but interest is growing again
Virus Defense

- Antivirus Software
  - Mainly signature based detection
    - database of byte-level or instruction-level signatures that match virus
    - wildcards and regular expression can be used
    - md5 of known malicious programs
  - Heuristics (check for signs of infection)
    - code execution starts in last section
    - incorrect header size in PE header
    - suspicious code section name
    - patched import address table
  - Behavioral signatures

- Sandboxing
  - run untrusted applications in restricted environment
Worms

- A computer worm is a self-replicating computer program that uses a network to send copies of itself to other nodes

Distinguishing features:

- Autonomous spread over network
- It is self-contained
  (it does not need to be embedded in a file like a virus)
- Speed of spreading is constantly increasing
  (can cause enormous damage to the network)
- Make use of techniques known by virus writers for long time
Worm Propagation

- Email harvesting
  - consult address books (W32/Melissa)
  - files might contain email addresses
    - inbox of email client (W32/Mydoom)
    - Internet Explorer cache and personal directories (W32/Sircam)
- Network share enumeration
  - Windows discovers local computers, which can be attacked
  - some worms attack everything, including network printers
- Scanning
  - randomly generate IP addresses and send probes
  - interestingly, many random number generators flawed
    - static seed
    - not complete coverage of address space
  - some worms use hit-list with known targets (shorten initial phase)
Worms Mail Propagation

- Often use social engineering techniques to get executed
  - fake from address
  - promise interesting pictures or applications
  - hide executable extension (.exe) behind harmless ones (.jpeg)
- Many attempt to hide from virus scanners
  - zipped, sometimes even with password (ask user to unpack)
- Some exploit Internet Explorer bugs when HTML content is rendered
- Speed of spread limited because humans are in the loop
  - can observe spread patterns that correspond to time-of-day
Exploit-based Propagation

- Require no human interaction
  - typically exploit well-known network services
  - can spread much faster

- Propagation speed limited either...
  - ...by network latency
    worm thread has to establish TCP connection (Code Red)
  - ...by bandwidth
    worm can send (UDP) packets as fast as possible (Slammer)

- Spread can be modeled using classic disease model
  - worm starts slow (only few machines infected)
  - enters phase of exponential growth
  - final phase where only few uncompromised machines left
Example: CodeRed (July 2001)

- Exploits known buffer overflow in `Idq.dll`
- Propagates through an HTTP request (TCP)
- Malicious code stays in memory
- Target: DDOS `www.whitehouse.gov`
- Vulnerable population (360,000 servers) infected in 14 hours
Example: Slammer  (January 2003)

- The worm
  - Exploits a vulnerability in Microsoft SQL Server disclosed 7 months before
  - Propagates through a single 380 byte UDP packet
  - In approximately three minutes, the worm achieved its full scanning rate (more than 55 million scans per second)
  - 75,000 victims infected within ten minutes

- Consequences
  - The worms packets saturated many Internet links
  - numerous routers collapsed under the burden of extremely high bombardment traffic
  - Problem in ATM and 911 systems
  - 2 to 5 of the 13 root DNS shut down
  - Cancelled airline flights
Detecting and Measuring Worms Propagation

- Observe and measure traffic targeting the dark (unused) address-space of the network
- All incoming traffic in a darknet is suspicious by definition
- Manifestation of different events
  - Attackers probing blindly
  - Random scanning from worms
  - DOS (backscattered effect due to IP spoofing)

- Example:
  - UCSD network telescope is a class 8 network (~ 1/256 of the Internet)
Worms Defense

- **Virus scanners**
  - effective against email-based worms

- **Host level defense**
  - mostly targeted at protecting the software from remote exploitation
  - stack protection techniques
  - address space layout randomization (ASLR)
  - attempts to achieve diversity to increase protection

- **Network level defense**
  - intrusion detection systems
    - fast and automatic signature generation is active research area
  - limit the number of outgoing connections
    - helps to contain worms that perform scanning
  - personal firewall
    - block outgoing SMTP connections (from unknown applications)
Trojan Horses

- Class of malware that appears to perform a desirable function but it also performs undisclosed malicious activities

- Distinguishing features:
  - Require the user to explicitly run the program
  - Unable to make copies of themselves or self-replicate
  - Can be used to perform any kind of malicious activity
Trojan Horses

- Two types of Trojan horses
  - Malicious functionality is included into useful program
    - games, disk utility, screensaver..
    - famous compiler that included backdoor into code
  - The malware is a stand-alone program
    - possibly disguised file name (sexy.jpg.exe)

- Many different types and malicious functionalities
  - spy on sensitive user data (spyware)
  - disguise presence (rootkit)
  - allow remote access (backdoor)
    - base for further attacks, mail relay (for spammers)
  - damage routines (corrupting files)
How did the Horse get Inside the Machine?

- Mainly through social engineering (aka, user stupidity)
- The user... open a suspicious mail attachment
How did the Horse get Inside the Machine?

- Mainly through social engineering (aka, user stupidity)
- The user... open a suspicious mail attachment
- The user... follow a link to an infected site
- The user... download and execute an unknown binary
How did the Horse get Inside the Machine?

- Mainly through social engineering (aka, user stupidity)
- The user... open a suspicious mail attachment
- The user... follow a link to an infected site
- The user... download and execute an unknown binary
- The user... download an infected file using a file sharing software
  - In Limewire, 68% of all downloadable executable, archival, and Microsoft Office file extensions contain malware
    (*Kalafut 06)
Spyware

- Software that secretly monitors the user's behavior and collects private information without the user knowledge

- Distinguishing features:
  - Threat to the user privacy
  - Hidden from the user and usually difficult to detect
  - Leak sensitive information (credit card numbers, visited webpages, passwords..) to someone else over the internet
  - Often implemented as a browser plugin/extension
Keyloggers

- Simple idea but dangerous threat

- Records key strokes typed by user
  - key logger runs as process in the background

- Typically used to capture interesting user data
  - passwords, information for social engineering attacks (emails)
  - successful independent of any application-level protection or network-level protection (e.g., by using encrypted traffic)

- Key loggers are quite easy to write
  - easy to re- implement and therefore difficult to recognize by antivirus scanners because not encountered in the wild before
**Keyloggers**

- **Message handling mechanism**
  - Windows NT kernel processes key press interrupt then sends message to active application

### SetWindowsHookEx (Windows API function)

> Installs an application-defined hook procedure into a hook chain. You would install a hook procedure to monitor the system for certain types of events. These events are associated either with a specific thread or with all threads in the same desktop as the calling thread.

### GetAsyncKeyState (Windows API function)

> Determines whether a key is up or down at the time the function is called, and whether the key was pressed after a previous call to GetAsyncKeyState.
Spyware Infection

- Bundled with legitimate software package
  - end-user license agreement (EULA) even informs about this fact
    - EULA is very long (often hundreds of pages), user accepts
    - classic examples are shareware programs

- Drive-by downloads
  - Javascript code contained in malicious webpage launches a number of exploit against the browser or its components (typically ActiveX)
    - WMF (Windows meta file) exploit, around Christmas 2005
    - insufficient ActiveX security settings

- Fake dialogs
  - display “Would you like to optimize your Internet” and perform installation when user agrees
Rootkits

- The goal of a rootkit is to disguise the existence of a malicious program in a system

- Distinguishing features:
  - Installed by an attacker after a system has been compromised
  - Hide the presence of the attacker and allows him to return on a later time
  - Difficult to detect
Rootkits Types

- **Userspace rootkits**
  - Replace system programs with trojaned versions
    - system browsing (`ls`, `find`)
    - system logging (`syslogd`)
    - system monitoring (`ps`, `top`, `netstat`)
    - user authentication (`login`, `sshd`)

- **Kernel rootkits**
  - Modify the kernel data structures to hide processes, files, and network activities
  - Installed by
    - Kernel patch
    - Loadable Kernel Module
    - Kernel memory patching (`/dev/kmem`)
Where is it?

- Userspace
- Kernel
- Hypervisor
- Hardware

Trojaned programs
Modified Kernel

Hypervisor Rootkits
- Shift the entire operating system under a virtualized environment
- Since it runs below the operating system, it may be **extremely** difficult to detect
The Sony Rootkit

- “Copy protection” measures included by Sony on compact discs in 2005
- Installed a rootkit that filters out any files/directories, processes and registry keys that contain $sys$
  - Very easy to reuse for other malware writers to hide their files by simply using the same name
- System call dispatcher
  - uses system service dispatch table (SSDT)
  - Windows NT kernel equivalent to system call table
- Entries can be manipulated to re-route call to custom function
  - ZwCreateFile - create or open file
  - ZwQueryDirectoryFile - list directory contents (list subdirectories and files)
  - ZwQuerySystemInformation - get the list of running processes (among other things)
  - ZwEnumerateKey - list the registry keys below a given key
Rootkit Defense

- **Userspace**
  - Tripwire (user-space integrity checker)
  - Chkrootkit (user-space, signature-based detector)

- **Kernelspace**
  - KSTAT (Kernel Security Therapy Anti-Trolls)
    Linux tool that uses several methods to check the integrity of the running kernel
    
    ```
    sys_read 0xc2846868  Warning! Should be at 0x12699c
    ```

- **Limitations**
  - Kernel rootkits have complete control over operating system
  - All applications can be arbitrarily fooled (including rootkit detectors)
  - At best, an arms race can be started

- **Hypervisor-based detectors (?)**
Bots and Botnets

- A bot (aka zombie or drone) is a compromised machine that can be controlled by an attacker remotely

  * Originally automated programs used to provide useful services on Internet Relay Chat (IRC) channels

- Distinguishing features:
  - Remote control facility
  - Implementation of different commands
  - Infected machines are incorporated in large networks (botnets) that are controlled by the botmaster like an army
  - Loaded on a computer after compromise (eg. by a trojan or a worm)
Botnets

- The attacker control a Command and Control (C&C) server
- All the bots contact the C&C to:
  - report their status
  - receive commands
Botnets (example of available commands)

- Harvest email addresses from host
- Log all keypresses
- Sniff network traffic
- Take screenshots
- Start an http server that allows to browse C:
- Kill a hard-coded list of processes (AV programs, rival malware)
- Steal windows CD keys
- Sets up a proxy to be used as a "stepping stone" for SPAM
- Download file at an url
- Run a shell command
- Update (allows to change the available commands!)
C&C Mechanisms

- The most distinguishing and powerful feature of Botnets...
  - As long as there is an update command defined for the bots the botmaster can change the command set by updating her bots
  - C&C brings a great flexibility to the activities that can be performed by bots

- However, C&C is also the weakest link of the system
  - shutdown C&C, and all bots become “harmless”
- Real botnets don't have a single C&C server
  - even trinoo (1999) allowed multiple masters!
Centralized C&C

- Two approach to issue commands
  - Push style C&C (e.g. IRC)
  - Pull style C&C (e.g. HTTP)
- Multiple C&C servers (typically http/IRC)
  - multiple layers of hosts between bot and botmaster (hard to trace back)
- Address of C&C server(s) must be available to each bot
  - in binary, config file, etc
  - frequently updated
- Large botnets often partitioned into several smaller ones
  - each bot knows only a few C&C servers
  - you do not know everything about a botnet just by looking at 1 sample
Push vs. Pull

Push-style C&C

Pull-style C&C
Common C&C Channels

- **IRC**
  - used because it is directly suitable to the purpose of sending commands to bots
    - legitimate IRC server can be used, acts as stepping stone
  - also for historic reasons
    - sdbot/agobot/phatbot source-code is available
  - quite unusual on modern networks! (suspicious/blocked...)

- **HTTP**
  - outgoing HTTP allowed everywhere (sometimes through proxy)
  - a lot of complex applications are delivered on top of HTTP
  - hard to model "normal" HTTP traffic
    - difficult to detect "unusual" botnet traffic
Peer-to-Peer C&C

- Alternative to client-server architecture
  - bot commands propagate in a p2p network
- More robust
  - more difficult to catch the botmaster
  - even if some of the nodes in the network are shut down, the gaps in the network are closed and the network continues its activities

- Custom peer-to-peer protocol
  - The new, unknown traffic can be “noisy”
  - A bot needs a way to find new neighbors
- Standard peer-to-peer protocol
  - allows C&C traffic to blend in with legitimate p2p traffic
  - legitimate p2p is still blocked in many locations
Bullet-Proof Hosting

- Sometimes C&C is hosted by ISPs that are completely unresponsive to abuse complaints
  - In these cases, a solution can be to take down the entire ISP
- De-peering: other ISPs rescind their peering agreements with the malicious ISPs
  - evidence is required that abuse breached clauses of peering contracts
  - may involve law enforcement, FTC, etc
- Examples:
Fast-Flux

- Round Robin DNS
  - A DNS answer consists of several DNS A records.
  - Each time the order of the answer list is different
  - The idea behind this is to balance the workload on different servers

- TTL value
  - Normally, the DNS records include a TTL value between 1 and 5 days for taking benefit from the DNS caching system
  - Recently, content delivery networks sets the TTL value to very low values, i.e. between 0 and 900 seconds

- Fast-Flux
  - Several IP addresses in the DNS record
  - Low TTL value
  - Usage of round robin DNS
Mebroot is a rootkit that takes control of a machine by replacing the system’s Master Boot Record (MBR)

- This allows Mebroot to be executed at boot time, before the operating system is loaded, and to remain undetected by most anti-virus tools

Mebroot has no malicious capability per se. Instead, it provides a generic platform to manage (install, uninstall, and activate) other malicious modules

- The Torpig botnet is an example of such malware
Hijacking the Torpig Botnet

- Torpig use a technique for locating its C&C servers called domain flux
  - each bot uses an algorithm to compute a list of domain names
  - this list is computed independently by each bot and is regenerated periodically (every week)
  - each bot attempts to contact the hosts in the list in order until one succeeds
- Researchers at UCSB reverse engineered the domain flux algorithm
  - They predict the C&C domain that was going to be used some time in the future
  - By registering that domain, they took control of the botnet for 10 days
- Data collected
  - 70GB from 52,540 different infected machines
  - 297,962 unique username/password (of which 8,310 for bank accounts)
  - 1,235,122 windows password
  - 1,258,862 email accounts
  - ....
Bots Defense

- Attack command and control infrastructure
  - take IRC channel off-line
  - when dynamic DNS is used for central command server, route traffic to black hole

- Honeypots
  - vulnerable computer that serves no purpose other than to attract attackers and study their behavior in controlled environments
  - when honeypot is compromised, bot logs into botnet
  - allows defender to study actions of botnet owners
The Malware Share (Panda 09)

- Trojan: 74%
- Worm: 13%
- Other: 9%
- Adware: 3%
- Spyware: 1%
Malware targeting Critical Infrastructures
Malware that (probably) targeted Iranian Nuclear centrifuges

Spreads by
  - Copying itself to USB drives
  - Network shares
  - Using two known and four 0-day Microsoft vulnerabilities

Windows rootkit to hide Windows binaries
  - Signed by one of 2 stolen certificates from ‘JMicron’ and ‘Realtek’
Stuxnet

- Injects STL code into Siemens PLCs (Programmable Logic Controllers)
  - Uses rootkit techniques to hide injected PLC code
  - Patches Siemens Step 7 software (used to view PLC code)
- Communicates with C&C servers using HTTP
  - www.mypremierfutbol.com
  - www.todaysfutbol.com
- Steals designs documents for industrial control systems
- Sabotages targeted industrial control systems
Part 1: Introduction

Part 2: Malware bestiary

Part 3: Malware Analysis
- Advanced malware techniques
- Static & Dynamic analysis
- Large scale malware analysis
- Anubis
- Malware statistics
Two Problems

Detecting malware samples
- It is not possible to build a perfect virus/malware detector (Cohen)
- Current anti-viruses are mostly black-list based
  - The number of malware samples is growing so fast that it (probably) already surpassed the number of good software (goodware)
  - Malware writers try to change the binary at every infection to defeat signatures

Identifying new malware samples
- Too many new samples to analyze (→ automated analysis)
- Many samples are variations of the same malware (→ clustering)
- Static analysis is insufficient (→ dynamic analysis)
- Dynamic analysis is problematic (→ ??)
Malware Hardening

- Entry Point Obfuscation
  - virus scanners quickly discovered to search around entry point
  - virus hijacks control later (after program is launched)
    - overwrite import table addresses
    - overwrite function call instructions

- Code Integration
  - merge malware code with target program
  - requires disassembly of target (difficult task on x86 machines)
  - W95/Zmist is an example for this technique
    - Decompile the target code
    - Insert itself
    - Re-generate code and data reference
    - Assemble the final code
Poli- and Meta-morphism

- **Polymorphic malware**
  - Change layout (shape) with each infection
  - but maintain the algorithm intact
  - The payload is encrypted using a different key for each infection
  - A decryption routine decrypt and execute at runtime the payload
    - of course, the decryption routine must be changed as well
  - Makes static string analysis practically impossible

- **Metamorphic malware**
  - Create different “versions” of code at each infection
  - The new code is different but equivalent to the previous one (i.e., it has the same semantic)
Chernobyl (CIH) Virus

```
5B 00 00 00 00  pop ebx
8D 4B 42     lea ecx, [ebx + 42h]
51            push ecx
50            push eax
50            push eax
0F 01 4C 24  FE sidt [esp - 02h]
5B pop ebx
83 C3 1C     add ebx, 1Ch
FA            cli
8B 2B         mov ebp, [ebx]
```
Dead Code Insertion

5B 00 00 00 00 pop ebx
8D 4B 42 lea ecx, [ebx + 42h]
51 push ecx
50 push eax
90 nop
50 push eax
40 inc eax
0F 01 4C 24 FE sidt [esp - 02h]
48 dec eax
5B pop ebx
83 C3 1C add ebx, 1Ch
FA cli
8B 2B mov ebp, [ebx]

5B 00 00 00 00 8D 4B 42 51 50 90 50 40 0F 01 4C 24 FE 48 5B 83 C3 1C FA 8B 2B
Instruction Reordering

| 5B 00 00 00 00 | pop ebx |
| EB 09          | jmp <S1> |

**S2:**
- 50 push eax
- 0F 01 4C 24 FE sidt [esp - 02h]
- 5B pop ebx
- EB 07 jmp <S3>

**S1:**
- 8D 4B 42 lea ecx, [ebx + 42h]
- 51 push ecx
- 50 push eax
- EB F0 jmp <S2>

**S3:**
- 83 C3 1C add ebx, 1Ch
- FA cli
- 8B 2B mov ebp, [ebx]
Instruction Substitution

```
5B 00 00 00 00    pop ebx
8D 4B 42 42       lea ecx, [ebx + 42h]
51                 push ecx
89 04 24 24       mov eax, [esp]  // push eax
83 C4 04          add 04h, esp
50                 push eax
0F 01 4C 24 FE    sidt [esp - 02h]
83 04 24 0C       add 1Ch, [esp]  // pop ebx
5B                 pop ebx  // add ebx, 1Ch
8B 2B              mov ebp, [ebx]
```

```
5B 00 00 00 00 8D 4B 42 51 89 04 24 83 C4 04 50 0F 01 4C 24 FE 83 04 24 0C 5B 8B 2B
```
Packers

- Originally, *runtime packers* were developed to compress the executable of an application in order to reduce the binary size
  - A packer takes an executable as input and generate a “packeted” version of the executable that include the runtime unpacking routine
- They soon started to be adopted to protect the application from reverse engineering
  - Features that malware writers found very useful
- Recent packers functionalities
  - Compress
  - Encrypt
  - Randomize (polymorphism)
  - Anti-debug technique (int / fake jmp)
  - Anti-VM
  - Virtualization (e.g., Themida)
Packers

- In some cases, the reverse transformation is possible
  - Unpacking tools exist for some packers

- In some cases, the reverse transformation is very hard or almost impossible
  - The original program is translated into a different randomly generated language
  - The executable is shipped with a metamorphic emulator that executes the new language

- The size of the final executable is sometimes much bigger than the original
  (helloworld.exe packed with Themida is 0.5 Megabyte)

- Packers make static analysis impossible, and dynamic analysis more complicated
The Anti-virus Reaction

- Simple evasion technique are very effective against static analysis
- Thus, dynamic analysis techniques introduced
  - virus scanner equipped with emulation engine
  - executes actual instructions (no disassembly problems)
  - runs until polymorphic part unpacks actual virus
  - then, signature matching can be applied
  - emulation must be fast

- Difficulties
  - virus can attempt to detect emulation engine
  - time execution, use exotic (unsupported) instructions, …
  - insert useless instructions in the beginning of code to deceive scanner
Behavior-Based Signatures

- Are there differences between malicious and benign behavior?
- Is it possible to exploit those differences to realize behavior-based signatures that can match bot known and unknown malware?
- Many antivirus products claim to already adopt this approach
- Still an open question....
Malware Analysis

- Understanding functionality of malware programs
  - modifications to compromised system
  - understand questions such as:
    - how is program launched, what malicious actions are performed, hidden functionality (with trigger), disabling of defense mechanisms, interaction with other processes …

- Necessary both for detection and removal
- Must keep up with increasing numbers of samples
  - fast
  - automated (at least, provide as much support as possible)
  - precise
Malware Analysis

- **Static techniques**
  - analyze binary for strings, symbols, and library functions
  - disassemble binary image

- **Dynamic techniques**
  - observe interaction with environment
    - file system, network, registry
  - observe interaction with operating system
    - system calls
  - debug process
Static Analysis: Disassembler

- Linear sweep disassembler
  - start at beginning of code (.text) section
  - disassemble one instruction after the other
  - assume that well-behaved compiler tightly packs instructions
  - `objdump` uses this approach

- Obfuscation Attack
  - insert data (or junk) between instructions and let control flow jump over this garbage
  - disassembler gets confused
Static Analysis: Disassembler

- Recursive traversal disassembler
  - aware of control flow
  - start at program entry point
  - disassemble one instruction after the other, until branch or jump is found
  - recursively follow both (or single) branch (or jump) targets
  - not all code regions can be reached
    - indirect calls and indirect jumps
      - use a register to calculate target during run-time
  - for these regions, linear sweep is used
  - IDA Pro uses this approach

- Obfuscation Attack
  - replace direct jumps (calls) by indirect ones
  - force disassembler to revert to linear sweep, and then use previous attack
The Static (and Manual) Approach

```
push ecx    ; x
call ds:connect
test eax, eax
jz short loc_4016A4

Sleep and loop back
```

```
loc_4016A4:   ; size_t
push 44h
push 0       ; int
lea eax, [ebp+StartupInfo]
push eax      ; void *
call _memset
add esp, 0Ch
mov [ebp+StartupInfo.cb], 44h
mov [ebp+StartupInfo.dwFlags], 10h
mov [ebp+StartupInfo.wShowWindow], 0
mov ecx, [ebp+0]
mov [ebp+StartupInfo.hStdError], ecx
mov edx, [ebp+0]
mov [ebp+StartupInfo.hStdOutput], edx
mov eax, [ebp+1]
mov [ebp+StartupInfo.hStdInput], eax
lea ecx, [ebp+ProcessInformation]
push ecx      ; lpProcessInformation
lea edx, [ebp+StartupInfo]
push edx      ; lpStartupInfo
push 0        ; lpCurrentDirectory
push 0        ; lpEnvironment
push 0        ; dwCreationFlags
push 1        ; bInheritHandles
push 0        ; lpThreadAttributes
push 0        ; lpProcessAttributes
push offset CommandLine ; "cmd.exe"
push 0        ; lpApplicationName
call ds:CreateProcess
push eax, eax
jnz short loc_401678
```
Limit of Static Analysis

- In a perfect world, static analysis could tell you everything about a program.
- But in real life, static analysis gives an approximate picture at best.
  - Impossible to predict the runtime behavior of all but the simplest programs.
  - Polymorphic, Encrypted, and Packeted samples are very very hard to analyze.
- It is still a powerful tools in the hand of an expert analyst.
- But it is very difficult to automatize and apply in a large scale.
Dynamic Analysis

- Execute code in a controlled environment
  - instrument program, operating system, or hardware
- Advantages
  - sees instructions that are actually executed
  - can inspect actual program behavior and data values
  - (at least one) target of indirect jumps (or calls) can be observed
- Problems of dynamic analysis
  - in general, single path (execution trace) is examined
  - analysis environment possibly not invisible
  - analysis environment possibly not comprehensive
  - may launch real attacks
  - scalability issues
Setting Up an Analysis Environment

- Running malware on your everyday machine is not a good idea
- **Real hardware vs Virtual Machine**
  - Virtual machine are detectable
  - Real machine are less scalable, more difficult to restore, and more difficult to instrument
- **Real Internet connection vs Simulated Network**
  - Containing malicious activities (spam, DOS, attacks, scans)
  - Internet connection may be required to observe the real behavior (bots)
- The analysis environment must be restored to a pristine state after each analysis
  - Norton Ghost (or similar tools) to restore the disk
  - CoreProtect write cache card
  - Virtual machine snapshots
Large-Scale Malware Analysis

- Refers to the problems that we need to deal with when dynamic malware analysis is used to examine
  1) Large numbers of malicious files
  2) Over longer periods of time
  3) Adversarial samples (i.e., evasion)

- Large-scale analysis is needed by…
  - Anti-malware companies and institutions
  - Security organizations (e.g., government)
  - Large ISPs
  - Automated honeypots
Anubis

- A platform for dynamic malware analysis
  - Runs the binary in a QEMU emulated environment
  - Monitors Windows API and system service calls
  - Records network traffic
  - Tracks data flows
- Anubis receives samples through
  - A public web-interface ([http://anubis.iseclab.org](http://anubis.iseclab.org))
  - A number of feeds from security organizations
- Some of our users are…
  - Shadow Server, Team Cymru, CERT Australia, law-enforcement agencies, ISPs, banks, anti-virus companies
- 20K-30K new samples analyzed per day
Sample Upload via HTTP

Sample Store

Report Store

Web / DB SRV

Worker VMs

Victim SRV

INTERNET

Firewall

ANUBIS ANALYSIS SYSTEM
<table>
<thead>
<tr>
<th>Description</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Autostart capabilities</strong>: This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.</td>
<td>![Red Alert]</td>
</tr>
<tr>
<td><strong>Changes security settings of Internet Explorer</strong>: This system alteration could seriously affect safety surfing the World Wide Web.</td>
<td>![Red Alert]</td>
</tr>
<tr>
<td><strong>Creates files in the Windows system directory</strong>: Malware often keeps copies of itself in the Windows directory to stay undetected by users.</td>
<td>![Red Alert]</td>
</tr>
<tr>
<td><strong>Sends Emails</strong>: This program sends out e-mails to other people possibly in order to propagate itself.</td>
<td>![Red Alert]</td>
</tr>
<tr>
<td><strong>Performs File Modification and Destruction</strong>: The executable modifies and destroys files which are not temporary.</td>
<td>![Red Alert]</td>
</tr>
<tr>
<td><strong>Performs Registry Activities</strong>: The executable reads and modifies registry values. It may also create and monitor registry keys.</td>
<td>![Red Alert]</td>
</tr>
</tbody>
</table>
**Autostart capabilities:** This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.

**Changes security settings of Internet Explorer:** This system alteration could seriously affect safety surfing the World Wide Web.

**Creates files in the Windows system directory:** Malware often keeps copies of itself in the Windows directory to stay undetected by users.

**Sends Emails:** This program sends out e-mails to other people possibly in order to propagate itself.

**Performs File Modification and Destruction:** The executable modifies and destructs files which are not temporary.

**Performs Registry Activities:** The executable reads and modifies registry values. It may also create and monitor registry keys.

### 1.a) - Network Activity

#### SMTP Conversations:

**from ANUBIS:1379 to 213.165.64.100:25**

- **Sender Address:** smith@gmx.de to **Recipient:** sam@gmx.de
- **Subject:** Love is...
- **Email Content:** Reply
- **Attached File:** "photo.zip" (application/octet-stream)

#### Unknown UDP Traffic:

#### Unknown TCP Traffic:

**from ANUBIS:1450 to 65.55.12.249:80**

- **State:** Connection established, not terminated
- **Transferred outbound Bytes:** 176 - **Transferred inbound Bytes:** 513

**Data sent:**

```
4745 5420 2f20 4854 5450 2f31 2e31 0d0a CET / HTTP/1.1..
4163 6365 7074 3a20 2a2f 2a2d 0a41 6363 3a20 6774 6875 6c6c 2e69 6e20 312e31 312e31 0d0a Accept: */*
5970 2c20 6465 6665 6c6c 696e 202f 7374 6865 7265 0a0d
```

```
Summary:

**Description**

- **Autostart capabilities**: This executable registers processes to be executed at system start. This could result in unwanted actions to be performed automatically.
- **Changes security settings of Internet Explorer**: This system alteration could seriously affect safety surfing the World Wide Web.
- **Creates files in the Windows system directory**: Malware often keeps copies of itself in the Windows directory to stay undetected by users.
- **Sends Emails**: This program sends out e-mails to other people possibly in order to propagate itself.
- **Performs File Modification and Destruction**: The executable modifies and destructs files which are not temporary.
- **Performs Registry Activities**: The executable reads and modifies registry values. It may also create and monitor registry keys.

### 1.a) - Network Activity

#### - SMTP Conversations:

*from ANUBIS:1379 to 213.165.64.100:25*

**Sender Address**: smith@gmx.de to **Recipient**: sam@gmx.de

**Subject**: Love is...

**Email Content**: Reply

**Attached File**: "photo.zip" (application/octet-stream)

### + Unknown UDP Traffic:

#### - Unknown TCP Traffic:

*from ANUBIS:1450 to 65.55.12.249:80*

**State**: Connection established, not terminated - **Transferred outbound Bytes**: 175 - **Transferred inbound Bytes**: 513

**Data sent**:

```
4745 5420 2f20 4854 5450 2f31 2e31 0d0a
4163 6365 7074 3a20 2a2f 2e6d 0a41 6363 6570 746f 6261 7273 696e 6720 3a20 226f 7267 696e 6722 2073 7465 7073 2069 6e69 6e67 6f6e 666f 7269 6461 7465 2072 6162 696e 6b20 7468 7265 616e 6765 2074 6f20 7374 6570 7369 6e74 2065 7665 7279 2074 6f20 696e 666f 7269 6461 7465 2066 6f72 206f 6e65 6c6c 6f77 6f72 616e 6420 7072 6f66 6573 7369 6e74 2069 6e74 6572 732e
```

### - Registry Values Modified:

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKLM\SYSTEM\CURRENTCONTROLSET\HARDWARE</td>
<td>ProxyEnable</td>
</tr>
<tr>
<td>PROFILES\CURRENT\Software\Microsoft\</td>
<td></td>
</tr>
<tr>
<td>windows\Current\Version\Internet</td>
<td></td>
</tr>
<tr>
<td>Settings 8</td>
<td></td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Windows\Current</td>
<td>Common AppData</td>
</tr>
<tr>
<td>Version\Explorer\Shell Folders</td>
<td></td>
</tr>
<tr>
<td>HKLM\Software\Microsoft\Windows\Current</td>
<td>jvzq</td>
</tr>
<tr>
<td>Version\Run 8</td>
<td></td>
</tr>
</tbody>
</table>
Malware Clustering

Partition a given set of malware samples into subsets so that subsets share some common traits (i.e., find “virus families”)

- Extremely useful to group together polymorphic samples
- Help analysts understanding the behavior of new samples
Malware Clustering

- Based on the concept of behavioral profile
  - abstraction of the program's execution trace that accurately captures the behavior of the binary
  - system calls can vary significantly, even between programs that exhibit the same behavior
- Approximate Clusters
  - Most clustering algorithms require to compute the distances between all pairs of points → \( O(N^2) \)
  - We use LSH (locality sensitive hashing), a technique to compute an approximate clustering that requires less than \( N^2 \) distance computations
- Promising results
  - Clustered 75,692 malware samples in 2h 18min
    (previous works clustered 500 samples in ~900h)
A View from Above

- Shift the focus from a single malware sample to a global picture of all the collected samples
- Extract concrete statistics on specific malware features
- Objectives
  - Shed light on common malicious host behavior
  - Confirm or deny folk wisdom
  - Investigate the influence of code polymorphism on statistics
A Glimpse into Common Malware Behaviors

**Filesystem**
- 37.2% of all samples create an executable file
  - 62% under the `C:\Windows` directory
  - 38% inside the user's home folder (under `C:\Document and Settings`)
    
    *Might indicate that malware is increasingly designed to run as a normal user!!*
  - Only 0.27% attempt to clear any records of its activity from log data

**Registry**
- 35.8% of all samples modify the registry to get launched at startup
- 33.7% disable the windows firewall
- 8.97% tamper with security settings (`MSWindows\Security`)

**Network**
- Only 47.3% of samples that show network activity also query the DNS to resolve a domain name (9.2% of the cases, no result was returned)
- 8.9% of the samples connect to a remote site to download a second-stage executable
The Importance of Clustering

- Comparing samples instead of families can skew the statistics and lead to potentially wrong interpretations.

- Malware Clustering:
  execution traces → behavioral profiles → malware families

*Network Protocols (by samples)*  
*Network Protocols (by clusters)*
The Importance of Clustering

- Comparing samples instead of families can skew the statistics and lead to potentially wrong interpretations.

- Malware Clustering:
  
  execution traces → behavioral profiles → malware families

  Botnets (by samples)  
  Botnets (by clusters)
Sandbox Detection

- How many samples hide their behavior by detecting Anubis and/or the underlying emulation system?

- Sandbox detection techniques:
  - Instruction-level: rely solely on CPU instructions
  - API-level: query the environment by calling (native or Windows) API functions

- API-level
  - 0.03% contain a known Anubis check
  - 0.51% are protected with a packer that is known to be not correctly emulated in Qemu

- So far, we did not have any way to find “instruction-level” techniques
  - We are developing a new technique based on parallel execution with input replay
But Remember... it's an Arms Race
But Remember... it's an Arms Race