01110111 01100101
01101100 01100011
01000000 01100100
01101111 01101110
01100101 01000001

Dynamic Analysis

Davide Balzarotti
davide.balzarotti@eurecom.fr

Binary Analysis Techniques

- Static Analysis
  - Examine the binary without running it
  - The only option when the program cannot be run
    (partial memory dump, missing pieces, unavailable architecture,...)
  - It can tell you everything the program can do

Binary Analysis Techniques

- Static Analysis
  - Examine the binary without running it
  - The only option when the program cannot be run
    (partial memory dump, missing pieces, unavailable architecture,...)
  - It can tell you everything the program can do

- Dynamic Analysis
  - Run the program and observe its behavior
  - It tells you exactly what the program does in a *given environment* and with a *particular input*

Observe Malware in its Habitat

- Before performing dynamic analysis, collect as much information as possible from the environment in which the malware was collected
  - Where was it detected?
    (in which OS, which version, in which country, ...)
  - How did it spread?
    (email attachment, dropped by a malicious PDF, pen drive, ...)
  - Can you get samples of the network traffic?
- The more you know about how the malware behaves in the wild, the better you can analyze it "in captivity"

Malware in a Cage

Malware Sandbox

- Idea:
  execute the malware inside an isolated and instrumented environment (the sandbox) to study its behavior

- Goal:
  - Collect network traffic
  - Observe filesystem and registry changes
  - Monitor processes and take memory snapshots
  - Obtain an instruction trace

- “Immune” to static analysis obfuscation, anti-disassembly tricks, and packers :)

Malware Sandbox Setup

- Set up a safe environment that
  1. Can easily be reverted to a pristine state
  2. Can control and contain the malicious activities performed by the malware sample
  3. Allows the analyst to collect information about the running sample

Malware Sandbox Setup

- Set up a safe environment that
  1. Setup
  2. Containment
  3. Instrumentation

Malware Sandbox Setup

- Set up a safe environment that
  1. Setup → *Security* / *Efficiency*
  2. Containment → *Isolation*
  3. Instrumentation → *Precision*

Malware Sandbox Setup

- Set up a safe environment that
  1. Setup
  2. Containment
  3. Instrumentation

- Several choices
  - Physical (bare metal) vs Virtual machines
  - Isolated or Internet-connected network
  - Internal vs External instrumentation

Do you want to *manually* test one sample,
or to *automatically* run one million?

Setting Up an Analysis Environment

- Running malware on your everyday machine is not a good idea
- The analysis environment must be restored to a pristine state after each analysis
  - Norton Ghost (or similar tools) to restore the disk
  - Dedicated write cache card
  - Virtual machine snapshots

Ready-to-Use Sandboxes

- Dozens of available choices
  - GFI Sandbox, CWSandbox, Anubis, Norman Sandbox, Comodo Instant Malware analysis system, ThreatExpert, Joebox, Cuckoo Sandbox, …
- Anubis
  - Available as a service - Discontinued in 2016
  - Used by: Shadow Server, Team Cymru, CERT Australia, law-enforcement agencies, ISPs, banks, anti-virus companies, ...
  - Was partially running at Eurecom
- Cuckoo
  - Free to download, easy to install, very well documented

Summary:

Description	Risk
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.	

Summary:

Description	Risk
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     GET / HTTP/1.1..
4163 6365 7074 3a20 2a2f 2a0d 0a41 6363     Accept: */*..Acc
6570 742d 456e 636f 6469 6e67 3a20 677a     ept-Encoding: gz
6970 2c20 6465 666c 6174 650d 0a55 7365     ip, deflate..Use
```

Summary:

Description	Risk
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

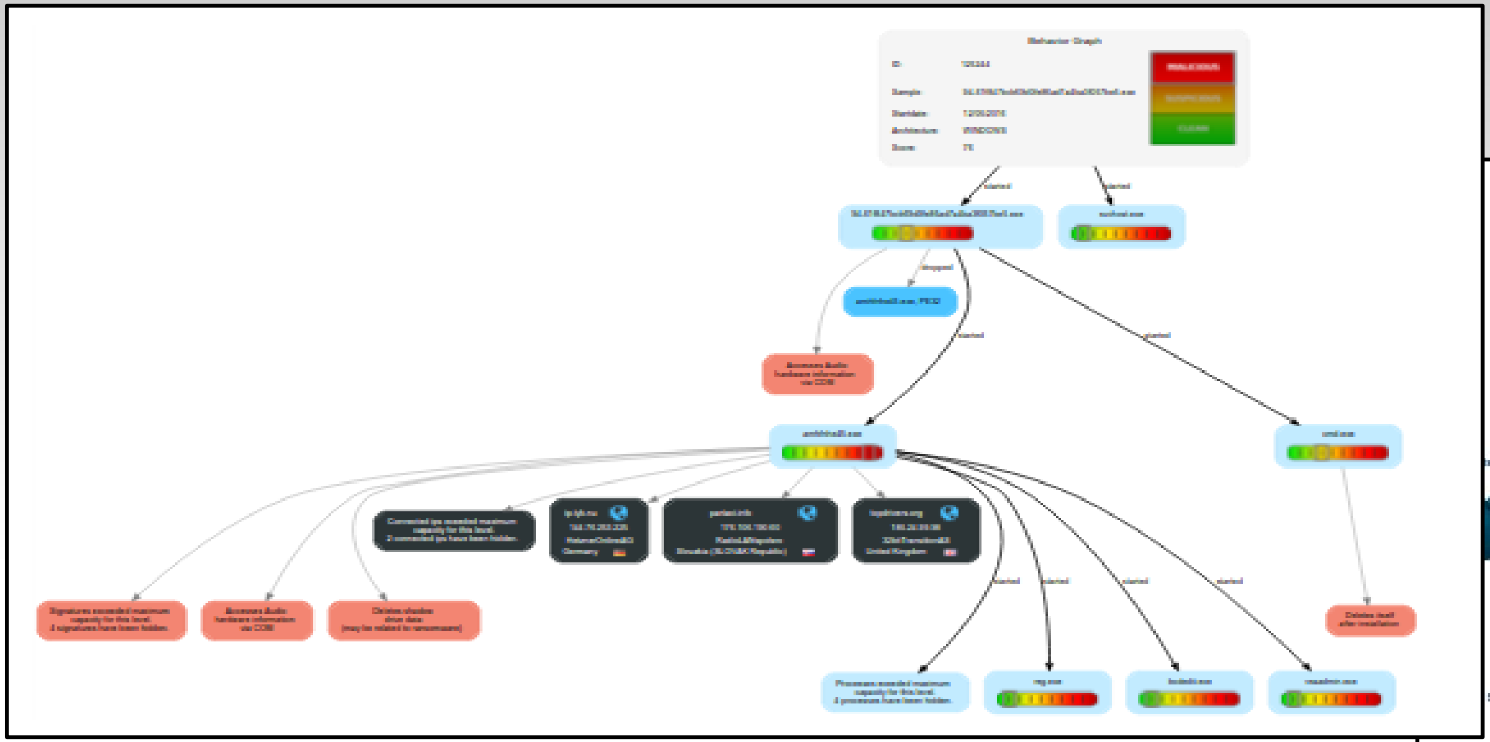
Data sent:

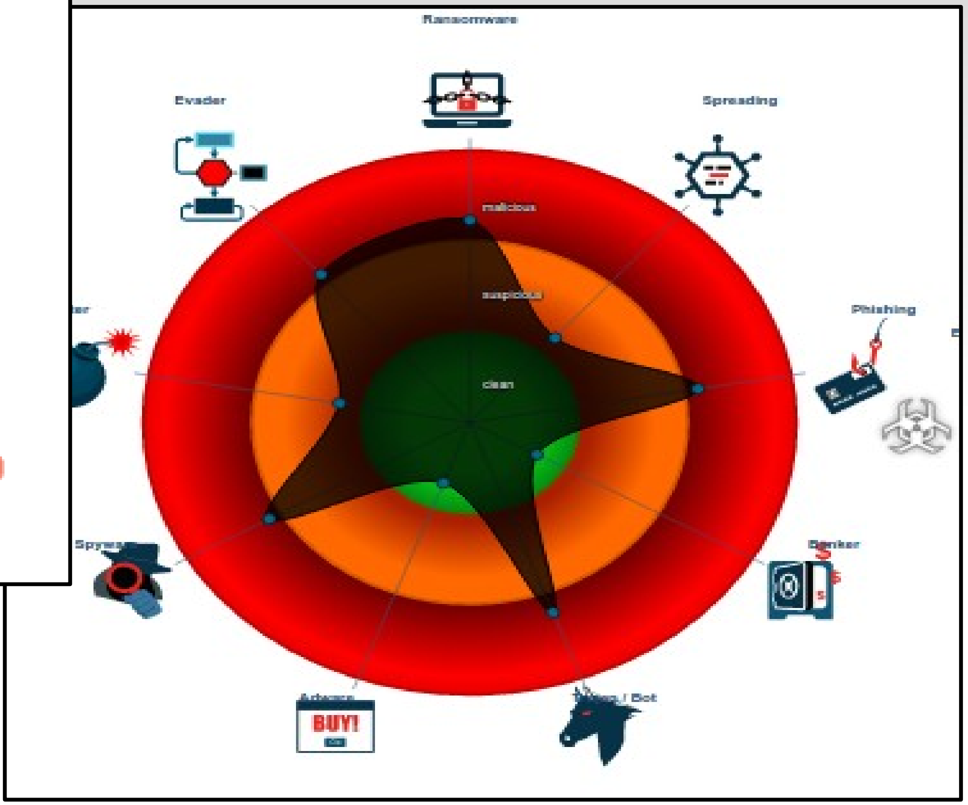
```
4745 5420 2f20 4854 5450 2f31 2e31 0d0a     GET / HTTP/1.1..
4163 6365 7074 3a20 2a2f 2a0d 0a41 6363     Accept: */*..Acc
6570 742d 456e 636f 6469 6e67 3a20 677a     ept-Encoding: gz
6970 2c20 6465 666c 6174 650d 0a55 7365     ip, deflate..Use
```

- Registry Values Modified:

Key	Name
HKLM\SYSTEM\CURRENTCONTROLSET\HARDWARE PROFILES\CURRENT\Software\Microsoft\windows\CurrentVersion\Internet Settings	ProxyEnable
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders	Common AppData
HKLM\Software\Microsoft\Windows\CurrentVersion\Run	jvzq

Modern Sandbox Report





http://www.joesecurity.org/reports/report-61f847bcb69d0fe86ad7a4ba3f057be5.html

Automated vs Manual Analysis

Automated systems

- Available in one click
- Already installed, configured, and supported by an expert team
- Do not require to run malware in your organization

Automated vs Manual Analysis

- Automated systems
  - 👍 Available in one click
  - 👍 Already installed, configured, and supported by an expert team
  - 👍 Do not require to run malware in your organization

  - 👎 Run the samples without parameters
  - 👎 No (or very limited and artificial) user interaction
  - 👎 Often miss the right trigger (e.g., visit a certain bank url)
  - 👎 Often limited to a particular Operating System
  - 👎 Known by the attackers (targeted evasion techniques, blocked IPs, …)
  - 👎 Fixed, not customizable environment

VM vs Bare metal

- Real machine
  - Accurate
    (malware can detect or malfunction when run inside a virtual machine)
  - Difficult to instrument
  - Difficult to clean after the analysis
  - Does not scale very well
- Virtual Machine
  - Easy to instrument and monitor
  - Easy to save/restore snapshots
  - Easily scalable
  - Prone to evasion

Emulators & Virtual Machines

- Memory + CPU Emulation
  - Read the sample instructions and execute them in an emulated environment
  - Employed by many anti-viruses to deal with packed binaries
- Full System Emulation
  - The entire computer system (with its peripherals) is emulated
  - A normal OS can be installed in the emulator
  - Transparent analysis
  - Semantic Gap (infer high-level info from a raw view of the memory)
  - Trivial to produce an instruction trace, not so trivial to monitor APIs
  - E.g., QEmu

Emulators & Virtual Machines

- Virtual Machine
  - The host and guest architectures are identical (no ARM on x86)
  - Guest Virtual Machines can execute non privileged instruction on the hardware
    - Hard to get an instruction trace
    - Need to single-step inside the OS
  - A Virtual Machine Monitor (VMM) controls the privileged instructions and is in charge of the hardware
  - E.g., VMWare

Ethical and Legal Issues

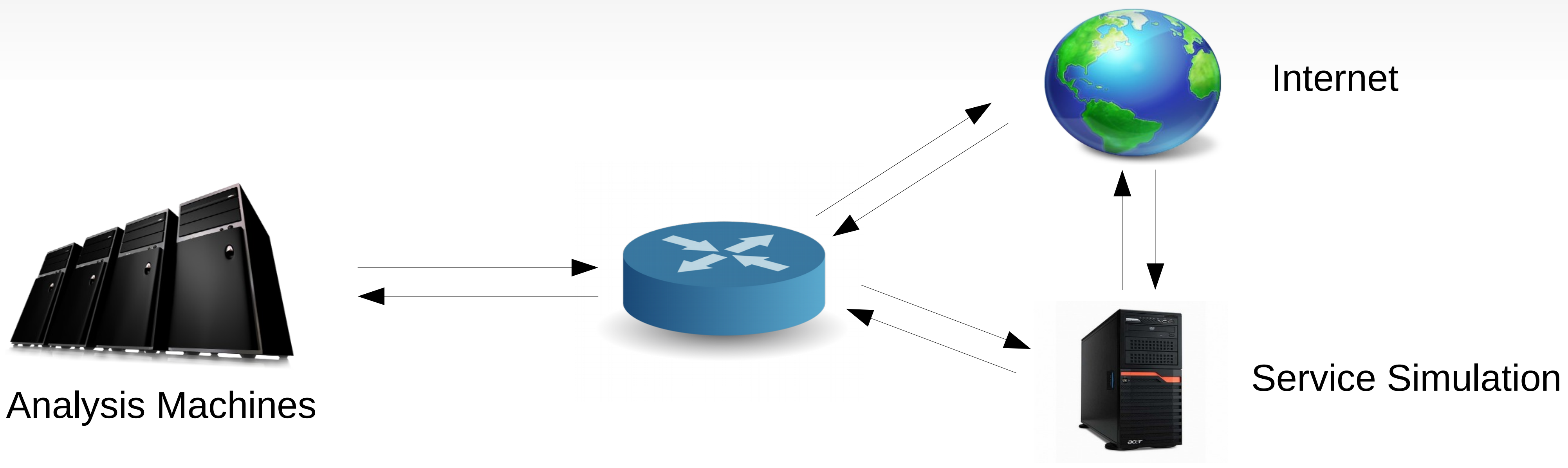
- Running malware is not illegal...

  ...but what the malware does when it runs it may be

- You should take the appropriate countermeasures to avoid
  - Sending SPAM
  - Running attacks against other machines
  - Performing DOS attacks
  - Scan other networks

Network Containment

- The traffic from the analysis environment should be carefully filtered
  - How? .. well this is still an open question :(
  - At least SMTP traffic should be redirected, and the number of outgoing connections limited
  - Well-known services can be simulated



INetSim

- Linux software that emulates a number of protocols
  - SMTP(S), HTTP(S), FTP(S), POP3(S), DNS, IRC, NTP, TFPT, Time, Echo
  - Generic TCP and UDP socket servers
- E.g, the web server can be configured to return a specific page, or to answer to any URL by returning the appropriate document type
  - If the malware requests `/docs/FGSX/new.pdf`, INetSim sends back a valid `pdf`

Limitations

- ✔ Beacons
- ✔ Probes
- ✔ Stalling code
- ✔ Logic bombs
- ✔ VM detection
- ✔ Analysis environment detection

Operational Security (OPSEC)

- Military concept used to describe the fact that our operations can leak important information when observed by the enemy
- In malware analysis:
  - Disclose the fact that a malware sample is under analysis
  - Disclose important information about the analysis environment
- Examples
  - Beacons
  - Probes

Beacons

- Many malware samples call back to report the successful infection, or just connect to a server to download upgrades or upload information
  - A connection coming from an unexpected IP (or one associated to public sandboxes) may alert the enemy that the sample is under analysis
- Samples can also contain decoy beacons
  - E.g., the binary contains a list of domain names, one of which is never used in practice
  - Any request for that domain is an evidence of a manual analysis

Antivirus Tracker

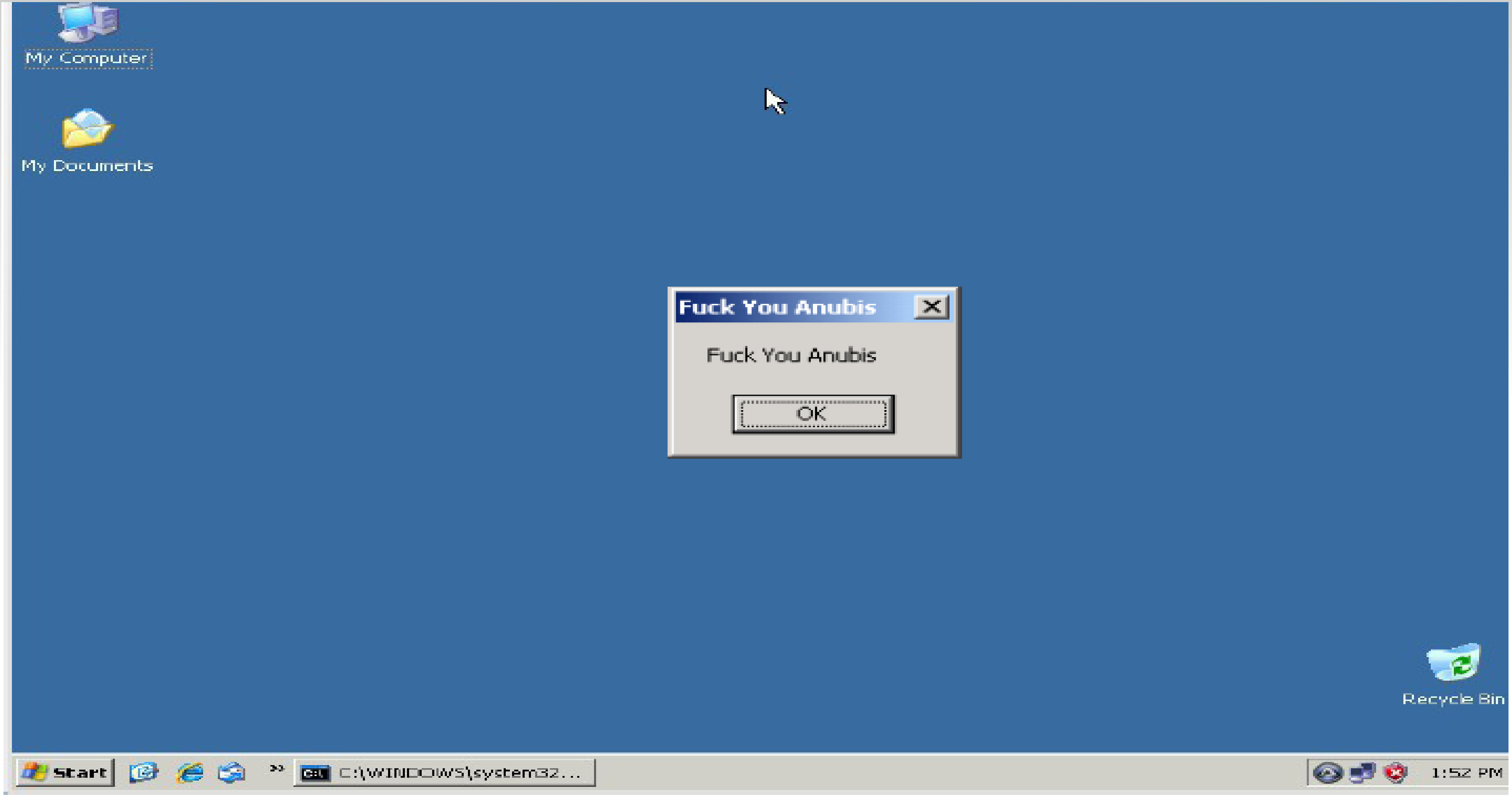
54 entrys in avtracker.info database | Plain IPs | IRC | IP Tables | API | .htaccess

IP	HOST	COUNTRY	DATE, TIME	COMPUTER	USER	OS	COMMENT
61.181.247.146	61.181.247.146	China	6th Jun 10			Windows 5.1	AhnLab
80.13.75.21	LRouen-152-83-12-21.w80-13.abo.wanadoo.fr	France	27th Jan 12	pc9	Administrator	Windows 5.1	Anubis
82.245.40.203	lac49-1-82-245-40-203.fbx.proxad.net	France	28th Jan 12				Anubis
128.130.56.11	128.130.56.11	Austria	20th Oct 09	pc8	Administrator	Windows 5.1	Anubis
128.130.56.12	128.130.56.12	Austria	20th Oct 09	pc5	Administrator	Windows 5.1	Anubis
128.130.56.14	128.130.56.14	Austria	17th Oct 09	pc5	Administrator	Windows 5.1	Anubis
128.130.56.16	128.130.56.16	Austria	15th Oct 09	pc5	Administrator	Windows 5.1	Anubis
128.130.56.23	worker-23.seclab.tuwien.ac.at	Austria	7th Jun 10	pc8	Administrator	Windows 5.1	Anubis
128.130.56.24	worker-24.seclab.tuwien.ac.at	Austria	19th Aug 10	pc4	Administrator	Windows 5.1	Anubis
128.130.56.68	128.130.56.68	Austria	6th Jun 10	pc9	Administrator	Windows 5.1	Anubis
80.13.75.21	LRouen-152-83-12-21.w80-13.abo.wanadoo.fr	France	26th Jan 12	pc8	Administrator	Windows 5.1	Anubis, iSecLab
217.86.133.28	pd956851c.dip0.t-ipconnect.de	Germany	7th Jun 10	HBXPENG	makrorechner	Windows 5.1	Avira Lab
64.95.48.100	64.95.48.100	United States	19th Oct 09	NONE-DUSEZ58JO1	Administrator	Windows 5.1	Basin Creations
91.199.104.3	3.bitdefender.com	Romania	16th Oct 09				Bitdefender
91.199.104.4	4.bitdefender.com	Romania	16th Oct 09				Bitdefender
91.199.104.15	15.bitdefender.com	Romania	16th Oct 09	tz	Administrator	Windows 5.1	Bitdefender
64.128.133.131	[*] 64-128-133-131.static.twtelecom.net	United States	19th Aug 10	HOME-OFF-D5F0AC	Dave	Windows 5.1	CWSandbox
88.130.42.70	mue-88-130-42-070.dsl.tropolys.de	Germany	7th Jun 10	DELL-D3E62F7E26	Administrator	Windows 5.1	CWSandbox
134.155.241.17	yoshi.informatik.uni-mannheim.de	Germany	15th Oct 09	DELL-D3E62F7E26	Administrator	Windows 5.1	CWSandbox
216.245.222.15	[*] 15-222-245-216.reverse.lstn.net	United States	19th Aug 10	HOME-OFF-D5F0AC	Dave	Windows 5.1	CWSandbox
46.102.243.70	70.243.102.46.static.intovps.com	Romania	28th Jan 12				Cuckoobox
208.118.60.155	208-118-60-155.alchemy.net	United States	26th Feb 10	rtrtrele	Administrator	Windows 5.1	CyberDefender
109.74.154.83	109.74.154.83	Slovakia	28th Jan 12				ESET
195.168.53.57	gw-hq.eset.com	Slovakia	15th Jun 10			Windows 5.1	ESET
66.129.97.254	[*] 66.129.97.254	United States	26th Jan 12	HOME-OFF-D5F0AC	Dave	Windows 5.1	GFI Sandbox
72.64.146.112	[*] static-72-64-146-112.tampfl.fios.verizon.net	United States	26th Jan 12	JONATHAN-C561E0	Administrator	Windows 5.1	GFI Sandbox
188.62.232.157	157-232.62-188.cust.bluewin.ch	Switzerland	7th Jun 10	HANS	Hanuele Baser	Windows 5.1	Joebox
212.5.80.7	muzzle.kaspersky-labs.com	Russian Federation	20th Oct 09		N00b	Windows 5.1	Kaspersky
91.103.66.1	ace1.kaspersky-labs.com	Russian Federation	15th Jun 10			Windows 5.1	Kaspersky Lab
91.103.66.2	ace2.kaspersky-labs.com	Russian Federation	15th Jun 10			Windows 5.1	Kaspersky Lab
91.103.66.3	ace3.kaspersky-labs.com	Russian Federation	15th Jun 10			Windows 5.1	Kaspersky Lab
91.103.66.4	ace4.kaspersky-labs.com	Russian Federation	15th Jun 10			Windows 5.1	Kaspersky Lab
149.20.63.55	[*] 149.20.63.55	United States	26th Jan 12	LAB	Me	Windows 5.1	Malwr Cuckoo Sandbox

Probes

- Special programs designed to
  - harvest data about the environment in which they are executed
  - send back the collected information using an hidden channel
- Used to analyze the analysis systems
  - Virtual machine, Windows version and registration code, Harware drivers, Installed software, User name, Filesystem information, network cards, …
  - The collected information are used to design malware that can evade detection

The Result



My Computer
My Documents
Fuck You Anubis
Fuck You Anubis
OK
Recycle Bin
start
C:\WINDOWS\system32...
1:52 PM

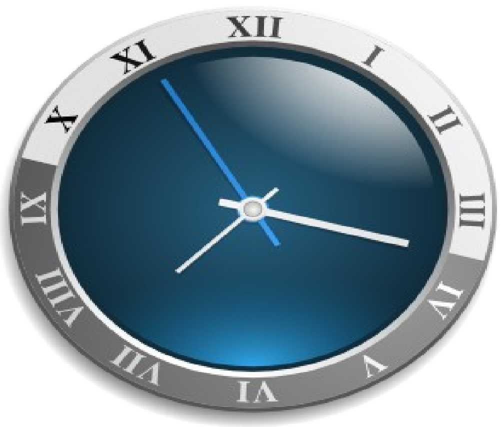
The Result


Mal Packer 1.2 Private for
Menu | Buy Private Version |
+
-
Compress File
Anti-Anubis
Pack the file!
Autostart
Use BeRoPaCkEr
XOR Encrypt
Use Other Stub [Private]
External Stub
Change Icon

Limitations: Stalling Code

- How long do you run/observe a given sample ?
  - One of the simplest problems..
    and still one of the most difficult to solve

- Large-scale malware analysis systems usually run each sample ~~20~~, ~~10~~, ~~5~~, ~1/2 minutes
- To evade detection, a malware sample could easily wait 1h before performing any malicious activity



Limitations: Logic Bombs

- Logic bombs are samples that wait for a certain trigger to exhibit the malicious behavior
- Examples
  - Perform the malicious task only on a certain date (e.g., the first day of the month)
  - Wait for a certain number of keystrokes

Limitations: Anti-VM

- Anti-VM:
  - Malware analysis often use VMs (or emulators) to run malware samples
  - The average user does not use (yet) VMs to browse the web

  → An increasing percentage of malware "refuses" to run inside virtual environments

- But, how a program can reliably identify if it is running inside a virtual environment?

VM-Detection

- Look for VM artifacts in processes, file system, and registry
- Look for difference in memory structures
- CPU semantics attacks
- Timing

VM-Detection

- Look for VM-specific artifacts
  - Over 300 references of “WMware” in the Registry
  - When `VMtools` are running, there are three processes in memory (`VMwareService.exe, VMwareTray.exe, VMwareUser.exe`)
  - Device drivers names
  - Network card MAC address
- Look for difference in memory structures
- CPU semantics attacks
- Timing

VM-Detection

- Look for VM-specific artifacts
- Look for difference in memory structures
  - Critical OS tables are typically relocated in a virtual machine
  - E.g.,
    - Interrupt Descriptor Table (IDT)
    - Global Descriptor Table (GDT)
    - Local Descriptor Table (LDT)
- CPU semantics attacks
- Timing

Red Pill

- Simple technique developed by Joanna Rutkowska in 2004
  - The tool runs a single assembly instruction:
    `SIDT` (Store Interrupt Descriptor Table)
  - This instruction, executable from user space, stores in memory the location of the IDT register (that points to the IDT)
  - Rutkowska observed that IDT is located at a higher address in a guest OS
  - Unreliable on multi-core machines

VM-Detection

- Look for VM-specific artifacts
- Look for difference in memory structures
- CPU semantics attacks
  - Illegal opcodes supported by the VM
    (e.g., to communicate with the host machine)
  - Emulated instructions can have different undocumented behaviors, or not have bugs known to be associated with certain CPUs
- Timing

VM-Detection

- Look for VM-specific artifacts
- Look for difference in memory structures
- CPU semantics attacks
- Timing
  - Executing certain instructions many times takes longer within a VM than on a normal system
  - E.g., execute the `CPUID` instruction (that is trapped by the hypervisor) and then measure the time to access pages in the Translation Lookaside Buffers (TLB)
  - Other techniques can require an external time source

Limitations: Analysis Env. Detection

- Detect that something is anomalous in the environment
  - Not enough entries in the system log / browser history
  - Inconsistency between the installation time and the “age” of the system
  - Simulated services that should not exist

Dealing with the Limitations

- Disguising
  - Randomize the environment
  - Use rootkit-like techniques to hide information about the analysis environment

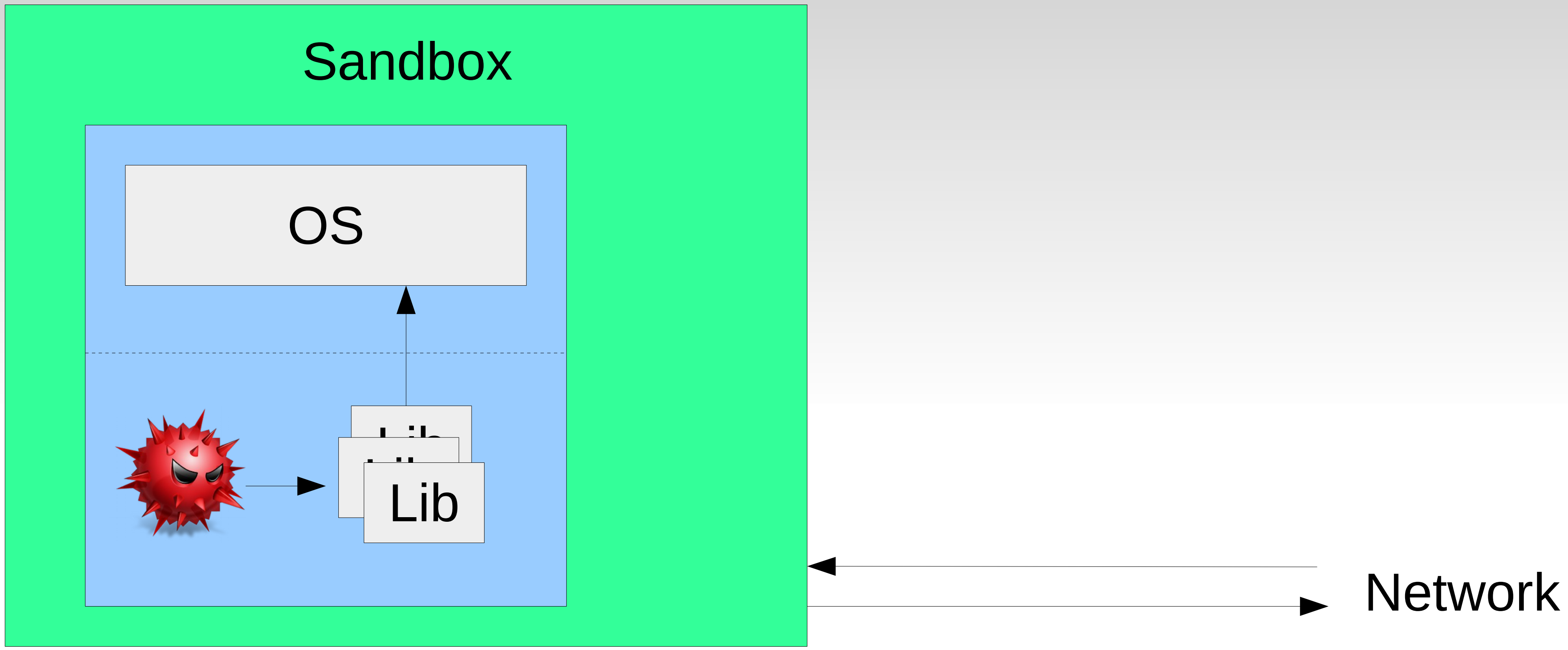
Dealing with the Limitations

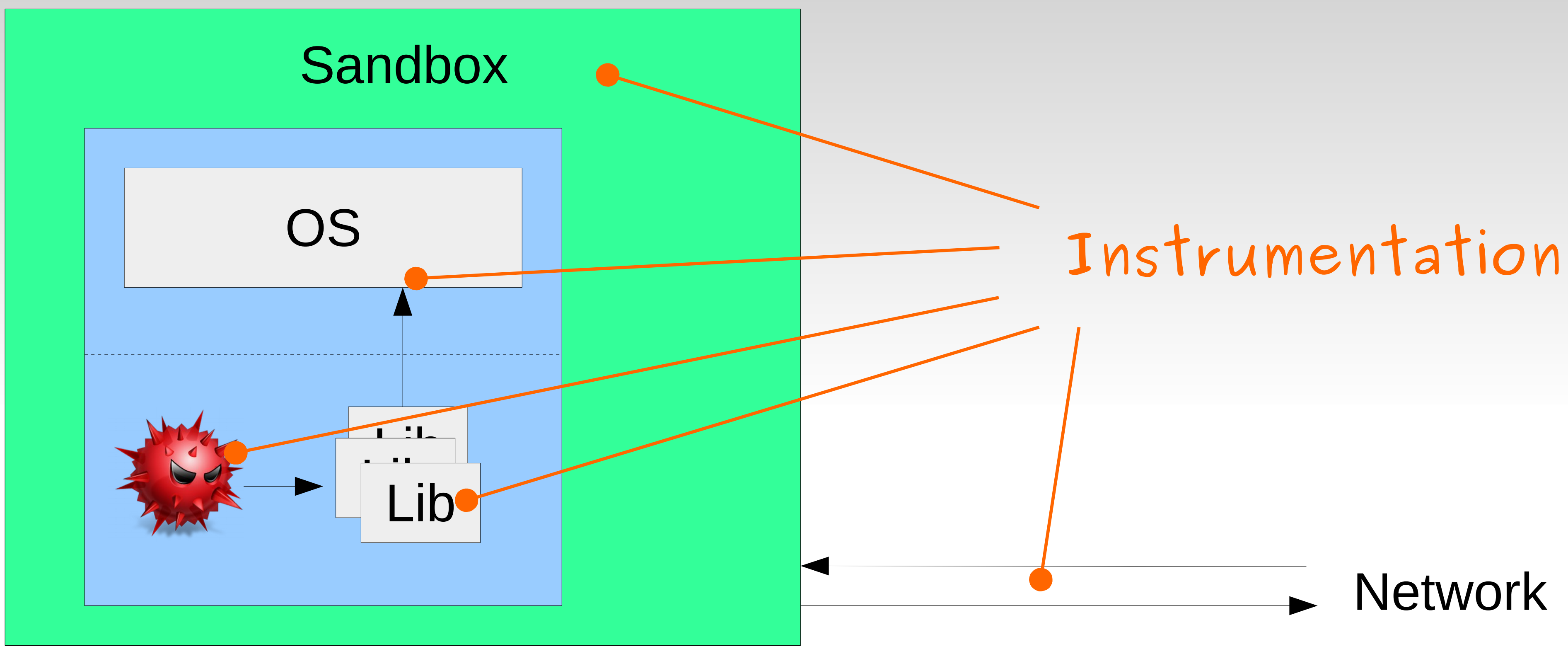
- Disguising
  - Randomize the environment
  - Use rootkit-like techniques to hide information about the analysis environment
- Multi-path exploration
  1. Recognize branching points where the control flow is based on data that originates outside the monitored process
  2. Take a snapshot of the process
  3. Execute one side of the branch, then restore the snapshot and force the process to take the other path

Dealing with the Limitations

- Disguising
  - Randomize the environment
  - Use rootkit-like techniques to hide information about the analysis environment
- Multi-path exploration
  1. Recognize branching points where the control flow is based on data that originates outside the monitored process
  2. Take a snapshot of the process
  3. Execute one side of the branch, then restore the snapshot and force the process to take the other path
- Time warp
  - Force loops to finish
  - Change the behavior of the `sleep()` function

Instrumentation


Sandbox
OS
Lib
Lib
Lib
Network


Sandbox
OS
Lib
Instrumentation
Network

Function Call Monitoring

- Intercepting the functions called by a program help gaining an overview of the behavior of the program
- Windows APIs
  - Set of APIs that provide access to different functionalities (networking, security, system services, management, …)
- System Calls
  - Interface between user-space and kernel-space
  - Mechanism that allows a user-mode application to request the operating system to perform a tasks on its behalf (e.g., opening a file)
- Windows Native APIs
  - Layer between the stable Windows APIs and the System calls
  - Not documented, and they often change between different versions

DLL Injection

DLL Injection

- Technique used to run code within the address space of another process by forcing it to load a dynamic library
  - After a DLL is loaded, the system executes the `DllMain`
  - Used by malware, goodware, OSs,..

- Windows
  - Use the registry
  - `SetWindowsHookEx`
  - Create a remote thread
- Linux
  - `$ LD_PRELOAD="./test.so" program`

DLL Injection

- Using the registry
  - Every DLL listed in

    ```
    HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
     NT\CurrentVersion\Windows\AppInit_DLLs
    ```

    is loaded by every application that uses user32.dll
- SetWindowsHookEx
- Creating a remote thread.

DLL Injection

- Using the registry
- SetWindowsHookEx
  - Handy Windows API function to hook window messages (e.g., mouse button pressed) inside *another process*

```
HHOOK WINAPI SetWindowsHookEx(
  __in  int idHook,
  __in  HOOKPROC lpfn,    # pointer to the callback
  __in  HINSTANCE hMod,   # DLL containing the callback
  __in  DWORD dwThreadId # which thread to hook
);
```

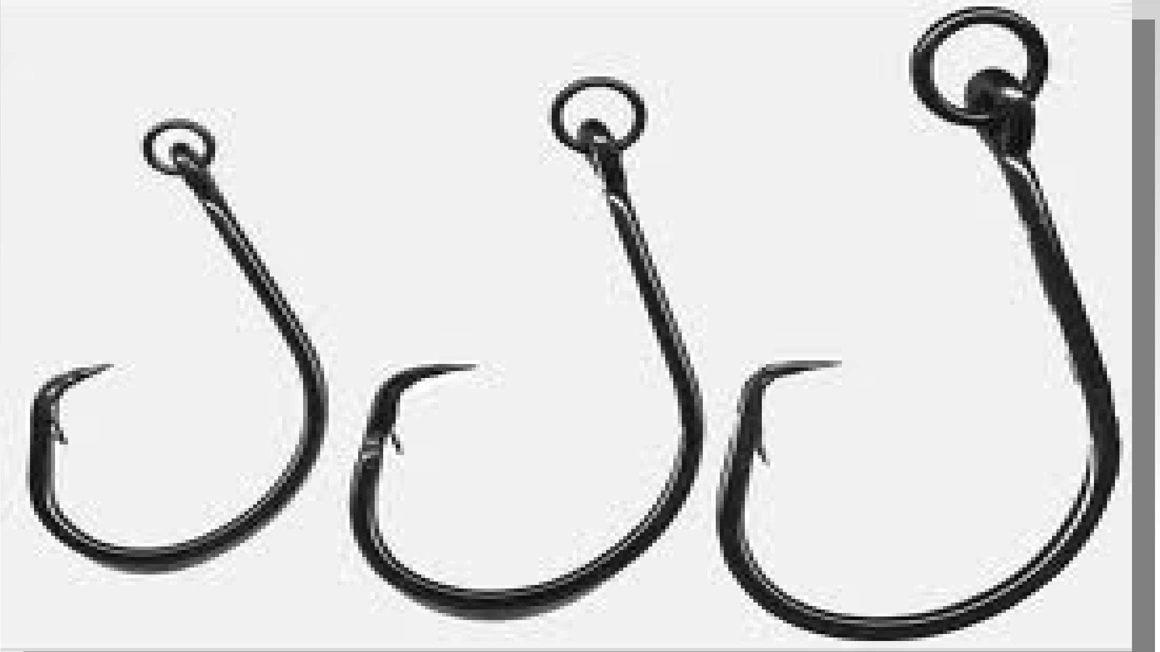
  - Once executed, the function will load the DLL inside the target process
- Creating a remote thread

DLL Injection

- Using the registry
- SetWindowsHookEx
- Creating a remote thread
  1. Obtain a handle to the target process
  2. Allocate some memory for the DLL
  3. Write the name of the DLL in the allocated memory
  4. Create a thread in the process and tell it to execute `LoadLibraryA`

DLL Injection

```
int Inject_DLL(long pidProcAInjecter , char* dll_to_inject);
long ProcessToPid(char* process);

int Inject_DLL(long pidProcAInjecter , char* dll_to_inject)
{
    long dll_size = strlen(dll_to_inject) + 1;

    printf("-> Opening the target process...\n");

    HANDLE MyHandle = OpenProcess(PROCESS_ALL_ACCESS , FALSE , pidProcAInjecter);
    if(MyHandle == NULL)return 0;

    printf("-> Memory Allocation...\n");
    LPVOID MyAlloc = VirtualAllocEx( MyHandle , NULL , dll_size , MEM_COMMIT , PAGE_EXECUTE_READWRITE);
    if(MyAlloc == NULL)
        return 0;

    printf("-> Writing DLL in memory...\n");
    int IsWriteOK = WriteProcessMemory( MyHandle , MyAlloc , dll_to_inject , dll_size , 0);
    if(IsWriteOK  == 0)
        return 0;

    printf("-> Creating the Thread...\n");
    DWORD identificateurThread ;
    LPTHREAD_START_ROUTINE addrLoadLibrary = (LPTHREAD_START_ROUTINE)GetProcAddress(LoadLibrary("kernel32"),"LoadLibraryA");
    HANDLE ThreadReturn= CreateRemoteThread( MyHandle , NULL , 0 , addrLoadLibrary , MyAlloc , 0 , &identificateurThread );
    if(ThreadReturn == NULL)
        return 0;

    if ((MyHandle != NULL) && (MyAlloc != NULL) && (IsWriteOK != ERROR_INVALID_HANDLE) && (ThreadReturn != NULL))
    { printf("-> DLL injected :]\n");
```



Hooking

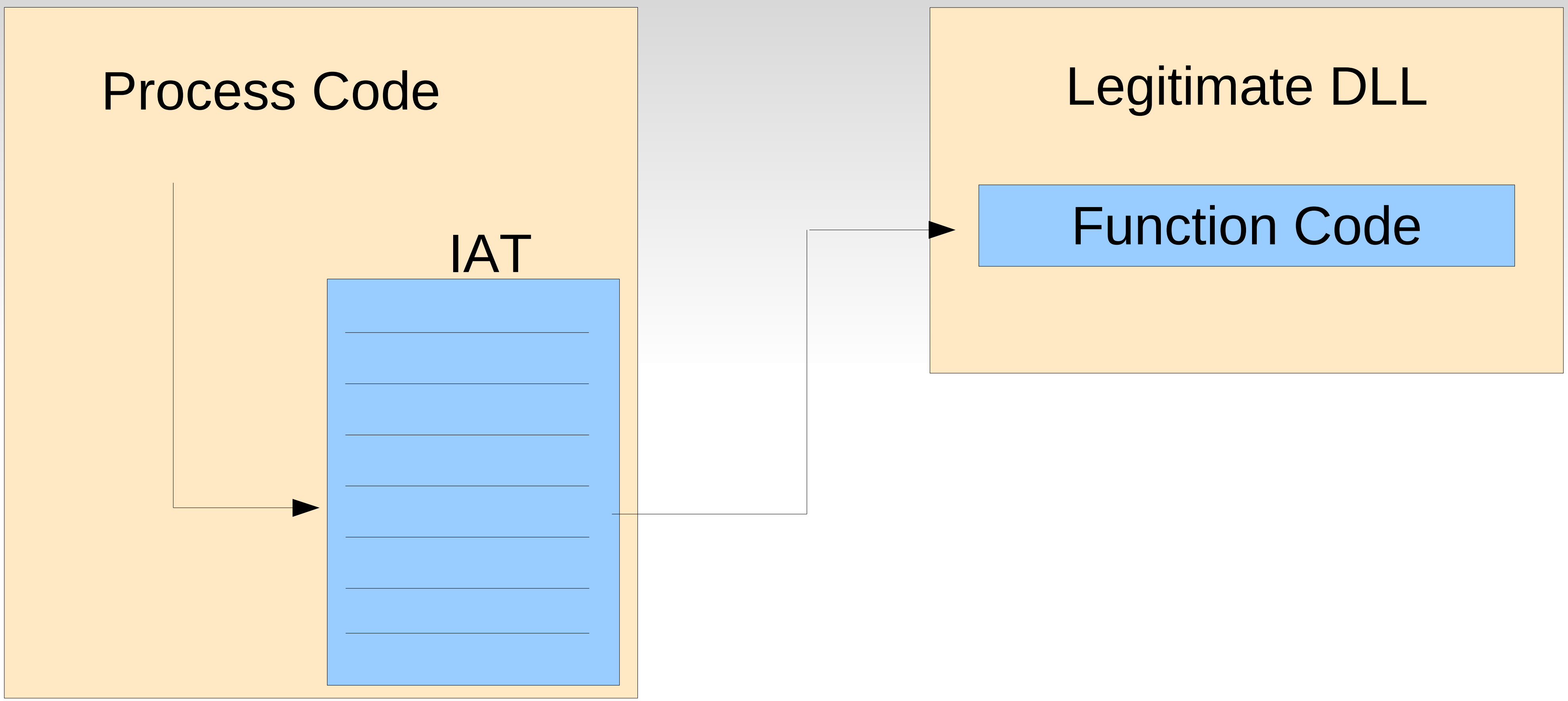
Hooking

- The process of intercepting and instrumenting API calls
- Often adopted by malware to tamper with other processes
  - But also very useful for dynamic analysis, to extract information about unknown programs
- Hooks can be placed in both user- or kernel-space
  - Inline hooking
  - Import Address Table (IAT) hooking
  - Export Address Table (EAT) hooking
  - System Service Table (SSDT) hooking
  - Interrupt Table hooking
  - I/O Request Packet hooking

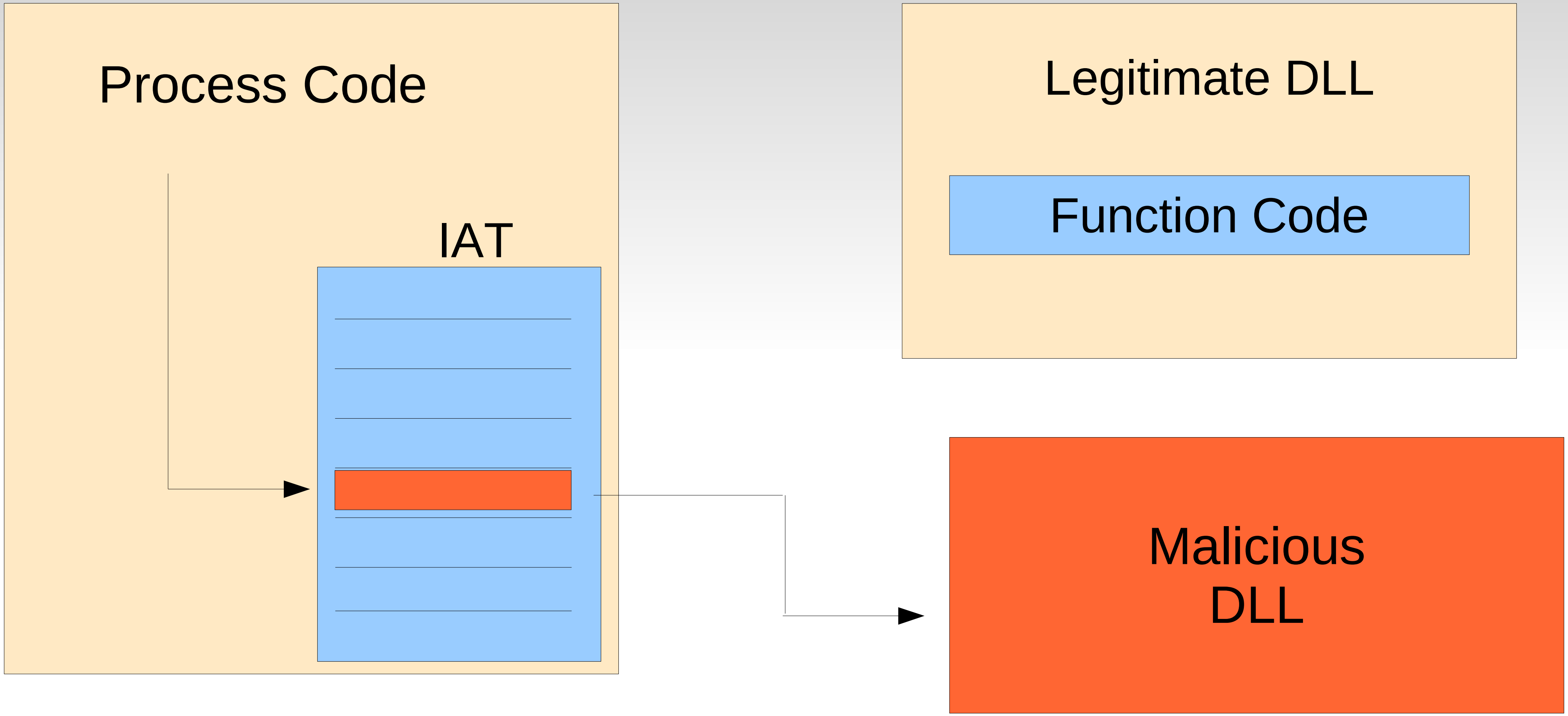
IAT Redirection

- Each imported API has its own reserved spot in the IAT where the address of the imported function is written by the Windows PE loader
- IAT redirection just overwrites the address of the imported function in the IAT, to point to the hooking function
- Limitations
  - It only works on imported functions (not on internal functions)
  - Hard to deal with runtime binding
  - Easy to detect
    (but not always easy to tell if the hook is malicious)

IAT Hooking



IAT Hooking



Process Code
IAT
Legitimate DLL
Function Code
Malicious
DLL

EAT Redirection

- The Export Address Table (EAT) works in the opposite direction of the IAT
  - When a module exports a function it stores the address of that function in it's EAT
  - EAT redirection works by overwriting that address with the offset of your hook (e.g., redirecting `WriteFile` in `Kernel32.dll`)
  - EAT redirection will not effect any currently loaded modules (!)

Inline Hooking

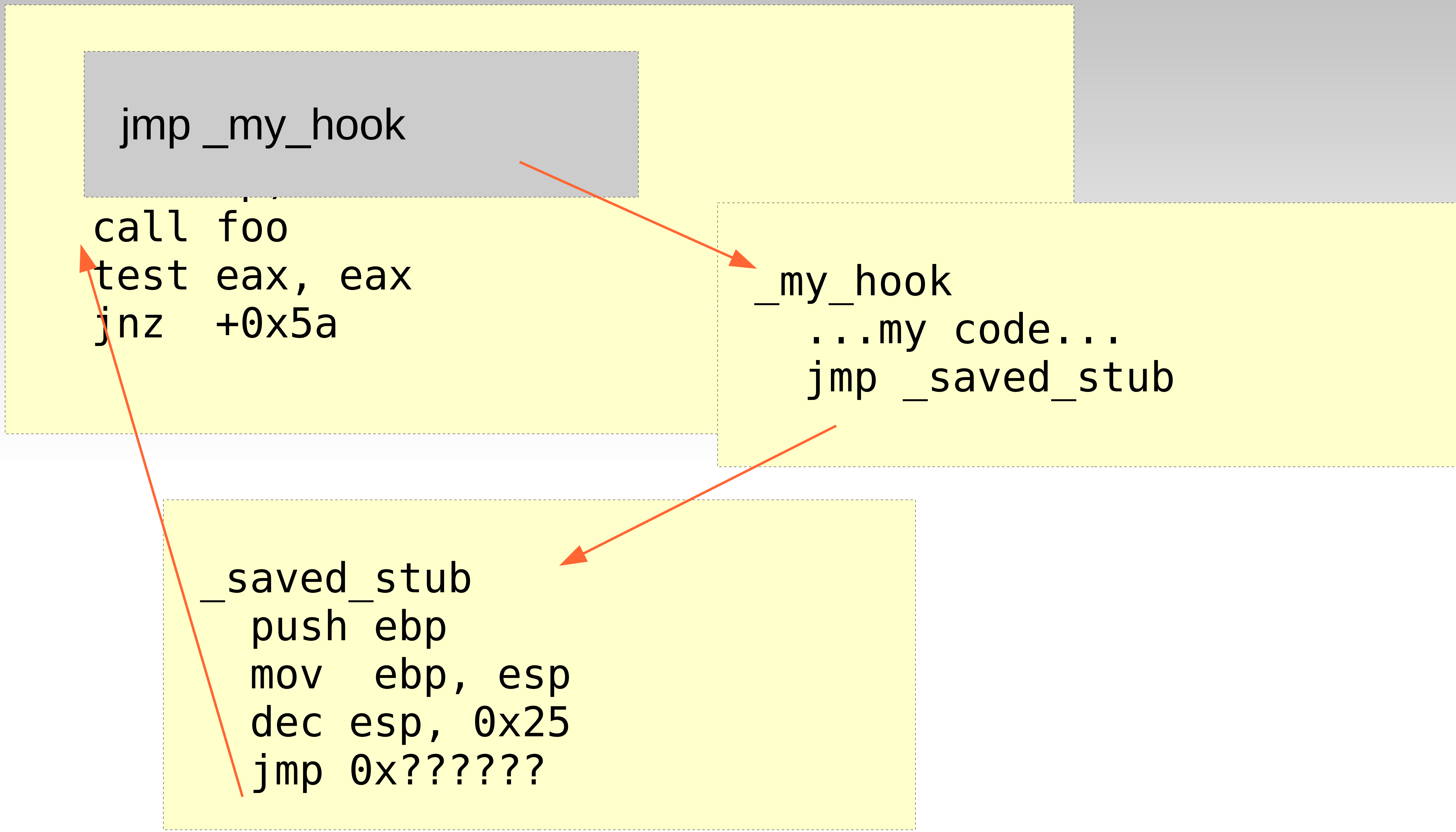
- Overwrite the first few bytes of a function to jump to the injected code
  - Can be used to hook any functions, also the internal ones
  - Used by many famous malware (Zeus, Hacker Defender,..)

- Approach:
  1. Inject a DLL with our code
  2. Copy the first few bytes of the target function
  3. Overwrite the first bytes of the target function with a jump (or call, or equivalent) to our hook
  4. The hook can execute the original function by re-establishing the original bytes
  5. After the original function is executed, the hook can regain control

```
push ebp
mov  ebp, esp
dec esp, 0x25
call foo
test eax, eax
jnz  +0x5a
```

```
call foo
test eax, eax
jnz  +0x5a
```

```
_saved_stub
  push ebp
  mov  ebp, esp
  dec esp, 0x25
  jmp 0x??????
```


jmp _my_hook
call foo
test eax, eax
jnz +0x5a
_my_hook
...my code...
jmp _saved_stub
_saved_stub
push ebp
mov ebp, esp
dec esp, 0x25
jmp 0x??????

Inline Hooking

- A long jump takes 5 bytes
- After Windows XP SP2, Microsoft decided to help inline hooking (they call it hot-patching) by changing the header of the functions

```
nop
nop
nop
nop
nop
<Function start address>:
mov edi, edi
push ebp
mov ebp, esp
```

Inline Hooking

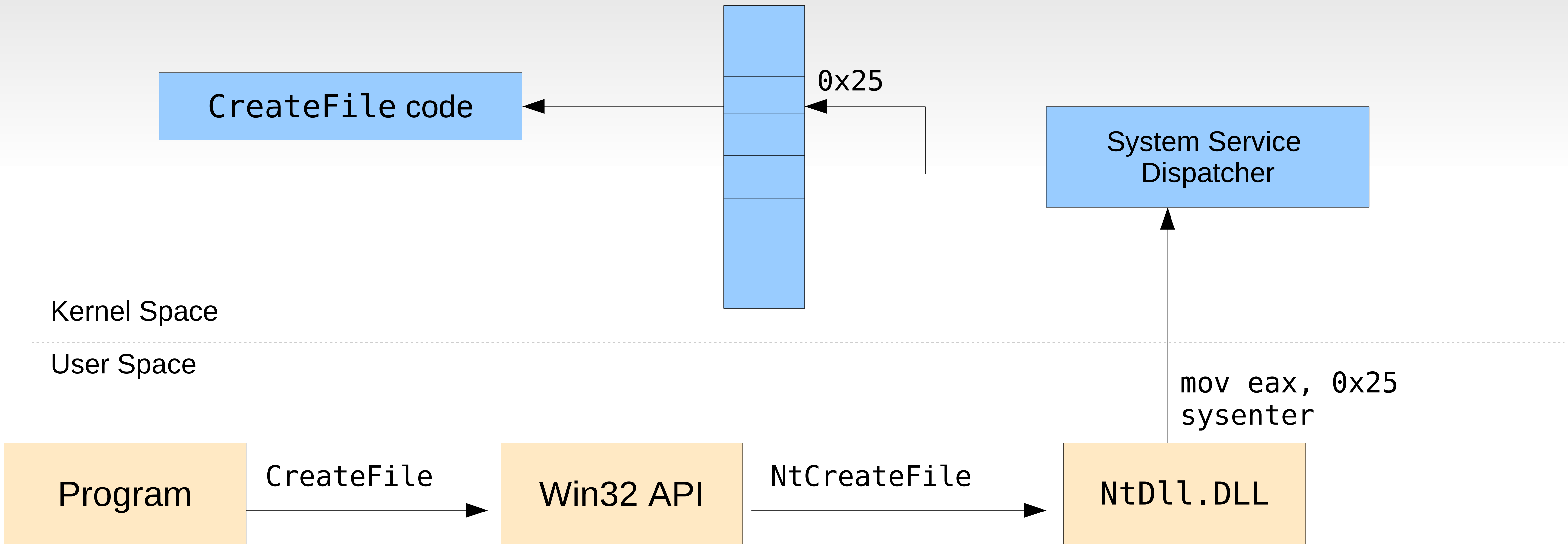
- A long jump takes 5 bytes
- After Windows XP SP2, Microsoft decided to help inline hooking (they call it hot-patching) by changing the header of the functions

```
nop
nop
nop
nop
nop
<Function start address>:
mov edi, edi
push ebp
mov ebp, esp
```

```
jmp hook



<Function start address>:
jmp -5
push ebp
mov ebp, esp
```

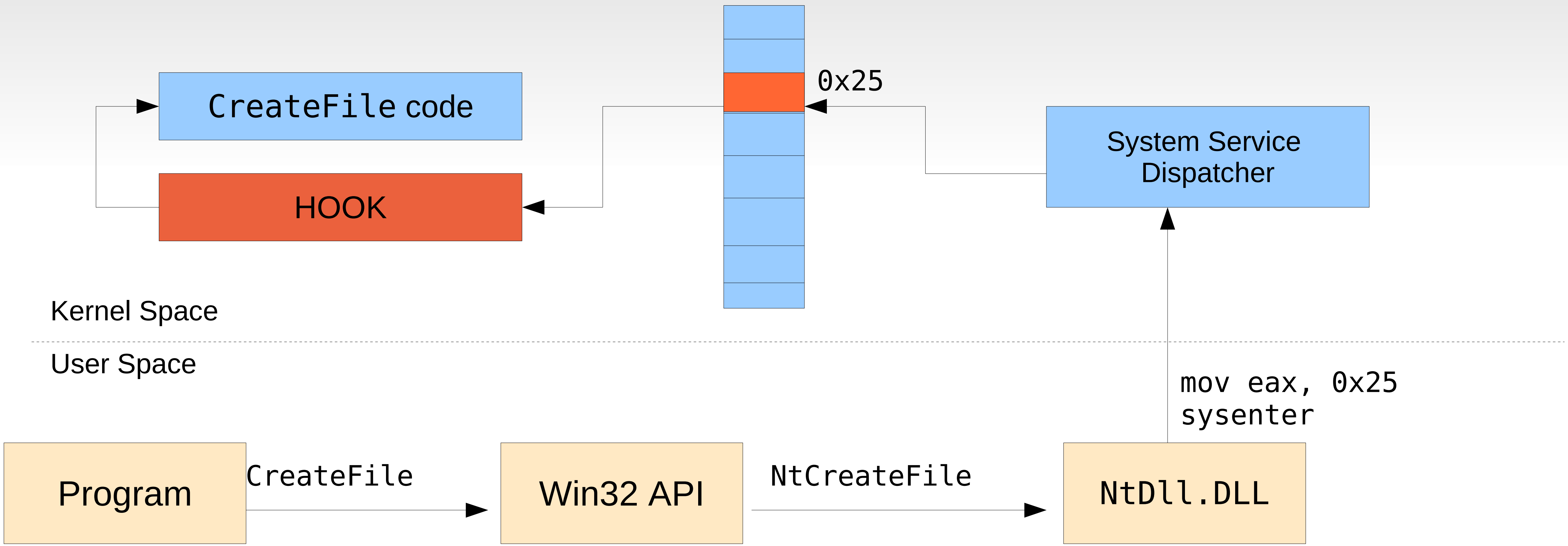
SSDT Hooking

- The SSDT is a kernel table that stores pointers to system API functions
  → Any hook affect the entire system



SSDT Hooking

- The SSDT is a kernel table that stores pointers to system API functions → Any hook affect entire system



How Sandbox Instrumentation Works

CWSandbox

- Can be executed in a VM or on bare metal
- Analysis
  - Performed by hooking the Windows API and intercepting the system calls
- Approach
  - A DLL (`cwmonitor.dll`) is injected in the malware process
  - The DLL performs inline hooking of all the API calls
  - The hook in the LoadLibrary function take cares of loading the `cwmonitor.dll` when new libraries are loaded
  - All system objects that could reveal the presence of the analysis framework are “sanitized” by the hooks so that they never reach the process under analysis

Joebox

- Analysis
  - Performed by hooking the Windows API and intercepting the system calls
  - Uses the AutoIT toolkit to emulate user interaction with the machine during the analysis phase
- Approach
  - SSDT hooking
  - User-mode EAT hooking
  - A kernel driver is responsible for cloaking the performed changes
    - the memory pages containing the executable code of the hooked function are marked as “not present”
    - When the process tries to read from that memory page (to detect possible modifications) the kernel module returns a fake, unmodified version of the page to the application

Anubis

- Windows XP running in a full-system emulator (Qemu)
- Analysis
  - Performed by monitoring the invocation (and the parameters) of Windows API functions, as well as the calls to the Windows Native API
  - Analysis limited to the sample process (identified by the CR3 register) and all processes created by it
- Approach
  - Function calls are identified by comparing the EIP of the emulated CPU with the known entry points of the Windows library functions
  - These addresses depend on where the libraries are loaded in memory → Anubis needs to track all the invocation of the dynamic loader