02
Introduction to Malware

An introduction to malware from zero to hero

- For a long time, malicious programs have been existing in our environment

- Today, they are a common threat. We can find them in our daily life: Browsing, reading e-mails, installing apps on our smartphone…

- For the purpose of this training, we will focus on **Windows** malware.

Common motives

For attackers, writing malware has different purposes.

- Harming users computers
- Creating botnets for DDoS attacks.
- Data theft: Credit-cards, user’s credentials, device information, confidential/valuable documentation.
- Other benefits: using huge networks of infected machines to perform an specific action (e.g: Mining monero)

Malware classification

- **Downloaders / Dropper**
- Rootkits
- **Banking trojans**
- **Ransomware**
- **Stealers / RATs**
- Worm-like malware
- **Botnets**

Downloaders

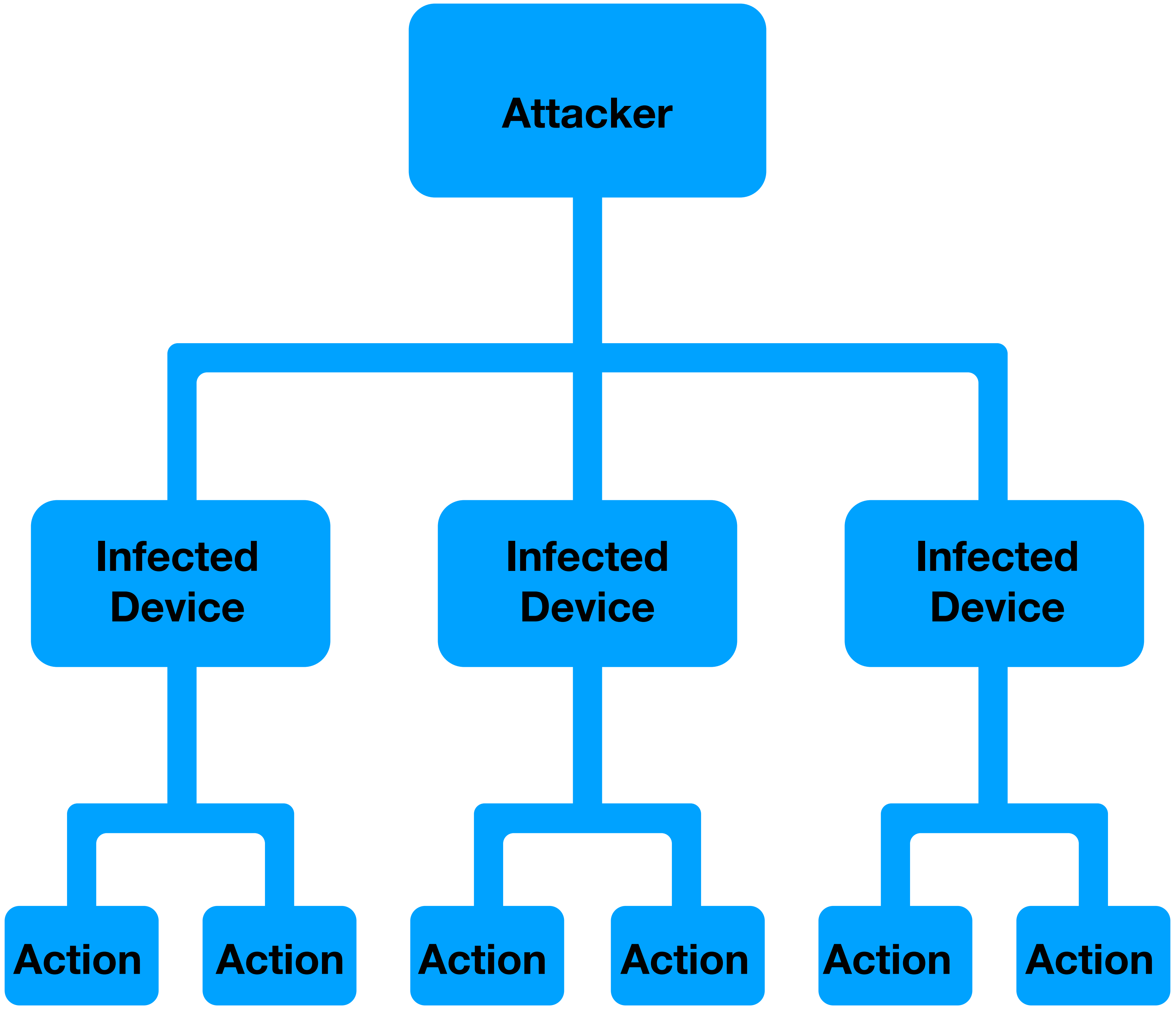
- This type of malware is not malicious by itself, however it serves a malicious purpose
- It is used to hide the real payload or malware
- Includes verification before downloading the malicious code to avoid being detected: Is it a VM? Is this country my target? Is this an outdated machine?
- Makes it more difficult for analysts to analyze the real sample
- Examples: Dofoil

Banking trojans

- The main purpose of this type of malware is stealing bank-related credentials
- Uses different yet known injection techniques to achieve their goal
- Hooks browser functionality to inject code into a reduced set of websites (only bank login websites)
- Complex infrastructure, multiple endpoints to avoid being taken down
- Examples: ZeuS,

RATs / Stealers

- Focused on stealing user’s credentials, similar but not equal to banking trojans
- Focuses on dumping passwords and login details from as much applications as possible in the target computer: Chrome, Firefox, FileZilla…
- May include key logging capabilities to monitor login details.
- Usually includes file-system access, VNC or other capabilities.
- Examples: DarkComet, njRAT


Attacker
Infected Device
Infected Device
Infected Device
Action
Action
Action
Action
Action
Action

Main differences between RATs and botnets

- Botnets focus on infecting as many machines as possible, while RATs infection count is way lower.
- Botnets usually do mass actions, while RATs focus on per-victim actions.
- RATs interaction with the victim is closer, allowing for more control of the machine
- RATs are mainly used for targeted attacks

Ransomware

- Focuses on using a ransom as a means of retrieving money
- In the recent years, it focuses on encrypting the user’s valuable documents using a strong and safe encryption (at early versions, they make implementation mistakes)
- To avoid being detected on an early stage, it focuses on valuable file extensions only: doc, docx, png, jpeg…
- Attackers request a payment of a large sum of money depending on the country, in order to get a decryption tool.
- Examples: WannaCry, Petya, NotPetya

Botnets

- Focuses on executing repeated actions on many computers at the same time: Port Scanning, DDoS…
- Therefore, the main purpose of this malware is to spread as much as possible during early stages.
- May include worm-like behavior to expand its range
- While RATs focus on per-victim management, botnets perform collective actions (control multiple machines at the same time)
- Examples: Mirai, Satori…

Persistence mechanisms

Discussing commonly persistence mechanisms of malware

Persistence mechanisms

- The first stage of a malware infection is essentially gaining access to the system (Spam campaigns, targeted attacks, lateral movement)
- Attacker seeks to remain on the system as long as possible. For example, after a user reboots the computer.
- If the persistence mechanism is specific enough, we can fingerprint it easily
- It can be used as a stealth mechanism. For example, waiting until a certain day, wait until certain conditions are met…

Windows registry

- The windows registry allows for different ways to run on boot, in this section we will list some of them
- The most common registry key used is:

  `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

- There are other registry places for hiding, however **SysInternals** tools such as outruns already do this job for us.
- Other notable mentions include APPInit, Winlogon

AppInit_DLLs

- Malware authors can gain persistence for their DDLs in using a special registry location
- They are loaded into every process that loads User32.dll
- **HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows**
- REG_SZ values of space-delimited strings of DLLs.
- Due to the huge quantity of processes loading user32.dll, current process checks are required. Usually @ DllMain

Winlogon

- This technique can allow attackers to run even in Safe mode.
- Malware authors hook certain events from winlogin, such as logon, logoff, startup…
- The registry key used uses the Notify value, a DLL will handle the event
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

SvcHost

- Svchost.exe is a generic host process for services that run from DLLs
- Systems have multiple instances of svchost.exe running on a non-infected machine
- They are blended with the system, harder for the average Joe to spot.
- To avoid detection, it doesn’t create a new service group and uses instead already available ones.
- We can detect this by hooking CreateService() or monitoring the registry.

- Groups defined at

  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost

- Services defined at

  HKLM\SYSTEM\CurrentControlSet\Services\ServiceName

Patching system binaries

Original system DLL

```
DllEntryPoint(HInstance hinstDLL, DWORD
     fdwReason, LPVOID lpReserved)

              mov edi, edi
                push ebp
              mov ebp, esp
                push ebx
            mov ebx, [ebp+4]
```

Modified system DLL

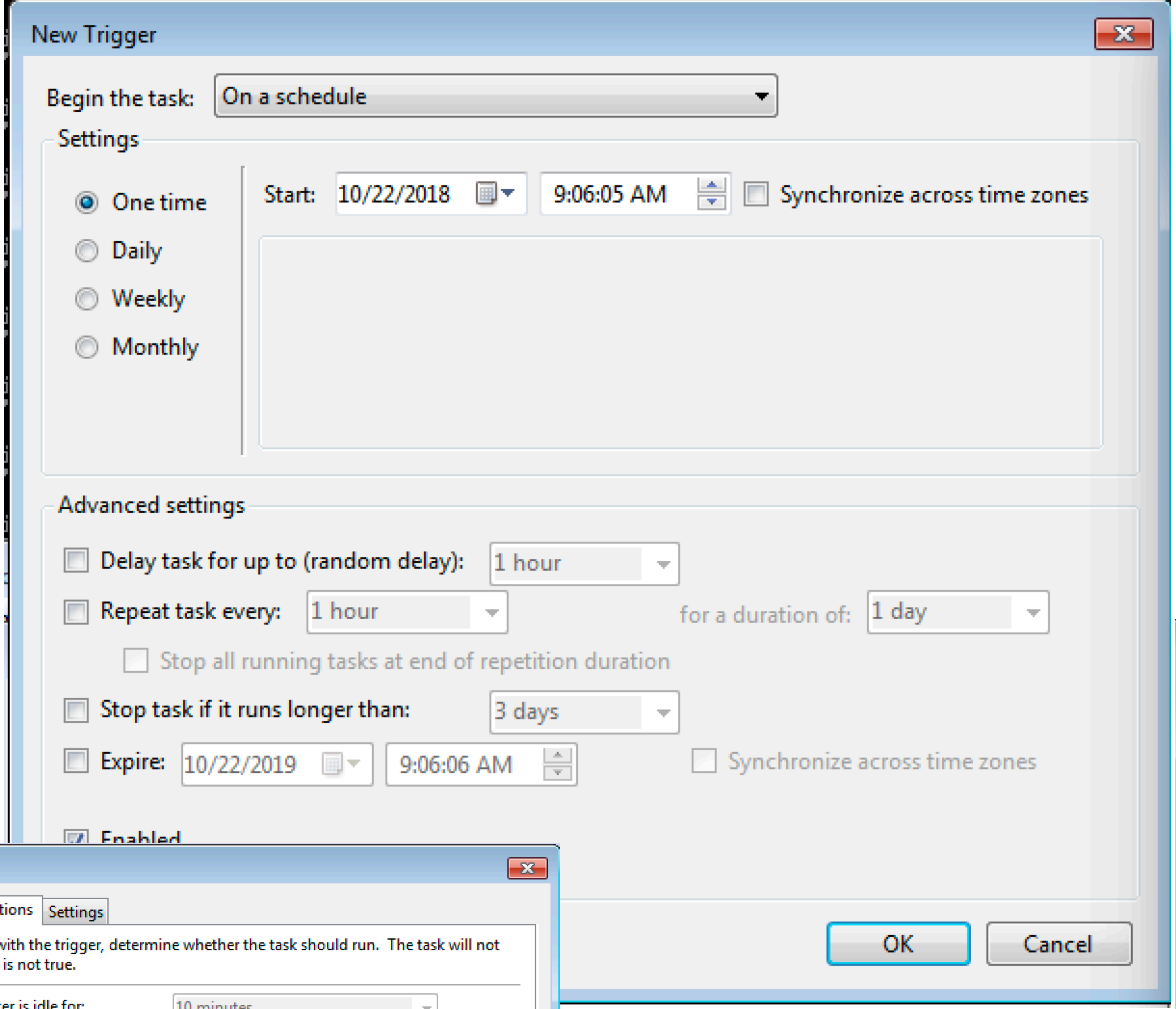
```
DllEntryPoint(HInstance hinstDLL, DWORD
     fdwReason, LPVOID lpReserved)

          jmp DllEntryPoint_0
```

DllEntryPoint_0 contains jumps to the malicious code.

Task scheduler

- Windows has a native service to set task executions
- This allows for certain actions to be executed by configuring an item on this tool
- Allows for binaries or script execution on events, such as start-up, logon, certain hour…
- Commonly used by TrickBot to ensure persistence on the target system.



New Trigger

Begin the task: On a schedule

Settings

(●) One time
( ) Daily
( ) Weekly
( ) Monthly

Start: 10/22/2018 9:06:05 AM [ ] Synchronize across time zones

Advanced settings

[ ] Delay task for up to (random delay): 1 hour

[ ] Repeat task every: 1 hour for a duration of: 1 day

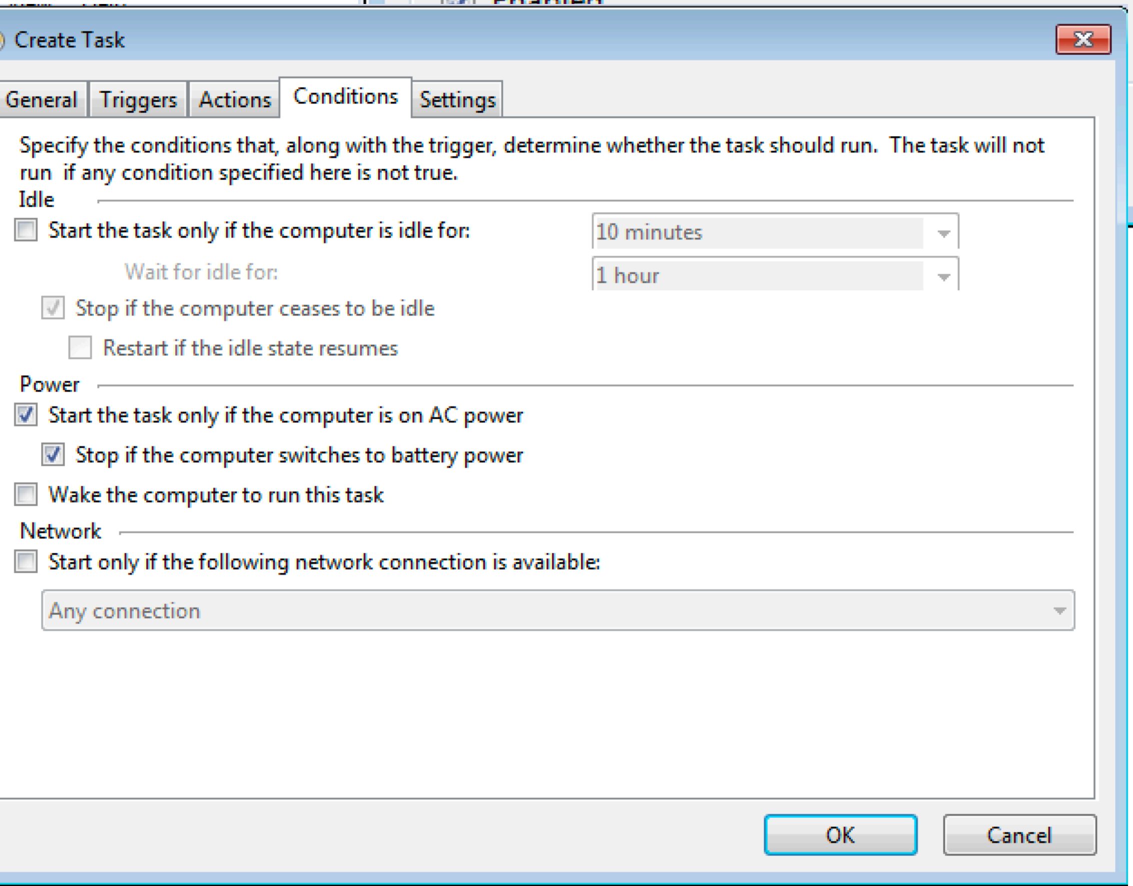
[ ] Stop all running tasks at end of repetition duration

[ ] Stop task if it runs longer than: 3 days

[ ] Expire: 10/22/2019 9:06:06 AM [ ] Synchronize across time zones

[✓] Enabled

OK Cancel



Create Task

General | Triggers | Actions | Conditions | Settings

Specify the conditions that, along with the trigger, determine whether the task should run. The task will not run if any condition specified here is not true.

Idle

[ ] Start the task only if the computer is idle for: 10 minutes

Wait for idle for: 1 hour

[✓] Stop if the computer ceases to be idle

[ ] Restart if the idle state resumes

Power

[✓] Start the task only if the computer is on AC power

[✓] Stop if the computer switches to battery power

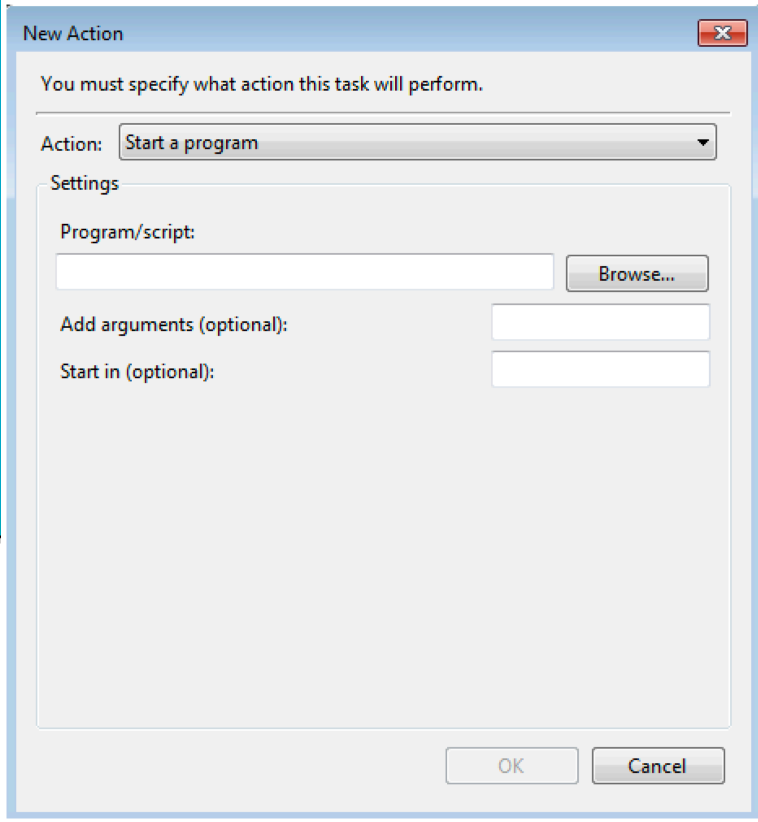
[ ] Wake the computer to run this task

Network

[ ] Start only if the following network connection is available:

Any connection

OK Cancel



New Action

You must specify what action this task will perform.

Action: Start a program

Settings

Program/script:

Browse...

Add arguments (optional):

Start in (optional):

OK Cancel

Common launching techniques

Anti-VM techniques

Because malware does not simple run

Why is it present?

- Sophisticated malware usually gets the attention of researchers and big companies
- Researchers don’t tend to run their samples on bare metal hardware, but rather virtual machines
- If researchers cannot gain useful information from the static analysis, they will execute the sample
- If there is no execution, it’s difficult to figure out what is going on

- There are various papers published regarding anti-virtualization techniques
- There are also some papers that mention how to counter some of these anti-virtualization techniques
- There are also tools that aggregate these most common techniques into one single binary: pafish, al-khaser…
- Even inexpert malware writers can open the repository of these samples and paste these techniques into their sample

- As a matter of fact, it’s impossible to hide our VM from attackers, the more exposed we are the harder attackers will try to evade us

- Most spread platforms have more literature on the wild: VirtualBox, VMWare…

- Even public sandboxes with custom set-ups end up being detected

Two detection groups

- Generic group: Features or traces that are found commonly between different sandboxes.
- Specific group: Target the detection of an specific sandbox software or virtualization product.

Detection via registry

- Registry keys allow for specific software detection
- VirtualBox and VMWare contain very specific registry keys that allow for a quick and easy detection
- Some can be avoided, some others require of on-boot deletions. (They are added again on reboot)

Some example registry keys…

- HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0
- HARDWARE\\Description\\System\\SystemBiosVersion\\VBOX
- HARDWARE\\Description\\System\\SystemBiosVersion\\VIRTUALBOX
- SOFTWARE\\Oracle\\VirtualBox Guest Additions

- HARDWARE\\ACPI\\DSDT\\VBOX__
- HARDWARE\\ACPI\\FADT\\VBOX__
- HARDWARE\\ACPI\\RSDT\\VBOX__

- SOFTWARE\\VMware, Inc.\\VMware Tools
- HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0\\Identifier\\VMWARE
- For a complete list of registry keys, refer to the registry_keys.md document found this module materials

Conclusions

As we may have noticed, the list of common registry keys is wider on VirtualBox than on VMWARE. However, that doesn’t mean that VMWare is harder to detect at all.

There are other methods for detecting specific sandboxes that don’t involve the use of registry keys at all

File-system inspection

- Virtualization software generally requires of some files to run properly as guest systems.
- These files are invisible to the user, but they are present to improve the user experience
- Guest tooling software inserts even more files that are easy to identify.

Generic sandbox detection

Sandboxes generally share some common points between them all.

Researchers don’t usually have access to mass-scale hardware for analysis. Therefore, the hardware used inside Virtual Machines is easy to identify

We will now expose some of these common points

Generic sandbox detection

- **Username checks**: Usually, malware checks for known sandbox usernames or generic usernames. These include john, user, malware… Or public sandboxes Windows usernames
- **File-paths**: Known execution file paths are checked. These include C:/sample.exe, C:/malware.exe, %appdata%/malware.exe …
- **Mouse activity**: There is no interaction from the user during mass scale analysis, therefore some actions such as mouse activity aren’t performed. For this reason, it’s used as an indicator of sandbox awareness

Generic sandbox evasion

- **Physical memory**: In order to run multiple sandboxes, RAM is limited. In case that we assign low RAM ranges to our sandbox, we will probably be detected. Malware usually checks for **RAMs lower than 2Gb or 1Gb**

- **Hard drive sizes:** Virtual machines are also limited in the disk allowance. For example, in case that your device has 200Gb of free disk space, you would need to allow at least 50Gb per VM (for ONLY 4 VMs). Malware usually checks for **HDD sizes lower than 60Gb or 120Gb**.

Generic sandbox detection

- **Processor capacity**: The devices that users have in their hands nowadays usually have at least 2 cores, and most of them 4 cores at least. High-end computers are getting to 8 cores.
  Malware wants to execute in real users machines, not controlled environments. Hence, it will check for a **minimum of 2 cores up to 4**

- **Filenames**: sample.exe, malware.exe, <sha256, md5, sha1>.exe - Are expected by malware authors, and they might not run under these conditions.

Generic sandbox detection, Time based attacks.

Malware authors assume that their time inside a controlled environment is limited. In case that we want to search for something between millions of samples, we don’t have time for all of them.

For this purpose time-delay attacks are implemented.

These attacks including introducing long Sleep()’s, RDTSC, SetTimer, timeSetEvent, WaitForSingleObject, WaitForMultipleObjects…

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.363.6295&rep=rep1&type=pdf

Generic sandbox detection

- System uptime: Controlled environments generally have a lower up-time values. They are not running for long periods of time (minutes, hours, days).
  Malware will check for low uptime values to prevent execution. **GetTickCount()** API can be used to check this.

  Low uptime such as 15 minutes and 25 minutes will likely get noticed by malware authors.

Virtualization dependent sandbox evasion

Generic Sandbox evasion, CPUID hypervisor feature bits

- It’s possible to track if the current environment that the sample is running in is inside a known hypervisor. Malware attackers use these known IDs to prevent execution.

  Some of these known bits include:

```
KVMKVMKVM\0\0\0 // KVM
VBoxVBoxVBox // VirtualBox
Microsoft Hv // Microsoft Hyper-V
VMwareVMware // VMware
XenVMMXenVMM // Xen
prl hyperv // Parallels
```

Running processes

Virtualization software runs specific processes to improve the user experience and manage certain features such as Copy/Pasting, Drag&Drop, Shared folders…

However, the drawback to having this software installed is that it will install several files, multiple registry keys will be inserted and extra processes will be spawned.

Examples of this software:

VirtualBox Guest Additions
VMWare Tools

If **VirtualBox Guest Additions** is installed or **VMWare tools**, masquerading our environment will become nearly impossible.

For a list of known running processes, please refer to **running_processes.md** document.

File-System files

Virtualization software contains drivers and DLLs that makes our user experience better while operating with the virtual machine. We will be able to get higher resolutions, better 2D/3D experience, mouse and keyboard integrations…

However, this also makes our machine easily detected.

For **VirtualBox**, several *vbox**, *VBox** **.sys** files will be created on our system32 folder.

For **VMWare**, plenty of *vm** **.sys** files fill be dropped in our system32/drivers folder.
For a complete list of known artifacts, please refer to the **filesystem.md** document.

Networking

Virtual machines have the possibility of connecting to the internet. However, it’s possible to trace known virtualization software by its **MAC Addresses**.

Since these addresses follow a pattern, it’s possible to trace them.

`08:00:27`	**VirtualBox**
`00:05:69`	**VMWare**
`00:0C:29`	**VMWare**
`00:1C:14`	**VMWare**
`00:50:56`	**VMWare**
`00:16:3E`	**Xen**

Installed Virtual Devices

There are known virtual devices that make it easy for malware authors to detect our system. In order to check them, they need to get a valid handler to them.

Some of these devices include:

```
\\.\VBoxGuest // VirtualBox
\\.\VBoxTrayIPC // VirtualBox
\\.\VBoxMiniRdrDn // VirtualBox
\\.\vmci // VMWare
\\.\HGFS // VMware
```

For a list, please refer to **vdevices.md**

Sandboxing software detection

It’s possible to fingerprint specific sandboxing products. For example, one of the most deployed world-wide sandboxes is **Cuckoo Sandbox**. This is an open-source software that allows for a quick deployment of a dynamic analysis cluster.

There are open-source tools such as anticuckoo whose objective is to detect whether it’s being executed by a cuckoo analyzer / environment and either don’t run or crash cuckoo.

For commercial products, there are some specific detections such has HA, JoeSandbox… But they are not included in an specific OSS tool.

Firmware detection

Virtualization software comes with emulated hardware. These include but not limited to network adapters, sound cards and videos (VGA)

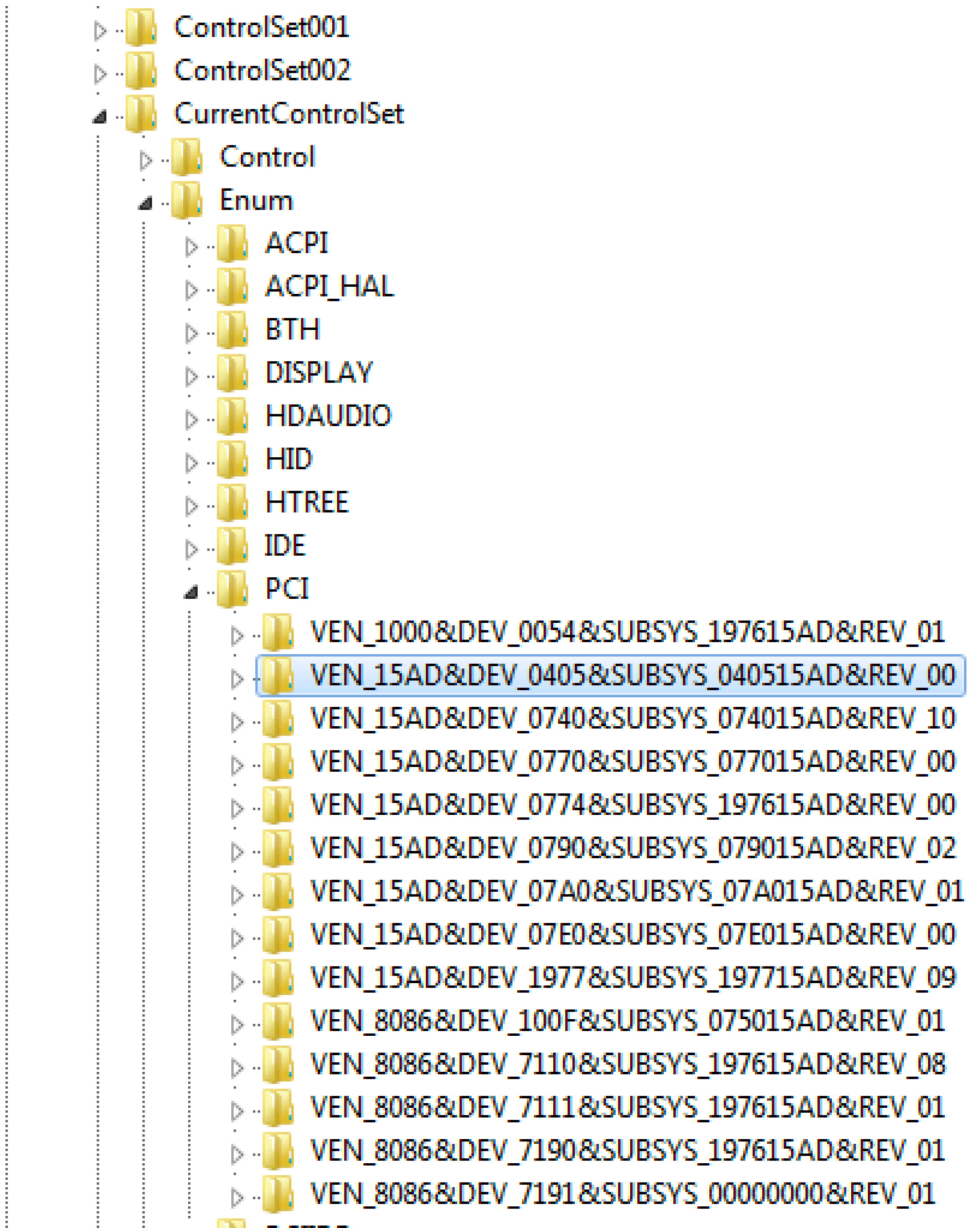
Even if we don’t install software such as Guest Additions, some if these will remain on the system because they are required, such as the VGA display emulation.

With the right set of queries, it’s possible to detect unique strings in the firmware that expose the current virtualization software.

It’s possible to avoid firmware dumping by queuing a registry key.

```
HKLM\\SYSTEM\\\
\CurrentControlSet\\Enum\
\\PCI
```

For more detailed information, refer to the following resource: https://www.heise.de/security/downloads/07/1/1/8/3/5/5/9/vmde.pdf



```
ControlSet001
ControlSet002
CurrentControlSet
    Control
    Enum
        ACPI
        ACPI_HAL
        BTH
        DISPLAY
        HDAUDIO
        HID
        HTREE
        IDE
        PCI
            VEN_1000&DEV_0054&SUBSYS_197615AD&REV_01
            VEN_15AD&DEV_0405&SUBSYS_040515AD&REV_00
            VEN_15AD&DEV_0740&SUBSYS_074015AD&REV_10
            VEN_15AD&DEV_0770&SUBSYS_077015AD&REV_00
            VEN_15AD&DEV_0774&SUBSYS_197615AD&REV_00
            VEN_15AD&DEV_0790&SUBSYS_079015AD&REV_02
            VEN_15AD&DEV_07A0&SUBSYS_07A015AD&REV_01
            VEN_15AD&DEV_07E0&SUBSYS_07E015AD&REV_00
            VEN_15AD&DEV_1977&SUBSYS_197715AD&REV_09
            VEN_8086&DEV_100F&SUBSYS_075015AD&REV_01
            VEN_8086&DEV_7110&SUBSYS_197615AD&REV_08
            VEN_8086&DEV_7111&SUBSYS_197615AD&REV_01
            VEN_8086&DEV_7190&SUBSYS_197615AD&REV_01
            VEN_8086&DEV_7191&SUBSYS_00000000&REV_01
```

情報セキュリティ国際会議
CODE BLUE

Researcher tools

Researchers commonly use virtual machines for malware inspection, therefore they posses an image with the necessary tooling for analysis.

Some of these tools are **IDA Pro, Wireshark, FileMon, RegMon, ProcMon, OllyDbg, ProcessExplorer**…

In case they are found running either in the system or installed, it might also prevent execution.

Interesting APIs

Interesting APIs

During this section we will cover different interesting Windows APIs used by Malware.

The ones listed in the slides will not be limited to all the possible APIs.

It's interested to know that, generally ANSI versions will end up calling its UNICODE counter-part.

For more information, refer to: https://doxygen.reactos.org/de/de3/dll_2win32_2kernel32_2client_2loader_8c_source.html

CreateMutex

- This API is also available in its UNICODE version and ANSI version

CreateService

- This API is also available in its UNICODE version and ANSI version
- It will allow for creation of new services. It’s generally used as a persistence method, since they can be either controlled by ControlService and because they are able to run on boot

ControlService

- This API is also available in its UNICODE version and ANSI version
- It’s used to control the installed services. Between the commands available, we have **modify**, **start** and **stop**

CreateRemoteThread

- This API is also available in its UNICODE version and ANSI version
- Usually used for code injection into other processes. Its purpose is to start a thread in a suspended process

GetAdaptersInfo

- This API call allows us to get information about the available network adapters on the current device.
- As we have mentioned before, malware checks for known MAC Addresses and GetAdaptersInfo allows their check
- It’s also a means of fingerprinting the machines where malware has already executed.

CreateProcess

- This API is also available in its UNICODE version and ANSI version
- It’s used to launch a new processes. Usually, malware consists of multiple-stage payloads.
- An undocumented counterpart exists, called CreateProcessInternalW

FindWindow

- This function is able to search for open windows on our desktop. This is commonly used to see if a current target process is open, such as chrome.exe and firefox.exe
- It’s also able to search for known tools such as IDA, RegMon, Wireshark…

CreateFile

- CreateFile is a Windows API that is present in the unicode version and the ANSI version.
- This API opens or creates a new file and returns a handler to it

GetAsyncKeyState

- This API is commonly abused by keyloggers to get the information about the user’s keystrokes.
- It allows to know whether an specific key is being pressed.
- It’s usually combined with GetKeyState to get the status of an specific key.

GetProcAddress

- This function is commonly abused by malware. The purpose of this API is to obtain the function address from an already known DLL.
- It allows to import functions from different DLLs, for example we can get the address of **kernel32!CreateFilew**

GetTickCount

- It’s called by malware to be used as an anti-analysis technique.
- It returns the elapsed milliseconds since startup
- If this function is called by malware, low return values might tell the attackers that the system is running under a controlled environment and **might not execute**.

IsDebuggerPresent

- This API checks whether the current running process is being debugged or not.
- In case that the application is being debugged, malware might not execute or allow itself to be debugged.

RegOpenKey

- As we have seen before, registry is usually used as a means of persistence for malware. It also allows for checks about the current devices or installed software.
- It returns a handler of a registry key which can later bemused for other operations.

NtQuerySystemIInformation

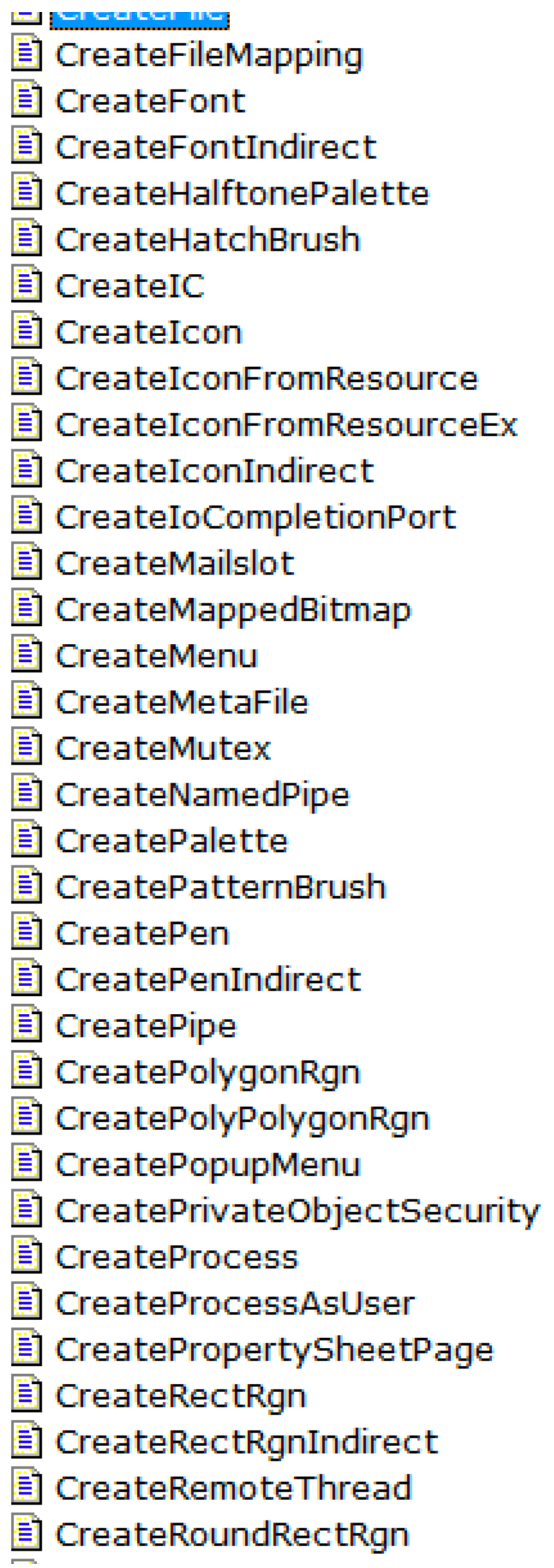
- One of the classes from is SystemProcessInformation, 0x5
- It’s used for getting listings of the current running processes (taskmgr, process explorer…)
- This API can be abused to hide processes from the list by iterating the list of processes.

More interesting APIs…

During this part, we have only listed some of the important APIs.

For a complete list of interesting API calls, it’s recommended to read section A from PMA

For detailed APIs description (purpose, arguments, return values…) please refer to win32.chm



CreateFileMapping
CreateFont
CreateFontIndirect
CreateHalftonePalette
CreateHatchBrush
CreateIC
CreateIcon
CreateIconFromResource
CreateIconFromResourceEx
CreateIconIndirect
CreateIoCompletionPort
CreateMailslot
CreateMappedBitmap
CreateMenu
CreateMetaFile
CreateMutex
CreateNamedPipe
CreatePalette
CreatePatternBrush
CreatePen
CreatePenIndirect
CreatePipe
CreatePolygonRgn
CreatePolyPolygonRgn
CreatePopupMenu
CreatePrivateObjectSecurity
CreateProcess
CreateProcessAsUser
CreatePropertySheetPage
CreateRectRgn
CreateRectRgnIndirect
CreateRemoteThread
CreateRoundRectRgn