Malware Persistence Methods

Persistence

- Adversaries want their malicious program to stay on the compromised computers even when the Windows restarts.

- Persistence allows malware to remain active in the event of a reboot.

- The persistence allows the attacker to remain on the compromised system without having to reinfect it.

Run Registry Key

- One of the most common persistence mechanism is achieved by adding an entry to the ***run registry keys***.

- The program that is added to the run registry key gets executed at system startup. Malware may add itself into various auto-start locations, the following is the list of the most common run registry keys.

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
```

Scheduled Tasks

- Adversaries can schedule a task which allows them to execute their malicious program at a specified time or during system startup.
- Windows utilities such as ***schtasks*** and ***at*** are generally used to schedule a program or script to execute at desired date and time.

Demo 2 - Scheduled Tasks
(ssub.exe)

Startup Folder

- Adversaries can achieve persistence by adding their malicious binary in the startup folders.

- When the operating system starts, the startup folder is looked up and files residing in this folder are executed.

- Windows operating system maintains two types of startup folders ***(a) user wide*** and ***(b) system-wide***

- A program residing in user's startup folder is executed only for a specific user and the program residing in system folder is executed when any user logs on to the system. Administrator privilege is required to achieve persistence using system-wide startup folder.

```
C:\%AppData%\Microsoft\Windows\Start Menu\Programs\Startup
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
```

Lab 3 - The case of Backdoor.Nitol

During your investigation of a suspect system, you come across a malicious file (***bllb.exe***). Analyze this file and answer the following questions:

1. What is the name of the file dropped by the malware?
2. Which C2 domains are contacted by the malware?
3. What do you see in the network traffic?
4. How is the malware persisting on the system?

Image File Execution Options (IFEO)

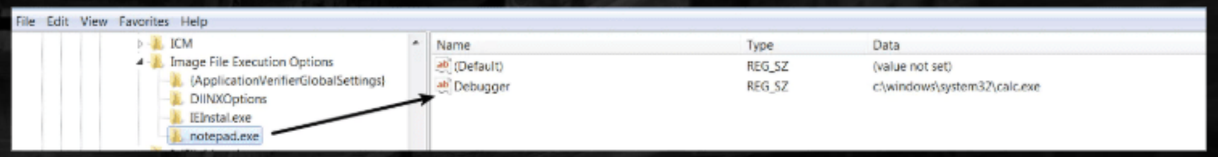
- **Image File Execution Options (IFEO)** allows one to launch an executable directly under the debugger. It gives the developer the option to debug their software to investigate issues in the executable's startup code.

- Developer can create a sub-key with the name of his executable under the below registry key and set the debugger value to the path of the debugger.

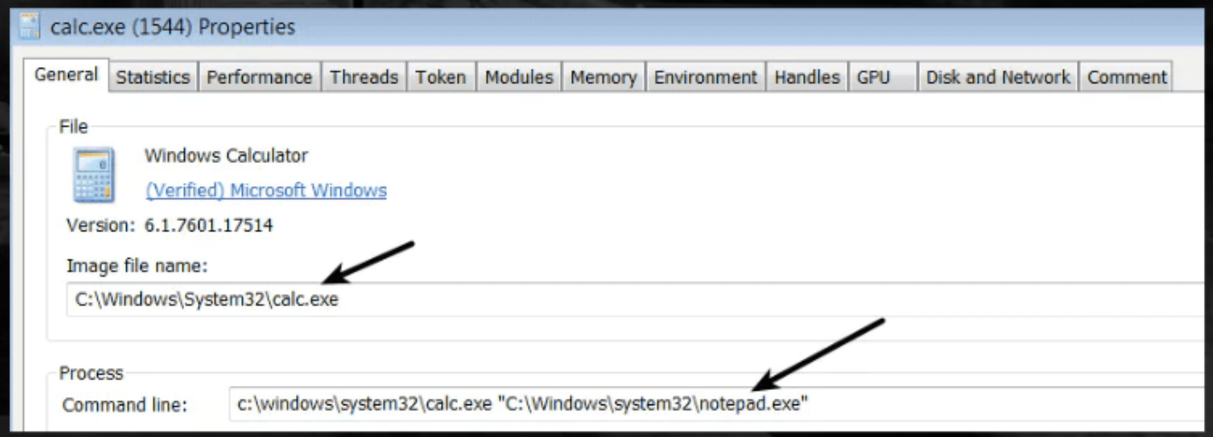
```
Key:'HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\<executable name>'

Value: Debugger : REG_SZ : <full-path to the debugger>
```

- Adversaries take advantage of this registry key to launch their malicious program.

- To demonstrate this technique, the debugger for the ***notepad.exe*** is set to ***calculator*** by adding the following registry entry.



- Now when you start notepad, it will be launched by calculator program (even though it is not a debugger), this behavior can be seen in the below screenshot.



DEMO 3 - IFEO

(dhic.exe)

Lab 4 - The case of Trojan Small

During your investigation of an infected system, you come across a file (***bas.exe***). Analyze this file and answer the following questions:

1. What is the name of the file dropped by the malware?
2. What is the full path on the disk where the file is dropped?
3. How is malware persisting on the system?
4. Can you open the registry editor (***regedit.exe***) and identify the entry added by the malware?
5. Can you spot the malicious process in the task manager (***taskmgr.exe***)?

Winlogon registry entries

- Attackers can achieve persistence by modifying the registry entries used by the ***Winlogon process***. Winlogon process is responsible for handling interactive user logons and logoffs.

- Once the user is authenticated, ***Winlogon.exe*** process launches ***Userinit.exe*** which runs logon scripts and re-establishes network connections. The ***Userinit.exe*** then starts explorer.exe which is the default User's shell.

- Winlogon process launches ***userinit.exe*** due to the below registry value. This entry specifies which programs need to be executed by Winlogon when a user logs on.

- By default, this value is set to the path of ***userinit.exe (C:\Windows\system32\userinit.exe)***. An attacker can change or add another value containing the path to the malicious executable which will then be launched by ***Winlogon*** process (when the user logs on)

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
or
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
```

- In the same manner, ***userinit.exe*** consults the below registry value to start the default User's shell.

- By default, this value is set to ***explorer.exe***. An attacker can change or add another entry containing the name of the malicious executable which will then be started by ***userinit.exe***

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
or
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
```

Lab 5 - The case of Brontok Worm

While investigating a suspect system for possible compromise, you find a suspect file (***tux.exe***). You also notice that when you try launching registry editor (***regedit.exe***) on the infected system, instead of registry editor notepad is invoked. Analyze the sample (tux.exe) and answer these questions:

1. Which executable files are created by the malware?
2. Apart from the entry added in the Run registry key, Which other registry entries are added by malware for persistence?
3. Why do you think notepad is launched, instead of registry editor? and which another legitimate application is targeted using the similar technique?

Accessibility programs

- Windows operating system provides various accessibility features such as the ***On-screen keyboard***, ***Narrator, Magnifier, Speech recognition***, etc.

- These accessibility programs can be launched without even logging into the system.

- Adversaries have been using sticky keys (***sethc.exe***) feature to gain unauthenticated access via ***Remote Desktop (RDP)***.

- For example, many of these accessibility programs can be accessed by pressing ***Windows + U*** key combination which launches ***C:\Windows\System32\utilman.exe*** or You can enable sticky keys by pressing shift key five times which will launch the program ***C:\Windows\System32\sethc.exe***.

- An attacker can modify the manner in which these accessibility programs (such as ***sethc.exe*** and ***utilman.exe***) are launched to execute the program of their choice or ***cmd.exe*** with elevated privileges (privilege escalation).

In the case of Hikit Rootkit, the legitimate ***sethc.exe*** program was replaced with ***cmd.exe***, this allowed the adversaries to access the command prompt with ***SYSTEM*** privileges over RDP just by pressing shift key ***five*** times. The newer versions of Windows enforces various restrictions which prevent replacing ***sethc.exe***.

To avoid replacing the files, adversaries make use of the ***Image File Execution Options*** registry. The following registry entry sets ***cmd.exe*** as the debugger for ***sethc.exe***, now an adversary can use RDP login and press ***SHIFT*** Key five times to get access to the System-level command shell. Using this shell, an adversary can execute any arbitrary commands even before authentication.

```
REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /t
REG_SZ /v Debugger /d "C:\windows\system32\cmd.exe" /f
```

Lab 6 - The Case of Trojan Bitrep

While you are investigating a suspect machine, you find the file (***mets.exe***). When this malicious program (***mets.exe***) is executed, it invokes cmd.exe to execute certain commands. Analyze mets.exe and answer the following questions:

1. What is the purpose of the commands executed via cmd.exe?
2. As a result of commands executed by the malware, what kind of capability it gives to the attacker?
3. Are there any network indicators that you use in network monitoring to identify the systems infected with this malware?

AppInit_DLLs

- The ***AppInit_DLLs*** feature in Windows provides a way to load custom DLLs into the address space of every interactive application.

- Once a DLL is loaded into the address space of any process, it can run within the context of that process and can hook well-known APIs to implement an alternate functionality.

- An Attacker can achieve persistence for their malicious DLL by setting the ***AppInit_DLLs*** value in the below registry key.

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

- ***AppInit_DLLs*** value typically contains space or comma-delimited list of DLLs.

- All the DLLs specified here are loaded into every process that loads ***User32.dll***. Since ***User32.dll*** is loaded by almost all the processes, this technique enables the attacker to load their malicious DLL into most of the processes and executes the malicious code within the context of the loaded process.

- In addition to setting the ***AppInit_DLLs*** value, an attacker may also enable the ***AppInit_DLLs*** functionality by setting ***LoadAppInit_DLLs*** registry value to ***1***.

- The ***AppInit_DLLs*** functionality is disabled on Windows 8 and later versions where the secure boot is enabled.

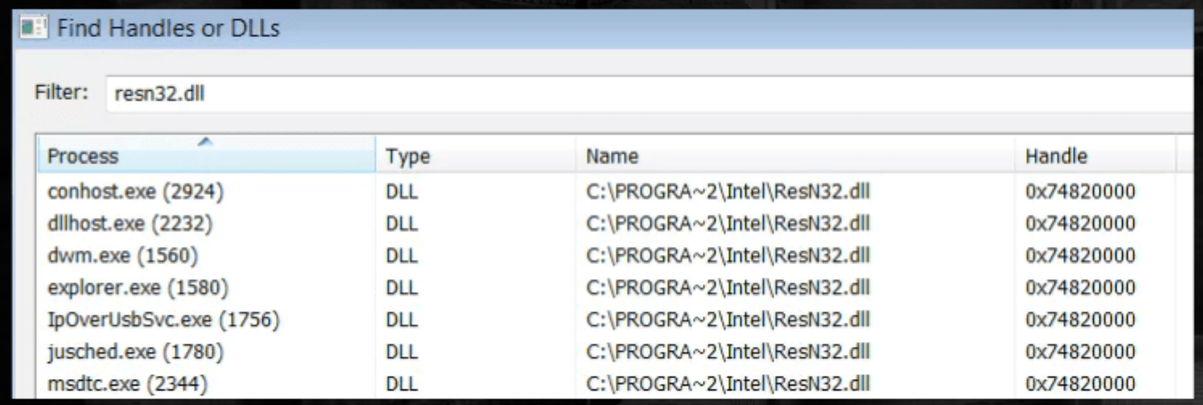
Demo 4 - T9000 Backdoor (AppInit_DLLs)
(legal.exe)

The following screenshot shows the AppInit DLL entries added by the ***T9000*** backdoor

AppInit_DLLs	REG_SZ	C:\PROGRA~2\Intel\ResN32.dll
DdeSendTimeout	REG_DWORD	0x00000000 (0)
DesktopHeapLogging	REG_DWORD	0x00000001 (1)
DeviceNotSelectedTimeout	REG_SZ	15
GDIProcessHandleQuota	REG_DWORD	0x00002710 (10000)
IconServiceLib	REG_SZ	IconCodecService.dll
LoadAppInit_DLLs	REG_DWORD	0x00000001 (1)

As a result of adding the above registry entries, when any new process (which loads ***User32.dll***) is started, it loads the malicious DLL (***ResN32.dll***) into its address space.

The following screenshot displays the operating system processes which loaded the malicious ***DLL (ResN32.dll)*** after rebooting the system. Since most of these processes run with high integrity levels, it allows an adversary to execute malicious code with elevated privileges.



- To detect this technique, you can look for the suspicious entries in the ***AppInit_DLLs*** registry value which does not relate to the legitimate programs in your environment.

- You can also look for any process exhibiting abnormal behavior due to the loading of the malicious DLL.

DLL Search Order Hijacking

- Windows operating system searches for the DLL to be loaded in a specific order in the predefined locations

- If any DLL is to be loaded, the operating system first checks if the DLL is already loaded in the memory, if yes, it uses the loaded DLL.

- If not it checks if the DLL is defined in the "***KnownDLLs***" registry key

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
```

- If the DLL to be loaded is in the list of "KnownDLLs" then the DLL is always loaded from the System32 directory.

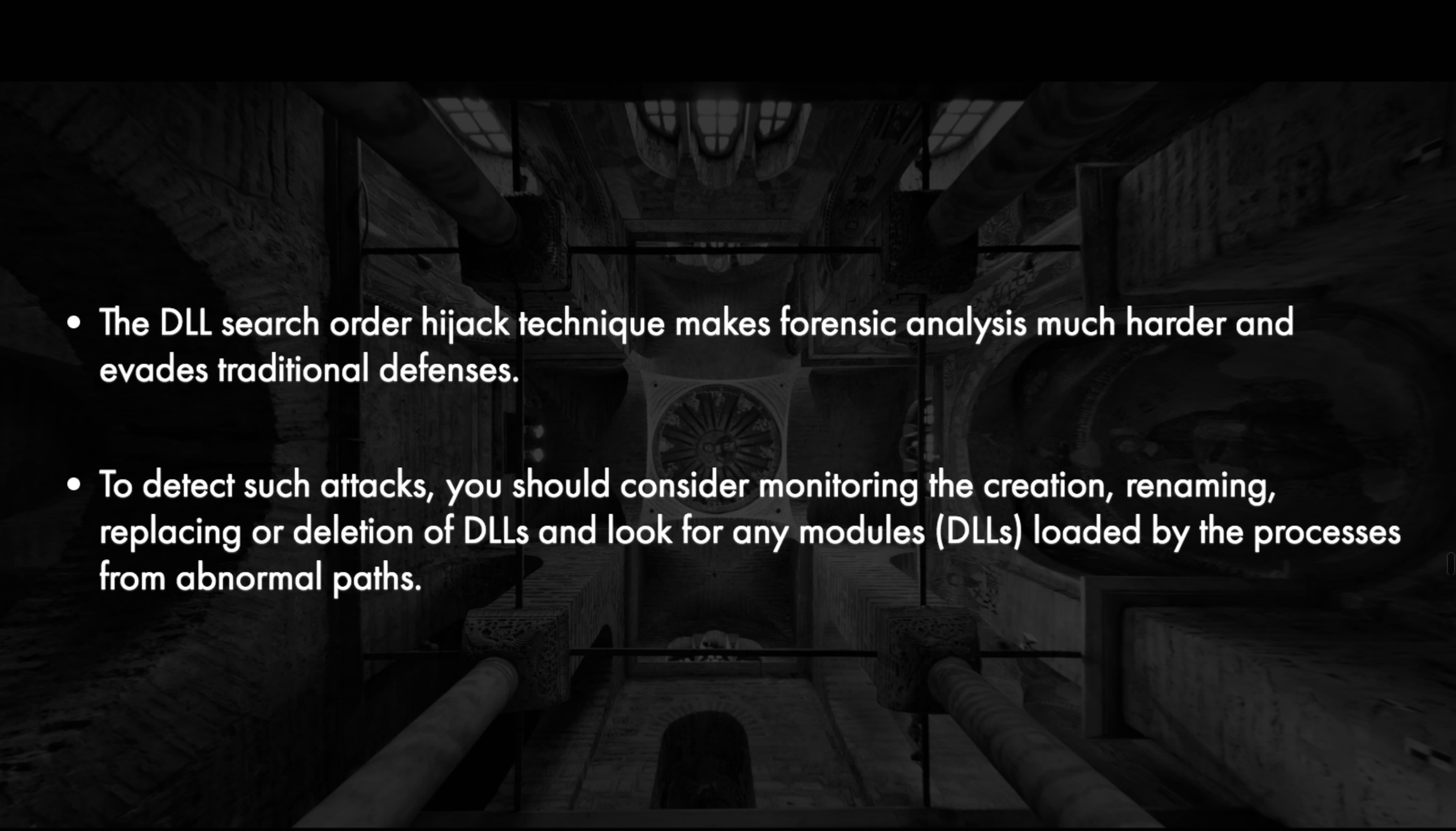
- If these conditions are not met then the operating system looks for the DLL in below locations in sequential order.

1. The directory from where the application was launched
2. The system directory (***C:\Windows\System32***)
3. The 16-bit system directory (***C:\Windows\System***)
4. The Windows directory (***C:\Windows***)
5. The current directory
6. Directories defined in the ***PATH*** variables

- Adversaries can take advantage of how the operating system searches for the DLL to escalate privilege and to achieve persistence

Demo 5 - DLL Search Order Hijacking

Operation Groundbait (toor.exe)



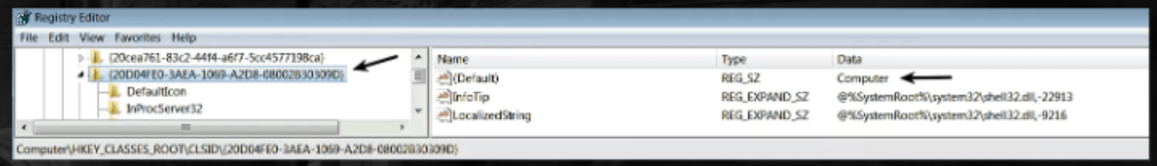
- The DLL search order hijack technique makes forensic analysis much harder and evades traditional defenses.
- To detect such attacks, you should consider monitoring the creation, renaming, replacing or deletion of DLLs and look for any modules (DLLs) loaded by the processes from abnormal paths.

COM Hijacking

- Component Object Model (COM) is a system that allows software components to interact and communicate with each other even if they have no knowledge of each other's code
- The software components interact with each other through the use of COM objects

- COM is implemented as a client/server framework. A COM client is a program that uses the service from the COM server (COM object) and a COM server is an object which provides service to the COM clients. The COM server implements interface consisting of various methods (functions) either in a DLL (called in-process servers) or in an EXE (called out-of-process server)

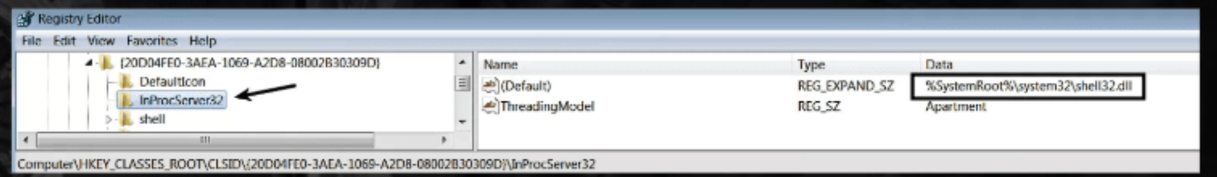
- COM Objects are identified by a unique number called class identifiers (CLSIDs) and they can be found in the registry key ***HKEY_CLASSES_ROOT\CLSID\<unique clsid>***



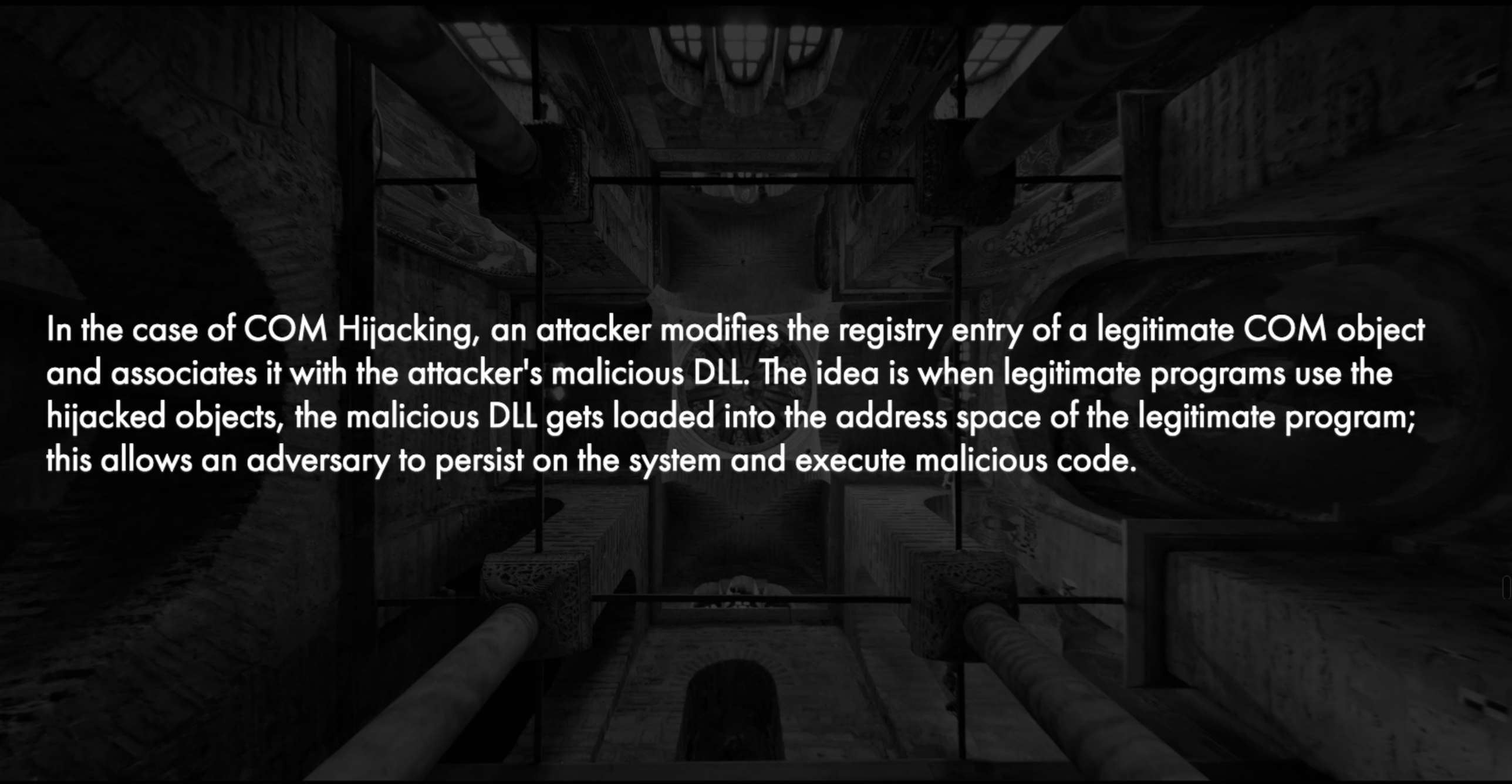
- For example, the COM object for My Computer is ***{20d04fe0-3aea-1069-a2d8-08002b30309d}*** which can be seen in the below screenshot.

For each CLSID key, you also have a sub key called ***InProcServer32*** which specifies the filename of the DLL that implements the COM server functionality.

- The following screenshot tells you that ***shell32.dll*** (COM server) is associated with My computer



- When the legitimate program (COM client) uses the service from a specific COM object (using its CLSID), its associated DLL gets loaded into the process address space of the client program.



In the case of COM Hijacking, an attacker modifies the registry entry of a legitimate COM object and associates it with the attacker's malicious DLL. The idea is when legitimate programs use the hijacked objects, the malicious DLL gets loaded into the address space of the legitimate program; this allows an adversary to persist on the system and execute malicious code.

Demo 6 - COM Hijacking

Trojan.Compfun (ions.exe)

Service

- A service is a program that runs in the background without any user interface
- An adversary with Administrator privilege can persist on the system by installing the malicious program as a service or by modifying an existing service
- An attacker may implement the malicious program as an EXE, DLL or kernel driver and run as service. Windows supports various service types.

- The following outlines some of the common service types used by the malicious programs.

1. ***Win32OwnProcess***: The code for the service is implemented as an executable, and it runs as an individual process.
2. ***Win32ShareProcess:*** The code for the service is implemented as a DLL, and it runs from a shared host process ***(svchost.exe)***
3. ***Kernel Driver Service:*** This type of service is implemented in a driver (.sys), and it is used to execute the code in kernel space.

- Windows stores the list of installed services and their configuration in the registry under ***HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services*** key.

Example 1 - Trojan Inject

In the following example, malware executes **sc** command (via **cmd.exe**) to create and start a service named update.

```
[CreateProcess] update.exe:3948 > "%WinDir%\System32\cmd.exe /c sc create
update binPath= C:\malware\update.exe start= auto && sc start update "
```

```
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\update\Type  =  16
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\update\Start  =  2
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\update\ErrorControl  =  1
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\update\ImagePath  =  C:\malware\update.exe
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\update\ObjectName  =  LocalSystem
```

You can query the service configuration using the ***sc*** utility by providing the service name, as shown here

```
C:\>sc qc update
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: update
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Users\training\Desktop\update.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : update
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

Example 2 - NetTraveler malware

Following example of the NetTraveler malware,
Upon execution, it first drops a dll

```
[CreateFile] d3a.exe:2904 > %WinDir%\System32\FastUser
SwitchingCompatibilityex.dll
```

After that, it drops and executes a batch script

```
@echo off

@reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
FastUserSwitchingCompatibility\Parameters" /v
ServiceDll /t REG_EXPAND_SZ /d
C:\Windows\system32\FastUserSwitchingCompatibilityex.dll
```

The following shows registry entries added with service configuration information

```
C:\>sc qc FastUserSwitchingCompatibility
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: FastUserSwitchingCompatibility
        TYPE               : 120  WIN32_SHARE_PROCESS (interactive)
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Windows\System32\svchost.exe -k netsvcs
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : FastUserSwitchingCompatibility
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

```
C:\>reg query "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FastUserSwitchingCompatibility\Parameters"

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FastUserSwitchingCompatibility\Parameters
    ServiceDll    REG_EXPAND_SZ    C:\Windows\system32\FastUserSwitchingCompatibilityex.dll
```

Hijacking Service

- Instead of creating a new service, an adversary can modify (hijack) the existing service
- Normally an attacker hijacks the service that is unused or disabled. This makes the detection slightly harder because if you are trying to find the non-standard or unrecognized service you will miss this type of attack.

Example 3 - BlackEnergy

Consider the example of BlackEnergy malware dropper which ***Hijacks*** the existing service to persist on the system, upon execution it replaces a legitimate driver ***aliide.sys*** (associated with service named "aliide") residing in ***system32\drivers*** directory with the malicious ***aliide.sys*** driver. After replacing the driver, it modifies the registry entry associated with the ***aliide*** service and sets it to autostart (service starts automatically when the system starts) as shown in the below events

```
[CreateFile] big.exe:4004 > %WinDir%\System32\drivers\aliide.sys
[RegSetValue] services.exe:504 > HKLM\System\CurrentControlSet\services\aliide\Start  =  2
```

The following screenshot shows the service configuration of "aliide" service before and after modification.

```
C:\>sc qc "aliide"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: aliide
        TYPE               : 1  KERNEL_DRIVER
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 3   CRITICAL
        BINARY_PATH_NAME   : \SystemRoot\system32\drivers\aliide.sys
        LOAD_ORDER_GROUP   : System Bus Extender
        TAG                : 0
        DISPLAY_NAME       : aliide
        DEPENDENCIES       :
        SERVICE_START_NAME :
```

Legitimate Driver

Before

```
C:\>sc qc "aliide"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: aliide
        TYPE               : 1  KERNEL_DRIVER
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 3   CRITICAL
        BINARY_PATH_NAME   : \SystemRoot\system32\drivers\aliide.sys
        LOAD_ORDER_GROUP   : System Bus Extender
        TAG                : 0
        DISPLAY_NAME       : aliide
        DEPENDENCIES       :
        SERVICE_START_NAME :
```

Replaced Malicious Driver

After