Press Play to Restart: Under the Hood of the Restart Manager

Mathilde Venault - REcon 2023

©2023 CROWDSTRIKE, INC. ALL RIGHTS RESERVED.

About me

> Security Researcher at CrowdStrike

> Ex-volunteer firefighter

> Previously talked at Black Hat & c0c0n

> c0c0n CFP/CFW review committee member

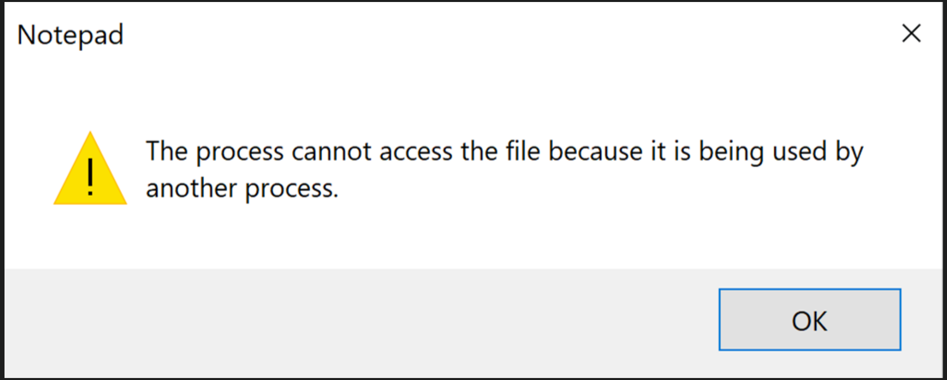
Introduction

> Default way of opening a file:

```
hFile = CreateFile(argv[1],                     // file to open
                   GENERIC_READ,                // open for reading
                   FILE_SHARE_READ,             // share for reading
                   NULL,                        // default security
                   OPEN_EXISTING,               // existing file only
                   FILE_FLAG_OVERLAPPED,        // overlapped operation
                   NULL);                       // no attr. template
```

Introduction

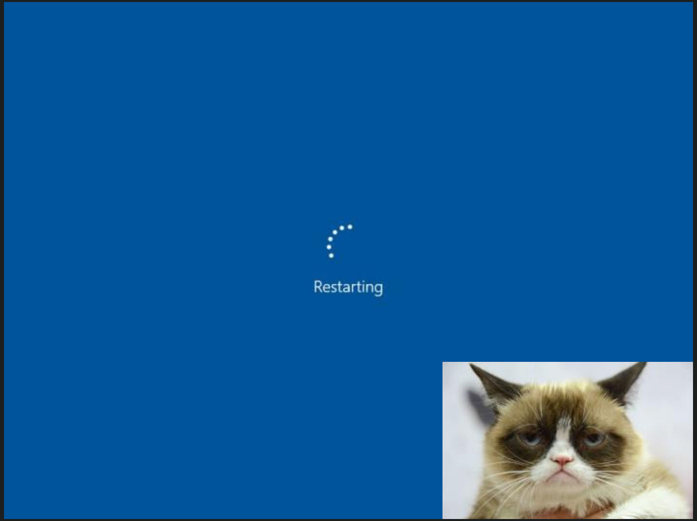
> What happens when a process opens a file without sharing access:


Notepad

The process cannot access the file because it is being used by another process.

Introduction

> and if another process really needs to access the file....



The Windows Restart Manager

The playing field

The Origin

> Introduced in Windows Vista in the “RstrtMgr.dll” library

> Goal: avoid/reduce OS reboots during updates

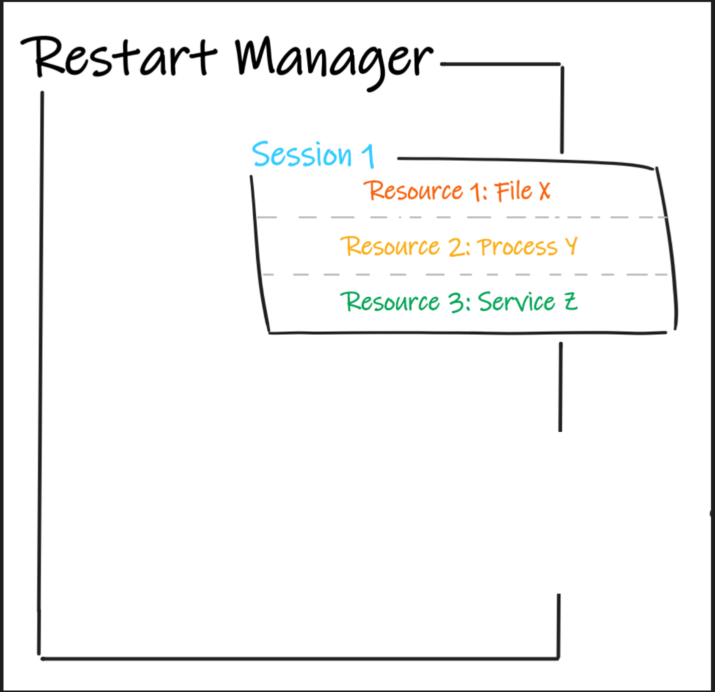
> Allow applications to check that resources they need aren’t locked by other processes and request the termination of the blocking process, if needed

Architecture

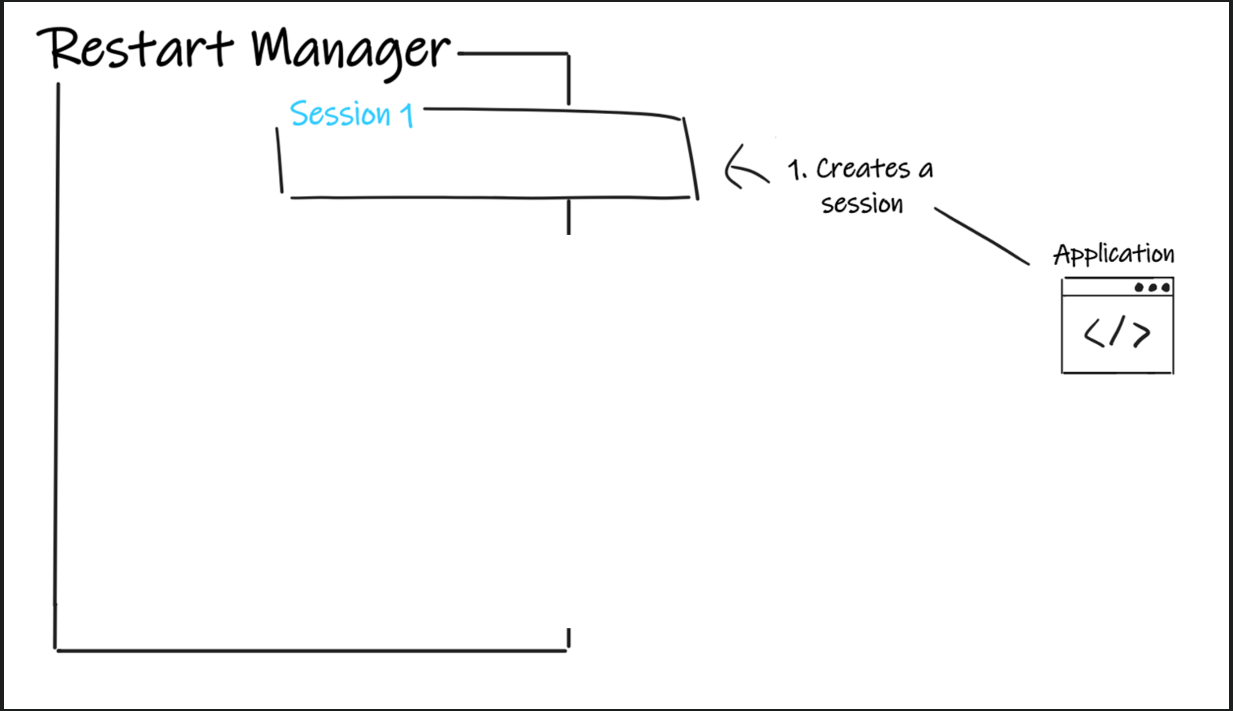
> Applications communicate with the Restart Manager through sessions

> Each session contains one or more resources, that can be:

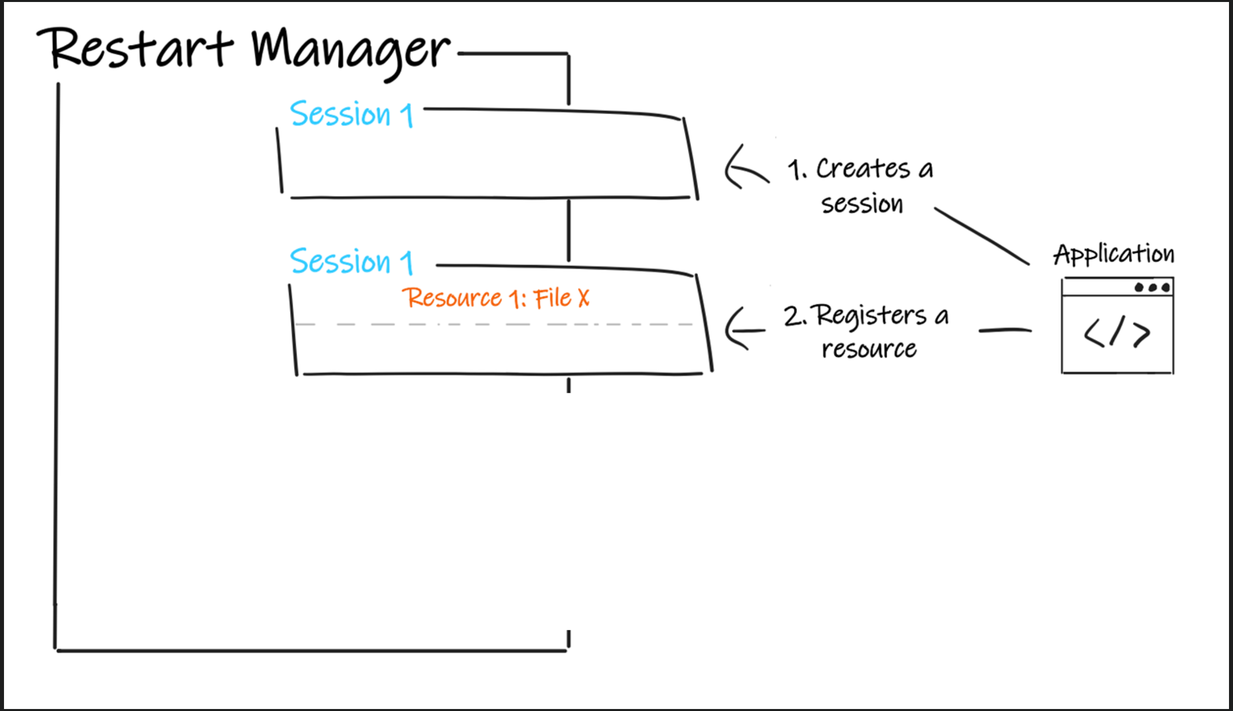
- Files
- Processes
- Services



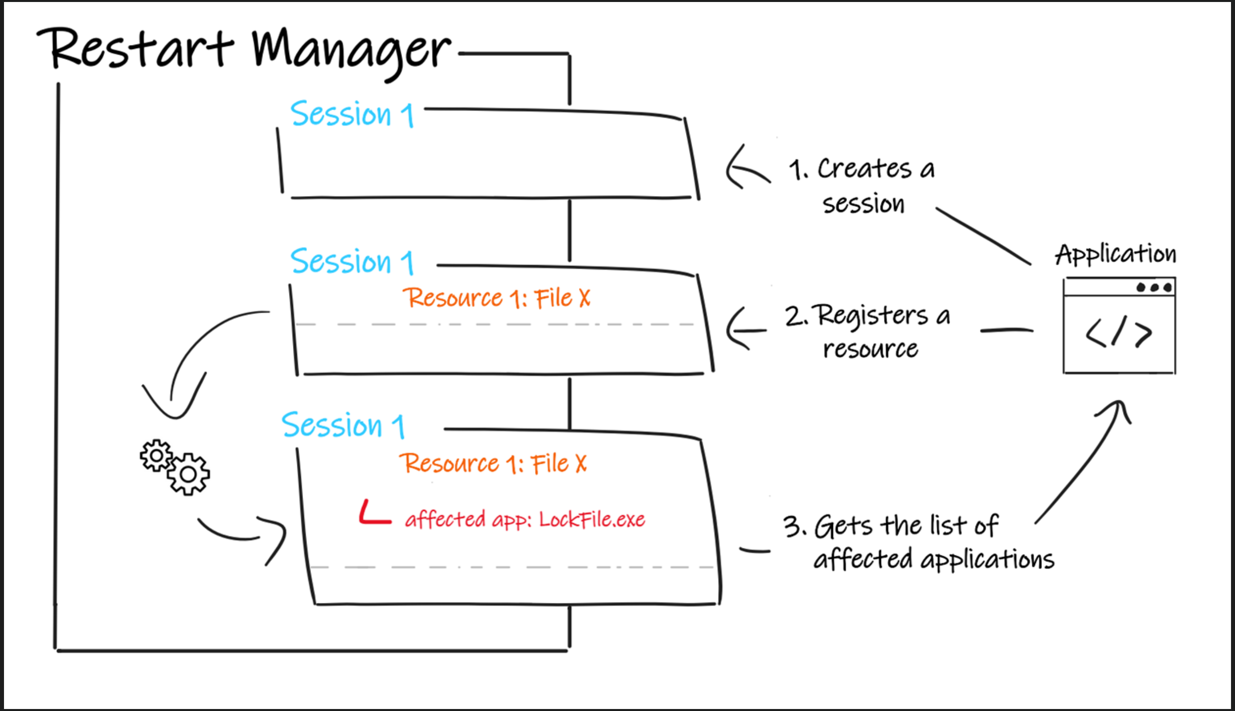
Global Operation



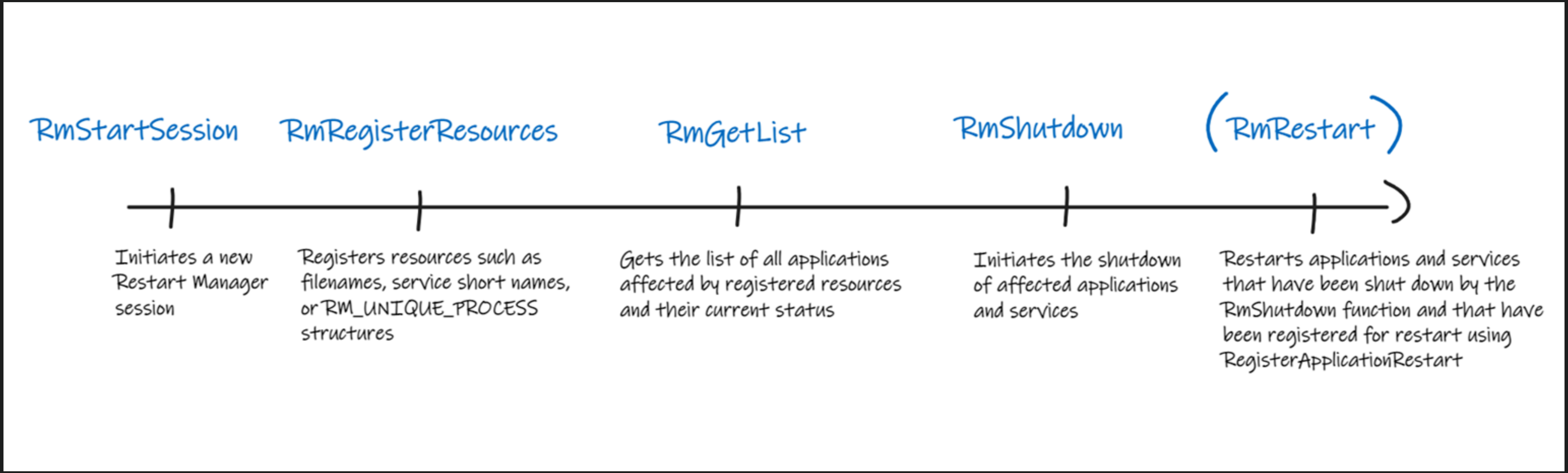
Global Operation



Global Operation



Exported Functions


RmStartSession
Initiates a new Restart Manager session
RmRegisterResources
Registers resources such as filenames, service short names, or RM_UNIQUE_PROCESS structures
RmGetList
Gets the list of all applications affected by registered resources and their current status
RmShutdown
Initiates the shutdown of affected applications and services
(RmRestart)
Restarts applications and services that have been shut down by the RmShutdown function and that have been registered for restart using RegisterApplicationRestart

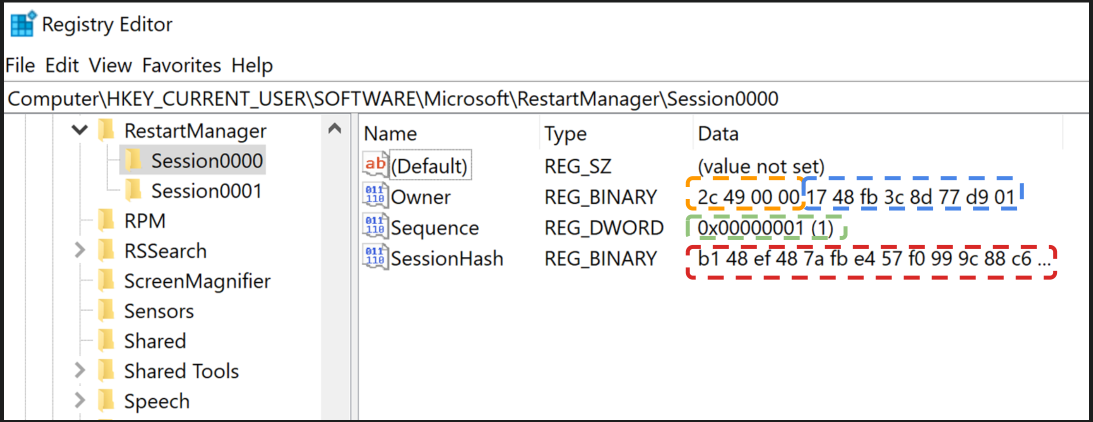
CROWDSTRIKE

RmStartSession()

> Assigns a session ID

> Creates the internal database of the session

> Initializes a registry hive for the session:



Registry Editor

File Edit View Favorites Help

Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\RestartManager\Session0000

```
RestartManager
    Session0000
    Session0001
RPM
RSSearch
ScreenMagnifier
Sensors
Shared
Shared Tools
Speech
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
Owner	REG_BINARY	2c 49 00 00 17 48 fb 3c 8d 77 d9 01
Sequence	REG_DWORD	0x00000001 (1)
SessionHash	REG_BINARY	b1 48 ef 48 7a fb e4 57 f0 99 9c 88 c6 ...

PID of the register's process

FILETIME of the register's process creation

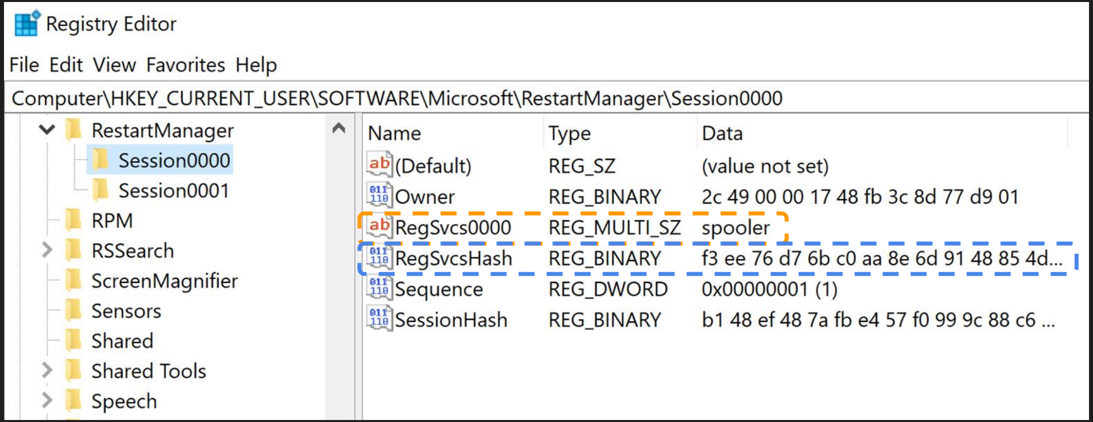
Current state of the session

Hash of the "Owner" + unique UUID used in the SessionKey

RmRegisterResources()

> Registers the resources in its internal database

> Updates the registry hive of the session:



Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\RestartManager\Session0000

Name	Type	Data
(Default)	REG_SZ	(value not set)
Owner	REG_BINARY	2c 49 00 00 17 48 fb 3c 8d 77 d9 01
RegSvcs0000	REG_MULTI_SZ	spooler
RegSvcsHash	REG_BINARY	f3 ee 76 d7 6b c0 aa 8e 6d 91 48 85 4d...
Sequence	REG_DWORD	0x00000001 (1)
SessionHash	REG_BINARY	b1 48 ef 48 7a fb e4 57 f0 99 9c 88 c6 ...

Registered Resource

Hash of the registered resource

RmGetList()

> Goal: find the affected applications for each different type of resource

> Relies on decorators: internal components designed to collect information

> 2 categories of information collected:

- System information
- Application information

RmGetList() - Decorators

System Information

> SysProcInfo

> SvcInfo

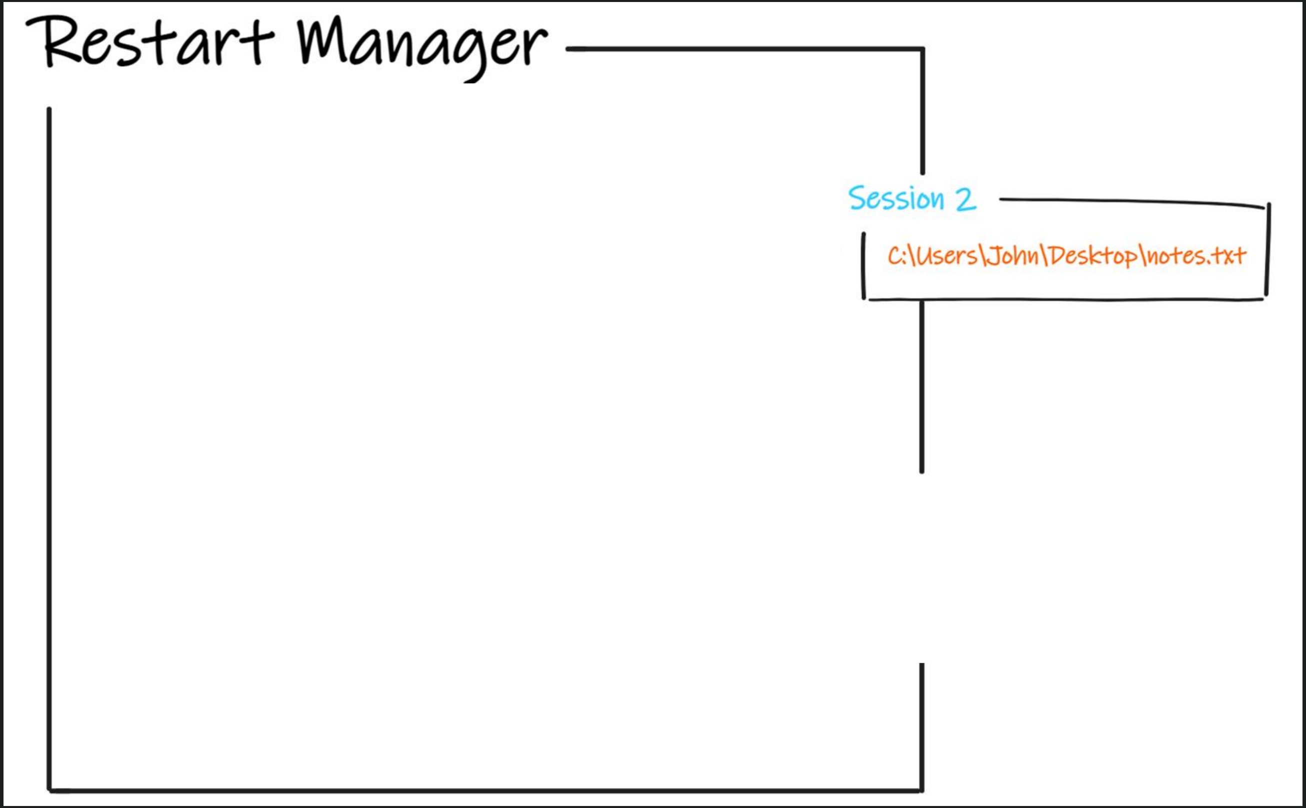
> WindowInfo

Application Information

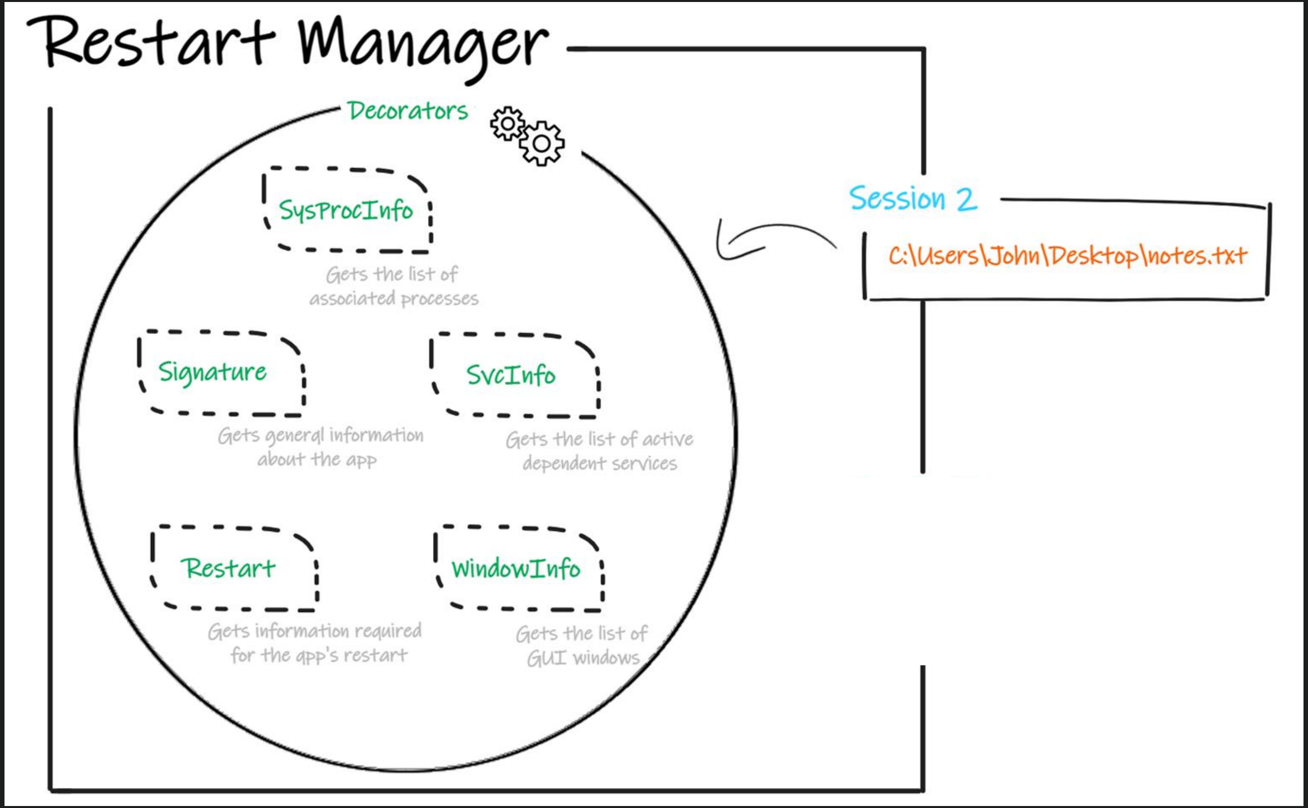
> Signature

> Restart

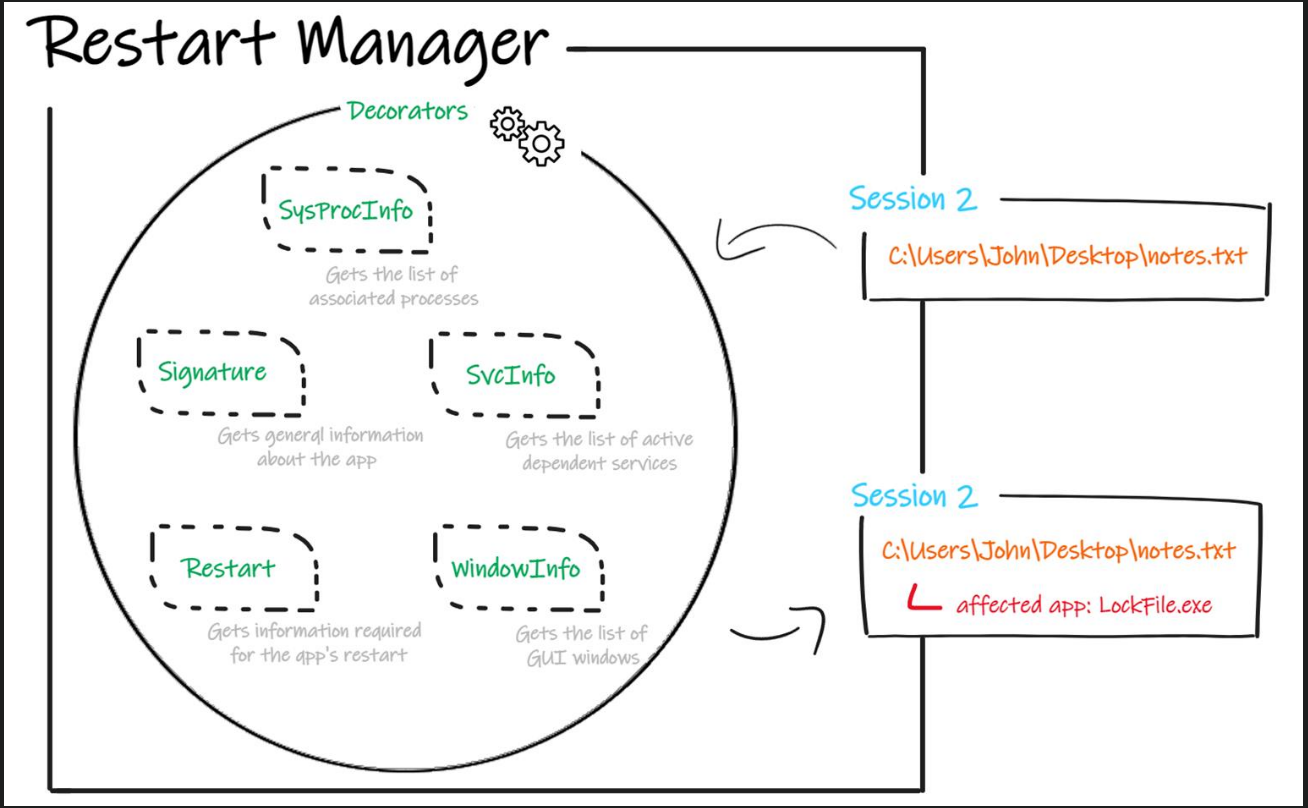
RmGetList() - Decorators



RmGetList() - Decorators

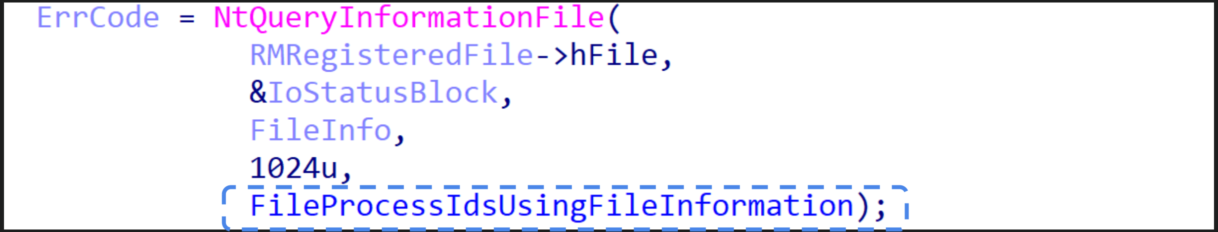

Restart Manager
Decorators
SysProcInfo
Gets the list of associated processes
Signature
Gets general information about the app
SvcInfo
Gets the list of active dependent services
Restart
Gets information required for the app's restart
WindowInfo
Gets the list of GUI windows
Session 2
C:\Users\John\Desktop\notes.txt

RmGetList() - Decorators


Restart Manager
Decorators
SysProcInfo
Gets the list of associated processes
Signature
Gets general information about the app
SvcInfo
Gets the list of active dependent services
Restart
Gets information required for the app's restart
WindowInfo
Gets the list of GUI windows
Session 2
C:\Users\John\Desktop\notes.txt
Session 2
C:\Users\John\Desktop\notes.txt
affected app: LockFile.exe

RmGetList() - For Files

> For registered files, retrieves the PIDs of processes using:



```
ErrCode = NtQueryInformationFile(
                RMRegisteredFile->hFile,
                &IoStatusBlock,
                FileInfo,
                1024u,
                FileProcessIdsUsingFileInformation);
```

RmGetList() - For Services

> For registered services:

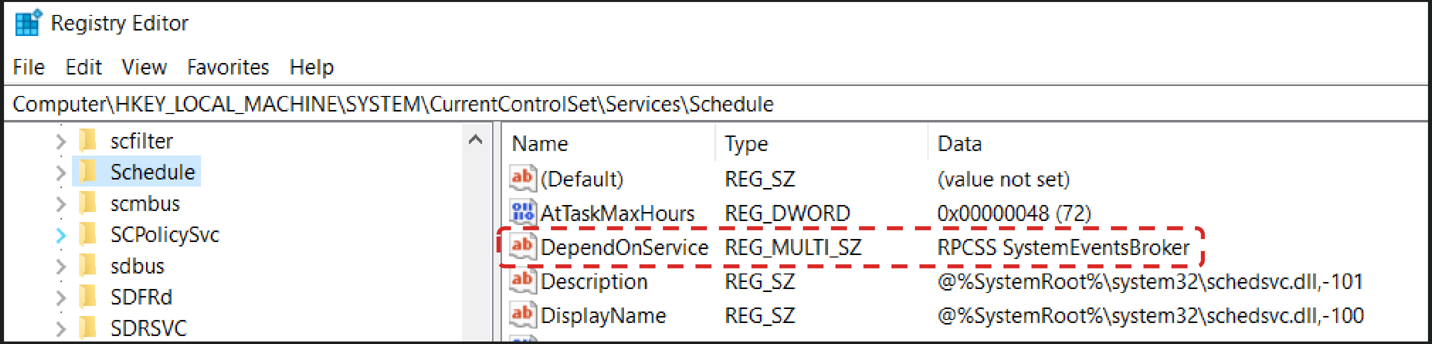
- Retrieves information about the service itself
- If the service is currently running, retrieves the list of active dependant services

RmGetList() - Dependent Services

> Services can depend on one or more other services

> The other service(s) must be running before the dependent service can run

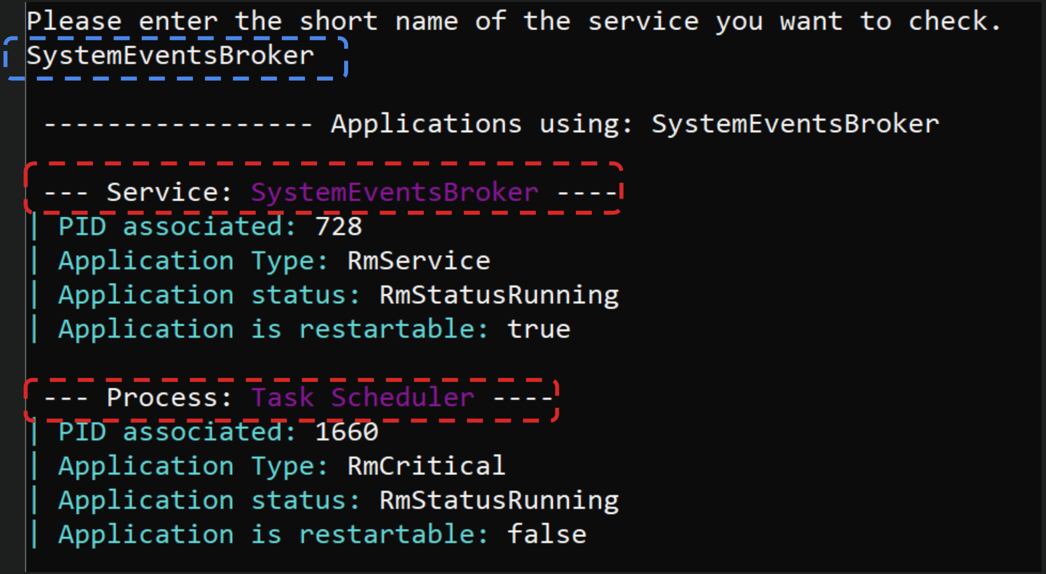
> Value defined in the registry hive of the service:



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Schedule

Name	Type	Data
(Default)	REG_SZ	(value not set)
AtTaskMaxHours	REG_DWORD	0x00000048 (72)
DependOnService	REG_MULTI_SZ	RPCSS SystemEventsBroker
Description	REG_SZ	@%SystemRoot%\system32\schedsvc.dll,-101
DisplayName	REG_SZ	@%SystemRoot%\system32\schedsvc.dll,-100

RmGetList() - For Services



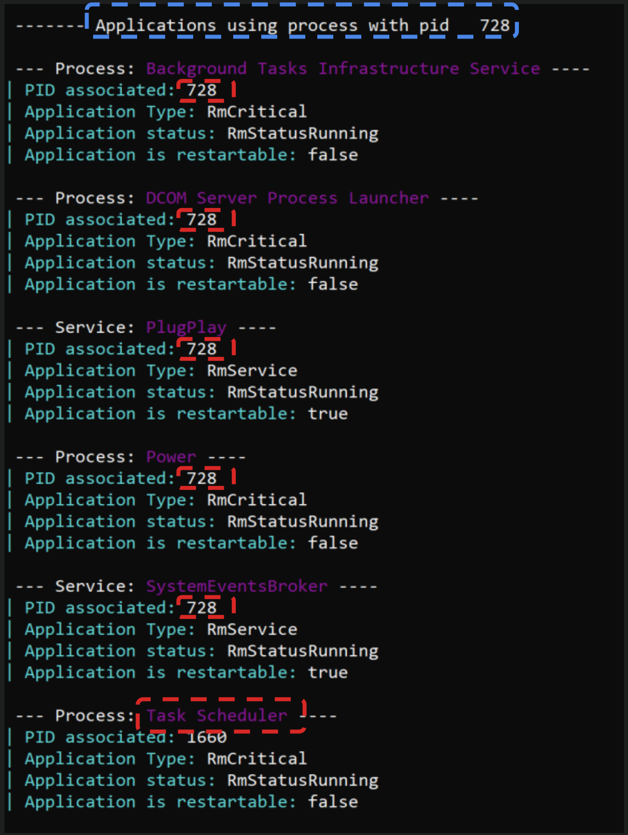
```
Please enter the short name of the service you want to check.
SystemEventsBroker

 ---------------- Applications using: SystemEventsBroker

 --- Service: SystemEventsBroker ----
|  PID associated: 728
|  Application Type: RmService
|  Application status: RmStatusRunning
|  Application is restartable: true

 --- Process: Task Scheduler ----
|  PID associated: 1660
|  Application Type: RmCritical
|  Application status: RmStatusRunning
|  Application is restartable: false
```

RmGetList() - For Processes

> For registered processes:

- Retrieves information about the process itself
- If associated with a service, retrieves the list of active dependant services

RmGetList() - For Processes

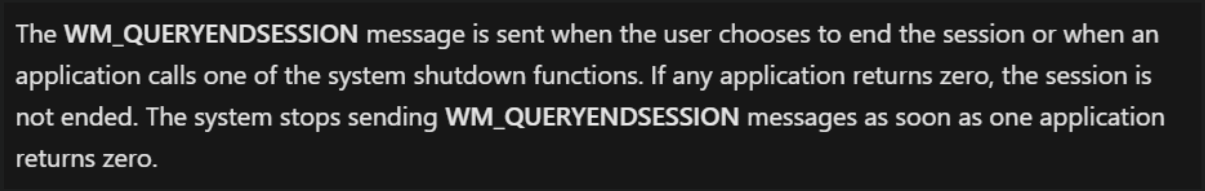


```
-------Applications using process with pid  728

--- Process: Background Tasks Infrastructure Service ----
| PID associated: 728
| Application Type: RmCritical
| Application status: RmStatusRunning
| Application is restartable: false

--- Process: DCOM Server Process Launcher ----
| PID associated: 728
| Application Type: RmCritical
| Application status: RmStatusRunning
| Application is restartable: false

--- Service: PlugPlay ----
| PID associated: 728
| Application Type: RmService
| Application status: RmStatusRunning
| Application is restartable: true

--- Process: Power ----
| PID associated: 728
| Application Type: RmCritical
| Application status: RmStatusRunning
| Application is restartable: false

--- Service: SystemEventsBroker ----
| PID associated: 728
| Application Type: RmService
| Application status: RmStatusRunning
| Application is restartable: true

--- Process: Task Scheduler ----
| PID associated: 1660
| Application Type: RmCritical
| Application status: RmStatusRunning
| Application is restartable: false
```

RmShutdown()

> Scenario 1: the affected application is a GUI application

- 1st call to SendMessageTimeoutW() with WM_QUERYENDSESSION



The **WM_QUERYENDSESSION** message is sent when the user chooses to end the session or when an application calls one of the system shutdown functions. If any application returns zero, the session is not ended. The system stops sending **WM_QUERYENDSESSION** messages as soon as one application returns zero.

- 2nd call to SendMessageTimeoutW() with WM_ENDSESSION

- If not, 3rd call to SendMessageTimeoutW() with WM_CLOSE

RmShutdown()

> Scenario 2: the affected application is a console application

- Sends a CTRL_C_EVENT notification

- By default, processed by the control handler that calls ExitProcess()

RmShutdown()

> Scenario 3: the affected application is associated with a service

- Stops the service using ControlService()

- Terminates the associated process using TerminateProcess()

RmShutdown()

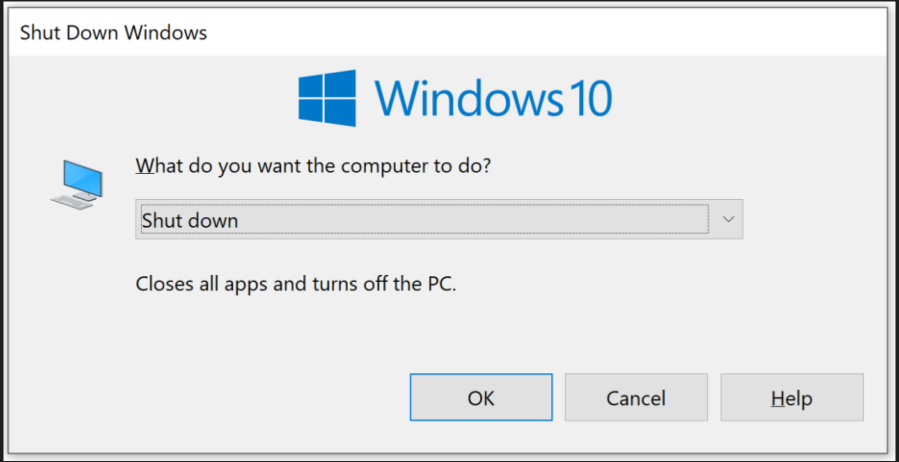
> Scenario 4: the affected application is explorer.exe

- *Same as for classic applications* - 1st call to SendMessageTimeoutW() with WM_QUERYENDSESSION
- *Same as for classic applications* - 2nd call to SendMessageTimeoutW() with WM_ENDSESSION

> The main difference with classic applications: no 3rd call to SendMessageTimeoutW() with WM_CLOSE

RmShutdown()

> What may happen when WM_CLOSE message is sent to explorer.exe:



Shut Down Windows

What do you want the computer to do?

Shut down

Closes all apps and turns off the PC.

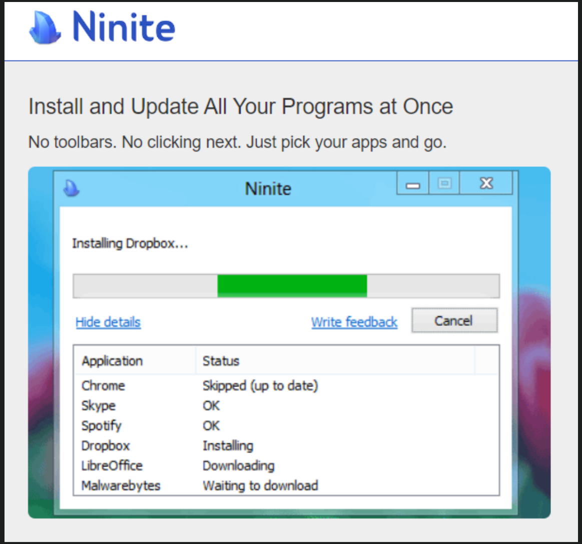
Legitimate Use Case Example

> Typical use case: installers & updaters

> Benefits:

- Ensures to be able to complete the update
- Avoids a reboot

> Let's find out who/how thanks to ninite :)



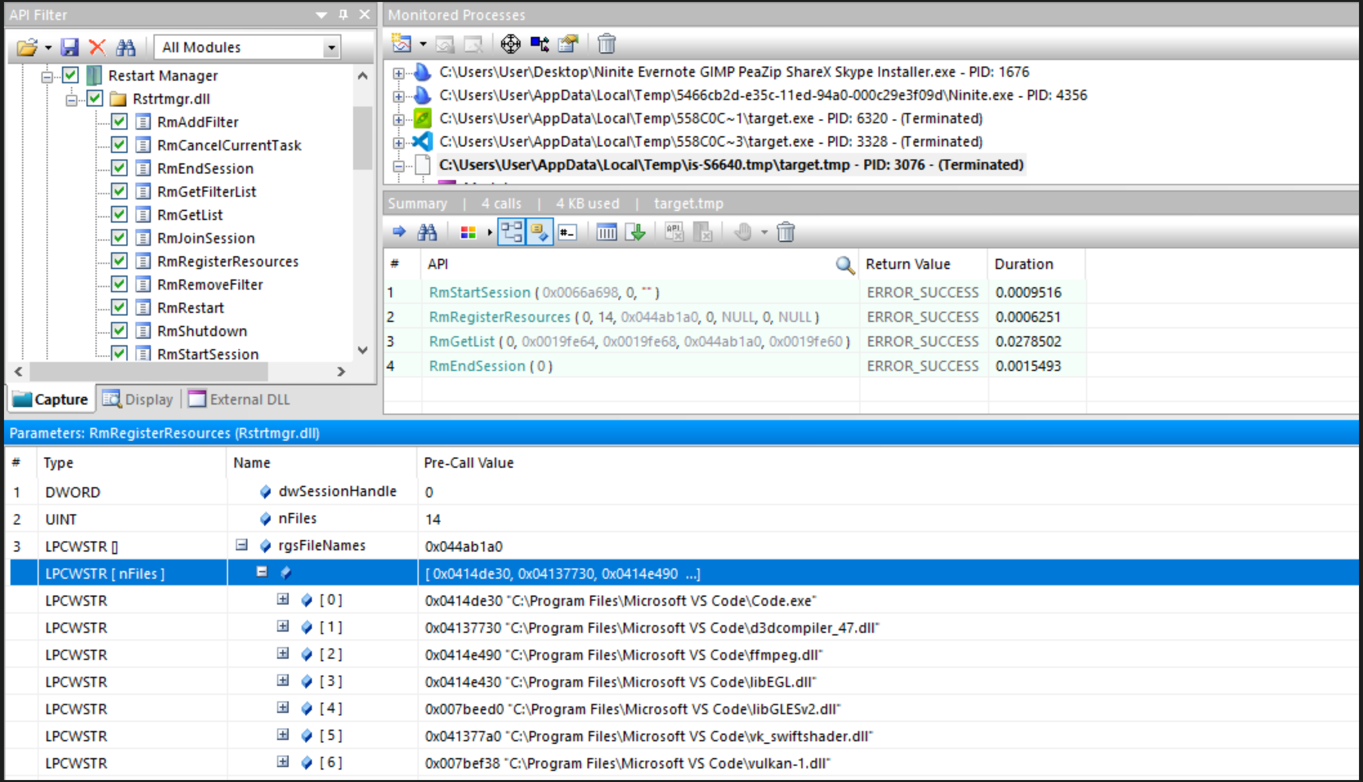
Install and Update All Your Programs at Once

No toolbars. No clicking next. Just pick your apps and go.

Installing Dropbox...

Application	Status
Chrome	Skipped (up to date)
Skype	OK
Spotify	OK
Dropbox	Installing
LibreOffice	Downloading
Malwarebytes	Waiting to download

Legitimate Use Case Example



API Filter

All Modules

- Restart Manager
  - Rstrtmgr.dll
    - RmAddFilter
    - RmCancelCurrentTask
    - RmEndSession
    - RmGetFilterList
    - RmGetList
    - RmJoinSession
    - RmRegisterResources
    - RmRemoveFilter
    - RmRestart
    - RmShutdown
    - RmStartSession

Capture | Display | External DLL

Monitored Processes

```
C:\Users\User\Desktop\Ninite Evernote GIMP PeaZip ShareX Skype Installer.exe - PID: 1676
C:\Users\User\AppData\Local\Temp\5466cb2d-e35c-11ed-94a0-000c29e3f09d\Ninite.exe - PID: 4356
C:\Users\User\AppData\Local\Temp\558C0C~1\target.exe - PID: 6320 - (Terminated)
C:\Users\User\AppData\Local\Temp\558C0C~3\target.exe - PID: 3328 - (Terminated)
C:\Users\User\AppData\Local\Temp\is-S6640.tmp\target.tmp - PID: 3076 - (Terminated)
```

Summary | 4 calls | 4 KB used | target.tmp

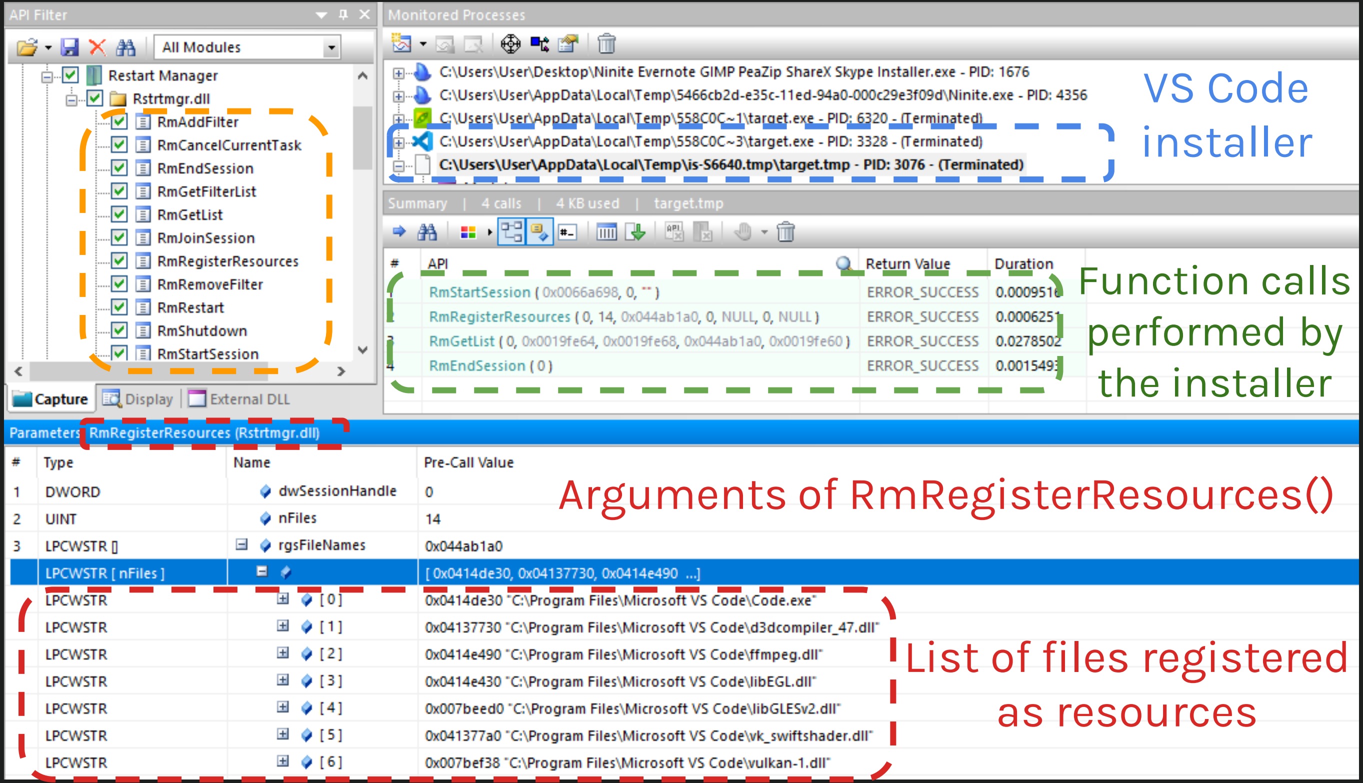
#	API	Return Value	Duration
1	RmStartSession (0x0066a698, 0, "")	ERROR_SUCCESS	0.0009516
2	RmRegisterResources (0, 14, 0x044ab1a0, 0, NULL, 0, NULL)	ERROR_SUCCESS	0.0006251
3	RmGetList (0, 0x0019fe64, 0x0019fe68, 0x044ab1a0, 0x0019fe60)	ERROR_SUCCESS	0.0278502
4	RmEndSession (0)	ERROR_SUCCESS	0.0015493

Parameters: RmRegisterResources (Rstrtmgr.dll)

#	Type	Name	Pre-Call Value
1	DWORD	dwSessionHandle	0
2	UINT	nFiles	14
3	LPCWSTR []	rgsFileNames	0x044ab1a0
	LPCWSTR [nFiles]		[0x0414de30, 0x04137730, 0x0414e490 ...]
	LPCWSTR	[0]	0x0414de30 "C:\Program Files\Microsoft VS Code\Code.exe"
	LPCWSTR	[1]	0x04137730 "C:\Program Files\Microsoft VS Code\d3dcompiler_47.dll"
	LPCWSTR	[2]	0x0414e490 "C:\Program Files\Microsoft VS Code\ffmpeg.dll"
	LPCWSTR	[3]	0x0414e430 "C:\Program Files\Microsoft VS Code\libEGL.dll"
	LPCWSTR	[4]	0x007beed0 "C:\Program Files\Microsoft VS Code\libGLESv2.dll"
	LPCWSTR	[5]	0x041377a0 "C:\Program Files\Microsoft VS Code\vk_swiftshader.dll"
	LPCWSTR	[6]	0x007bef38 "C:\Program Files\Microsoft VS Code\vulkan-1.dll"

Legitimate Use Case Example

Hooked functions



API Filter

All Modules

Restart Manager
- Rstrtmgr.dll
  - RmAddFilter
  - RmCancelCurrentTask
  - RmEndSession
  - RmGetFilterList
  - RmGetList
  - RmJoinSession
  - RmRegisterResources
  - RmRemoveFilter
  - RmRestart
  - RmShutdown
  - RmStartSession

Capture | Display | External DLL

Monitored Processes

```
C:\Users\User\Desktop\Ninite Evernote GIMP PeaZip ShareX Skype Installer.exe - PID: 1676
C:\Users\User\AppData\Local\Temp\5466cb2d-e35c-11ed-94a0-000c29e3f09d\Ninite.exe - PID: 4356
C:\Users\User\AppData\Local\Temp\558C0C~1\target.exe - PID: 6320 - (Terminated)
C:\Users\User\AppData\Local\Temp\558C0C~3\target.exe - PID: 3328 - (Terminated)
C:\Users\User\AppData\Local\Temp\is-S6640.tmp\target.tmp - PID: 3076 - (Terminated)
```

VS Code installer

Summary | 4 calls | 4 KB used | target.tmp

#	API	Return Value	Duration
1	RmStartSession (0x0066a698, 0, "")	ERROR_SUCCESS	0.0009516
2	RmRegisterResources (0, 14, 0x044ab1a0, 0, NULL, 0, NULL)	ERROR_SUCCESS	0.0006251
3	RmGetList (0, 0x0019fe64, 0x0019fe68, 0x044ab1a0, 0x0019fe60)	ERROR_SUCCESS	0.0278502
4	RmEndSession (0)	ERROR_SUCCESS	0.001549

Function calls performed by the installer

Parameters: RmRegisterResources (Rstrtmgr.dll)

#	Type	Name	Pre-Call Value
1	DWORD	dwSessionHandle	0
2	UINT	nFiles	14
3	LPCWSTR []	rgsFileNames	0x044ab1a0
	LPCWSTR [nFiles]		[0x0414de30, 0x04137730, 0x0414e490 ...]
	LPCWSTR	[0]	0x0414de30 "C:\Program Files\Microsoft VS Code\Code.exe"
	LPCWSTR	[1]	0x04137730 "C:\Program Files\Microsoft VS Code\d3dcompiler_47.dll"
	LPCWSTR	[2]	0x0414e490 "C:\Program Files\Microsoft VS Code\ffmpeg.dll"
	LPCWSTR	[3]	0x0414e430 "C:\Program Files\Microsoft VS Code\libEGL.dll"
	LPCWSTR	[4]	0x007beed0 "C:\Program Files\Microsoft VS Code\libGLESv2.dll"
	LPCWSTR	[5]	0x041377a0 "C:\Program Files\Microsoft VS Code\vk_swiftshader.dll"
	LPCWSTR	[6]	0x007bef38 "C:\Program Files\Microsoft VS Code\vulkan-1.dll"

Arguments of RmRegisterResources()

List of files registered as resources

Malicious Use Cases

How can the Restart Manager be hijacked?

Supporting Ransomware Encryption

Cheat code Δ✚XO

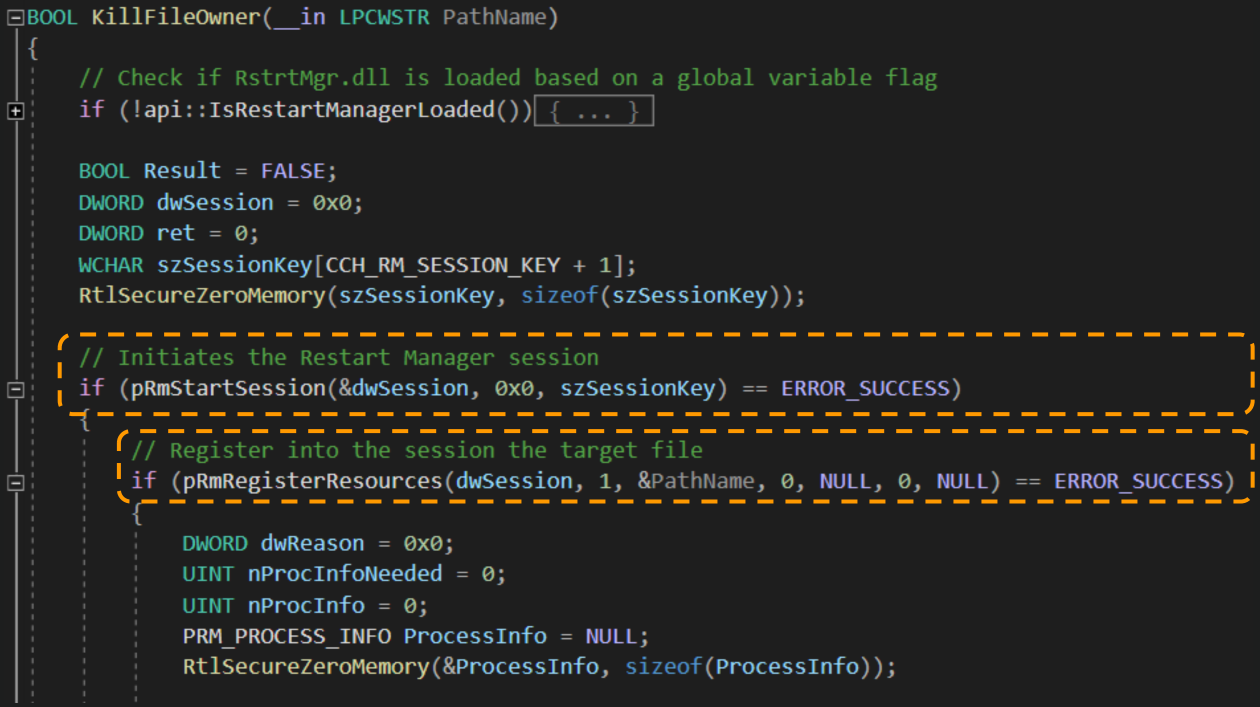
A real world example: Conti Ransomware

> Source code leaked in March 2022

> Goal: check if a potential file to encrypt is blocked by another process and attempts to terminate it if necessary

> Method: iterating over files to register each potential target as a resource in a Restart Manager session

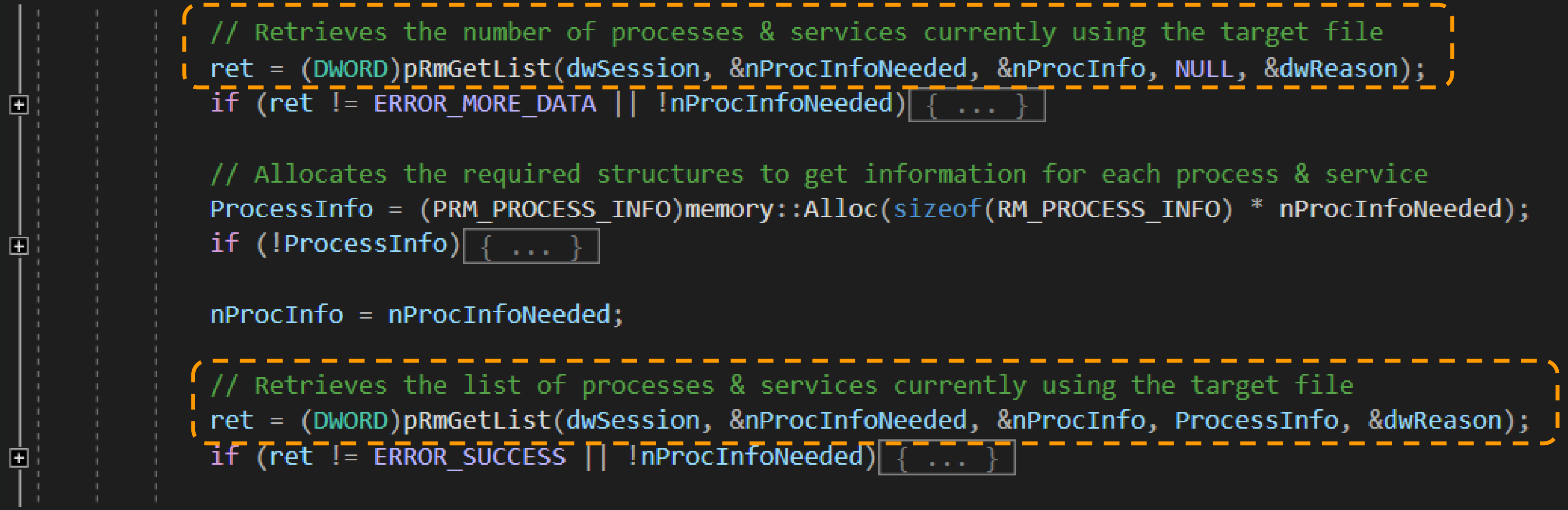
Step 1: Register the target file



```
BOOL KillFileOwner(__in LPCWSTR PathName)
{
    // Check if RstrtMgr.dll is loaded based on a global variable flag
    if (!api::IsRestartManagerLoaded()) { ... }

    BOOL Result = FALSE;
    DWORD dwSession = 0x0;
    DWORD ret = 0;
    WCHAR szSessionKey[CCH_RM_SESSION_KEY + 1];
    RtlSecureZeroMemory(szSessionKey, sizeof(szSessionKey));

    // Initiates the Restart Manager session
    if (pRmStartSession(&dwSession, 0x0, szSessionKey) == ERROR_SUCCESS)
    {
        // Register into the session the target file
        if (pRmRegisterResources(dwSession, 1, &PathName, 0, NULL, 0, NULL) == ERROR_SUCCESS)
        {
            DWORD dwReason = 0x0;
            UINT nProcInfoNeeded = 0;
            UINT nProcInfo = 0;
            PRM_PROCESS_INFO ProcessInfo = NULL;
            RtlSecureZeroMemory(&ProcessInfo, sizeof(ProcessInfo));
```

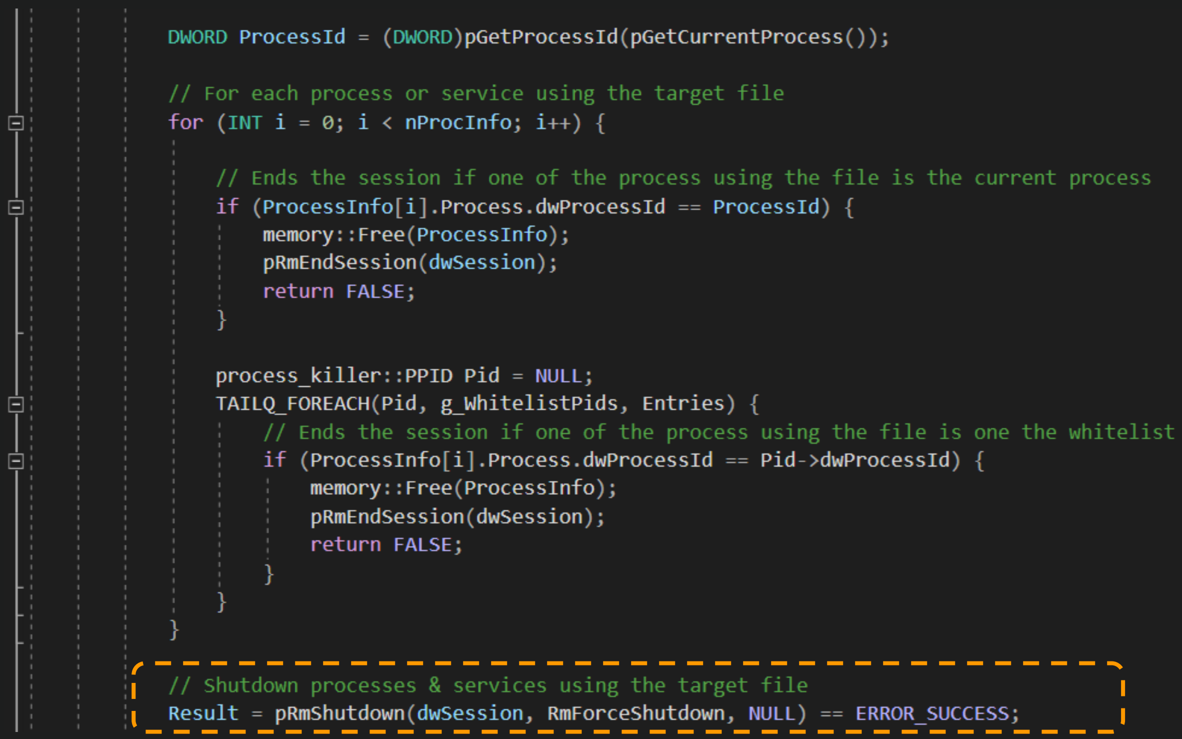
CROWDSTRIKE

Step 2: Retrieve the list of affected apps



```
// Retrieves the number of processes & services currently using the target file
ret = (DWORD)pRmGetList(dwSession, &nProcInfoNeeded, &nProcInfo, NULL, &dwReason);
if (ret != ERROR_MORE_DATA || !nProcInfoNeeded) { ... }

// Allocates the required structures to get information for each process & service
ProcessInfo = (PRM_PROCESS_INFO)memory::Alloc(sizeof(RM_PROCESS_INFO) * nProcInfoNeeded);
if (!ProcessInfo) { ... }

nProcInfo = nProcInfoNeeded;

// Retrieves the list of processes & services currently using the target file
ret = (DWORD)pRmGetList(dwSession, &nProcInfoNeeded, &nProcInfo, ProcessInfo, &dwReason);
if (ret != ERROR_SUCCESS || !nProcInfoNeeded) { ... }
```

Step 3: Terminating the affected apps

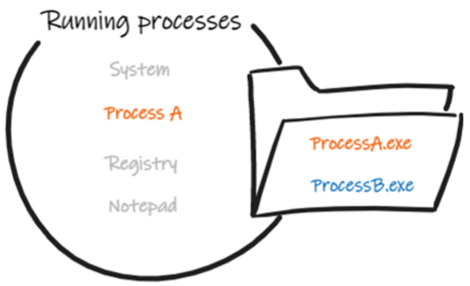


```
DWORD ProcessId = (DWORD)pGetProcessId(pGetCurrentProcess());

// For each process or service using the target file
for (INT i = 0; i < nProcInfo; i++) {

    // Ends the session if one of the process using the file is the current process
    if (ProcessInfo[i].Process.dwProcessId == ProcessId) {
        memory::Free(ProcessInfo);
        pRmEndSession(dwSession);
        return FALSE;
    }

    process_killer::PPID Pid = NULL;
    TAILQ_FOREACH(Pid, g_WhitelistPids, Entries) {
        // Ends the session if one of the process using the file is one the whitelist
        if (ProcessInfo[i].Process.dwProcessId == Pid->dwProcessId) {
            memory::Free(ProcessInfo);
            pRmEndSession(dwSession);
            return FALSE;
        }
    }
}

// Shutdown processes & services using the target file
Result = pRmShutdown(dwSession, RmForceShutdown, NULL) == ERROR_SUCCESS;
```

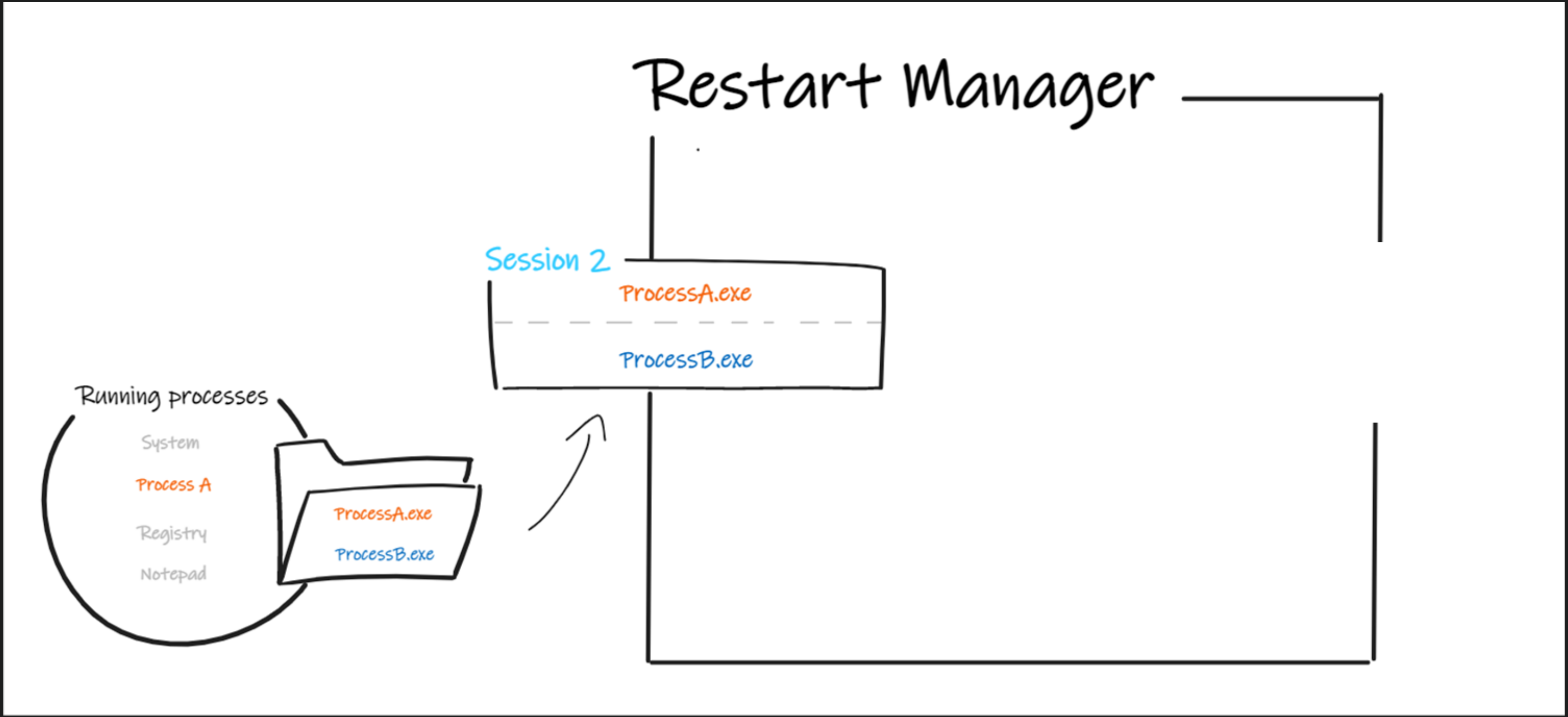
Anti-analysis and Evasion Purposes

Cheat code ΟΔΧΔ

Identifying Running Processes

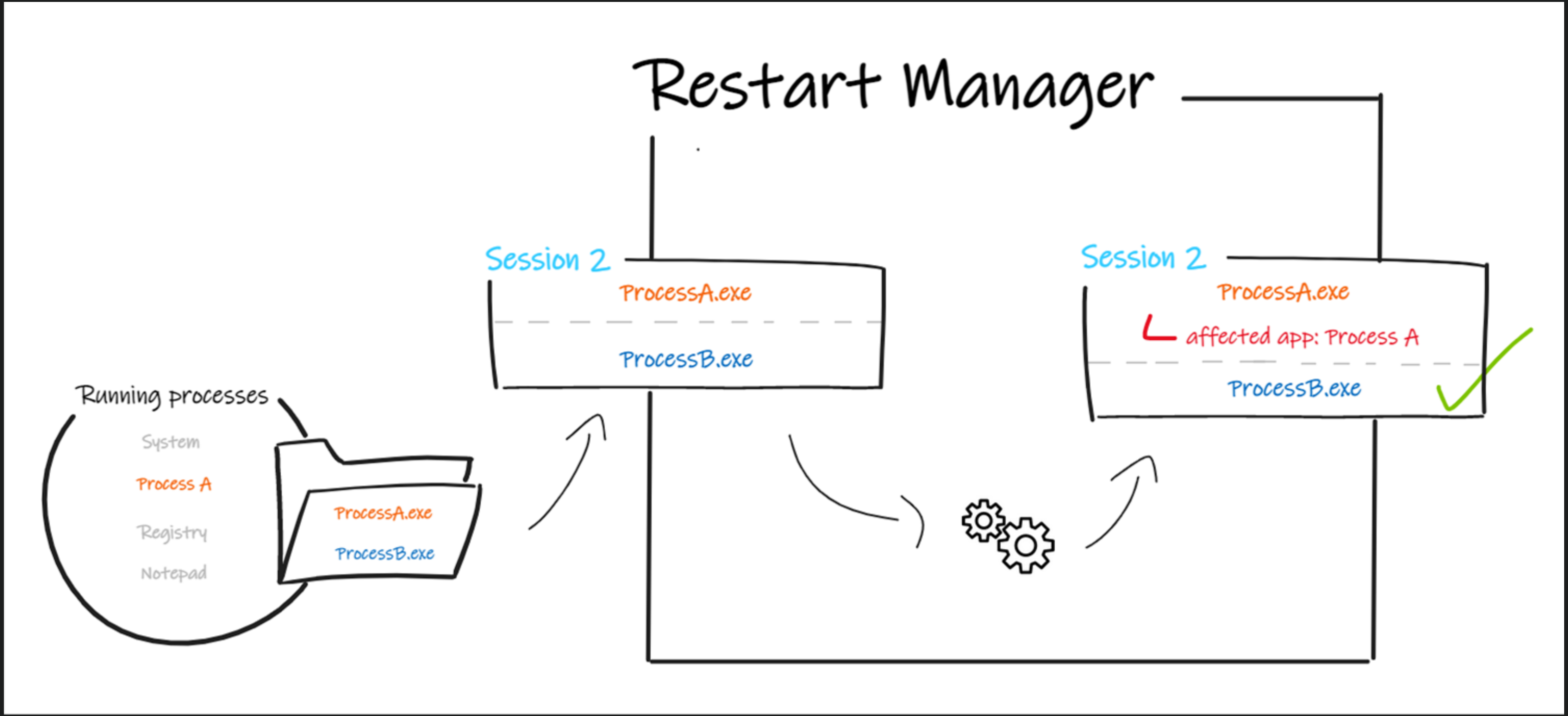


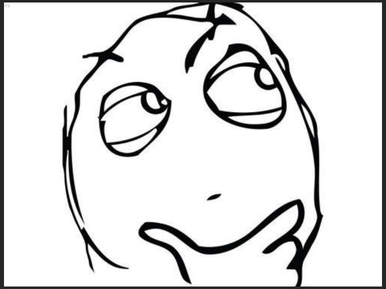
Identifying Running Processes



CROWDSTRIKE

Identifying Running Processes



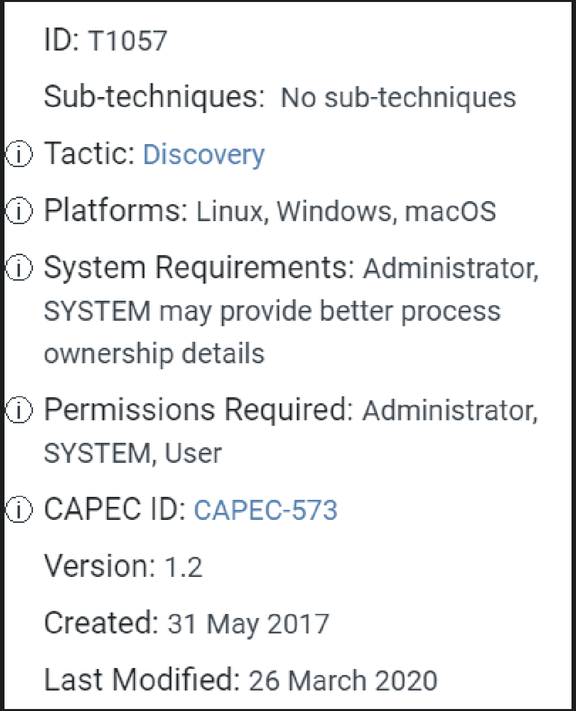


What can be done with this?

Process Discovery

> MITRE: *Process Discovery*

> Gather information about running processes and services (reconnaissance purposes)



ID: T1057

Sub-techniques: No sub-techniques

Tactic: Discovery

Platforms: Linux, Windows, macOS

System Requirements: Administrator, SYSTEM may provide better process ownership details

Permissions Required: Administrator, SYSTEM, User

CAPEC ID: CAPEC-573

Version: 1.2

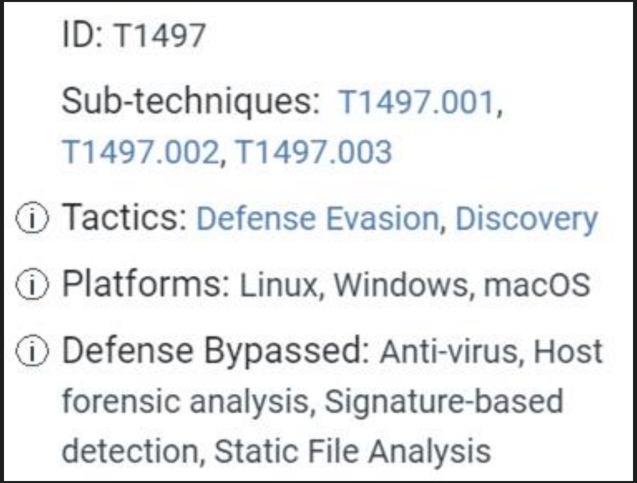
Created: 31 May 2017

Last Modified: 26 March 2020

Sandbox/Debugger Evasion

> MITRE: *Debugger Evasion, Virtualization/Sandbox Evasion*

> Detect and avoid debuggers, sandboxes and virtualized environments



ID: T1497

Sub-techniques: T1497.001, T1497.002, T1497.003

ⓘ Tactics: Defense Evasion, Discovery

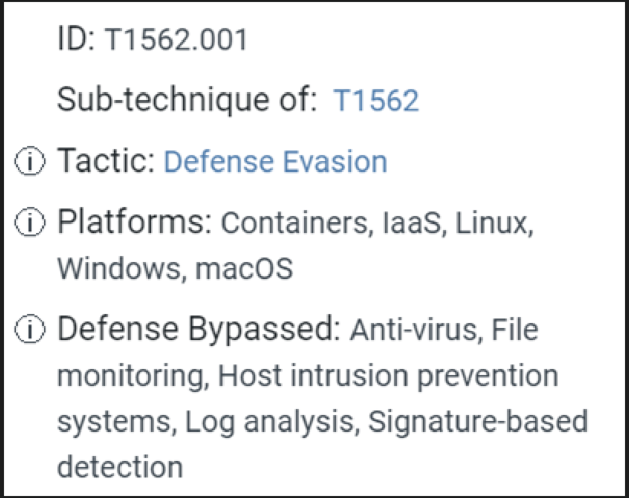
ⓘ Platforms: Linux, Windows, macOS

ⓘ Defense Bypassed: Anti-virus, Host forensic analysis, Signature-based detection, Static File Analysis

Anti-analysis

> MITRE: *Impair Defenses: Disable or Modify Tools*

> Detect & disable monitoring tools



ID: T1562.001

Sub-technique of: T1562

ⓘ Tactic: Defense Evasion

ⓘ Platforms: Containers, IaaS, Linux, Windows, macOS

ⓘ Defense Bypassed: Anti-virus, File monitoring, Host intrusion prevention systems, Log analysis, Signature-based detection

Time for a demo!

Protect Processes

Game over?

What Makes Applications Immune

> For applications associated with service: Protected Process & Protected Process Light

- Associated with binaries complying with specific signature requirements
- Defined by an attribute in the EPROCESS structure
- Limits accesses granted to protected processes

What Makes Applications Immune

> For other applications: User Interface Privilege Isolation (UIPI)

- Boundary between applications based on their integrity level
- Prevents apps with a lower privilege from sending messages to more privileged apps

Conclusion

Conclusion

> More information on the internals of this little known component of Windows

> New techniques for performing process discovery, evasion and impair defense

> Release of the tool on GitHub: https://github.com/MathildeVenault

Thank you for your attention!

Any questions?