FalconForce

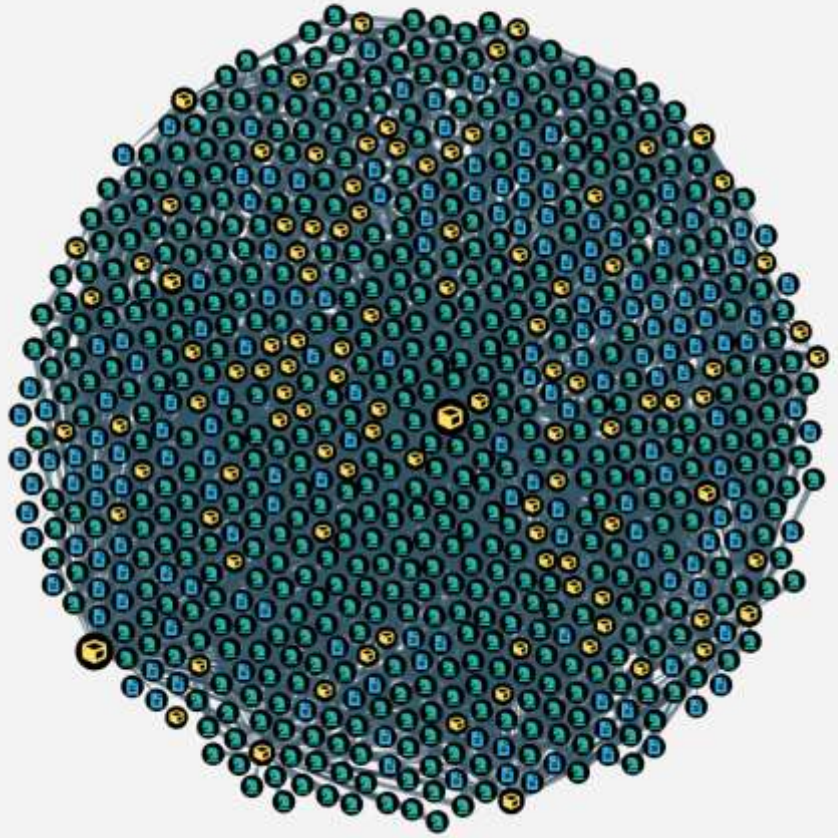
red canary

Would you rather have telemetry into 2 attacks or 20 ?

MITRE ATT&CKCON 3.0

Agenda

- Why do we care about data sources
- How do we determine what to utilize
- Conclusion



Oh, and if this does not excite you, please enjoy yourself with this mind soothing video

Jonny Johnson

Sr Threat Researcher @ Red Canary

Defensive Security Researcher

- Detection, Threat Hunting, Compromise Assessments
- Windows Internals, All Things Data, Reverse Engineering

Formerly Detection Engineer @SpecterOps
Host of the Detection: Challenging Paradigms Podcast
COD 1v1 Knife Pro (self-proclaimed)

@jsecurity101
github.com/jsecurity101
jonny.johnson@redcanary.com
medium.com/@jsecurity101

Olaf Hartong

Defensive Specialist @ FalconForce

Detection Engineer and Security Researcher

- Built and/or led Security Operations Centers
- Threat hunting, IR and Compromise assessments

Former documentary photographer
Father of 2 boys
“I like warm hugs”

@olafhartong
github.com/olafhartong
olaf@falconforce.nl
olafhartong.nl / falconforce.nl

Why do we care about data sources?

Value of understanding data sources

A good understanding of the (available) data sources allows us to understand:

- Where to start looking
- Which fields are of value for a good detection
- What is required to focus on behavioral detections
- What events have a good volume versus value balance
- How to ingest/enable them to broaden visibility

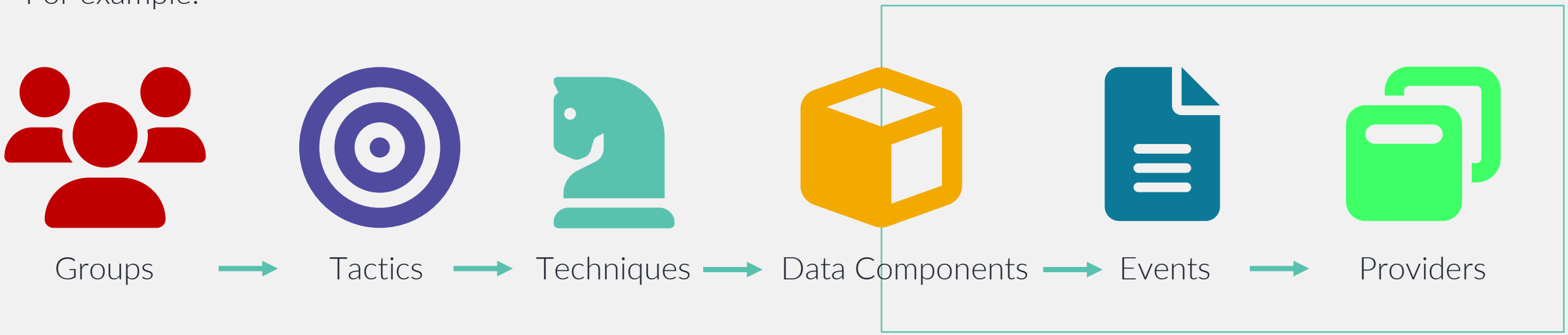
“Defenders think in lists.
Attackers think in graphs.
As long as this is true, attackers win.”

John Lambert - 2015

Linking data sources > data components > events

Since ATT&CK contains all kinds relations we can start combining sets of relationships with other sets.

For example:



The same can be done for;
tools, detection rules, attack/validation scripts, event fields and much, much more!

https://github.com/OTRF/OSSEM-DM

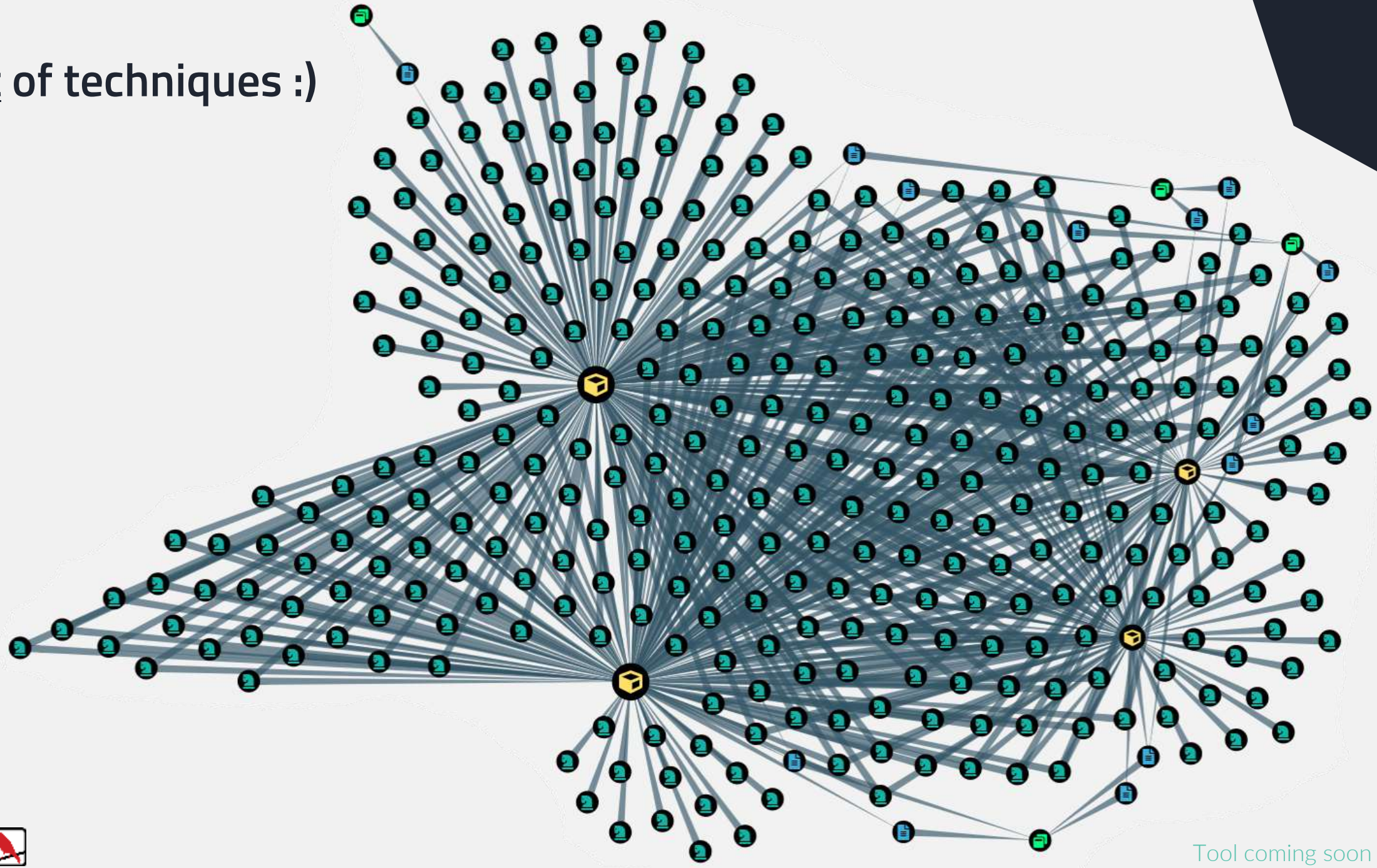
The top 5 data sources in ATT&CK

The data sources with the most technique references in ATT&CK are:

- Command: Command Execution (253x)
- Process: Process Creation (204x)
- File: File Modification (96x)
- Network Traffic: Network Traffic Content (96x)
- File: File Creation (87x)

Remember, dominance does not necessarily mean importance.

A lot of techniques :)



Tool coming soon

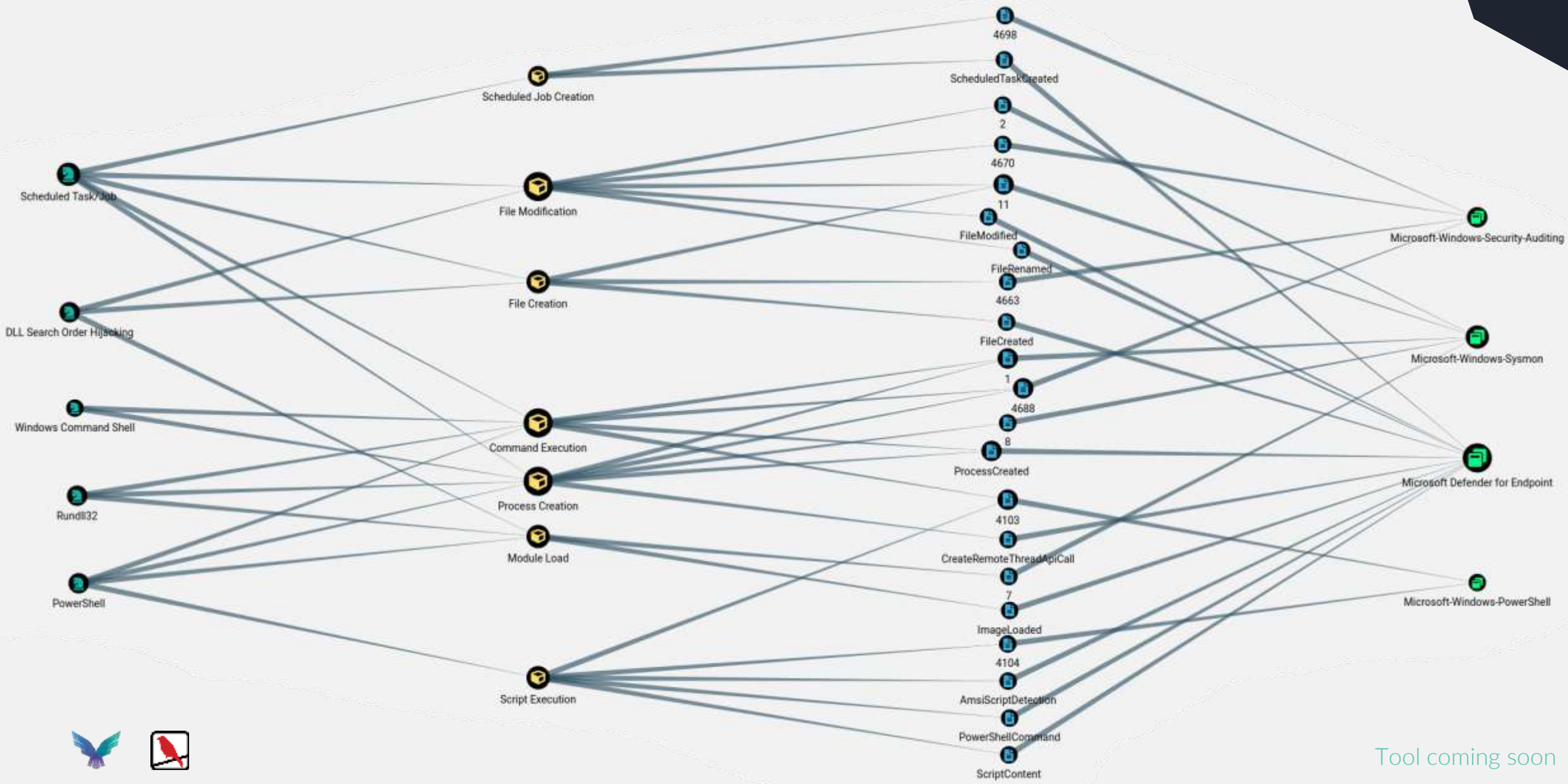
Why is this relevant ?

Top 5 most observed techniques by Red Canary:

- T1059.001 - Command and Scripting Interpreter: PowerShell
- T1059.003 - Command and Scripting Interpreter: Windows Command Shell
- T1218.011 - Signed Binary Proxy Execution: Rundll32
- T1053.005 - Scheduled Task/Job: Scheduled Task
- T1574.001 - Hijack Execution Flow: DLL Search Order Hijacking


http://redcanary.com/threat-detection-report

Top 5 most observed techniques by Red Canary



Tool coming soon

Data components for those techniques

- Scheduled Job Creation
- File Modification
- File Creation
- Command Execution
- Process Creation
- Module Load
- Script Execution

Top 5 data sources in ATT&CK

Command: Command Execution

Process: Process Creation

File: File Modification

Network Traffic: Network Traffic Content

File: File Creation

Windows Registry

Component numbers 8, 19, 32 and 51
in ATT&CK in terms of references

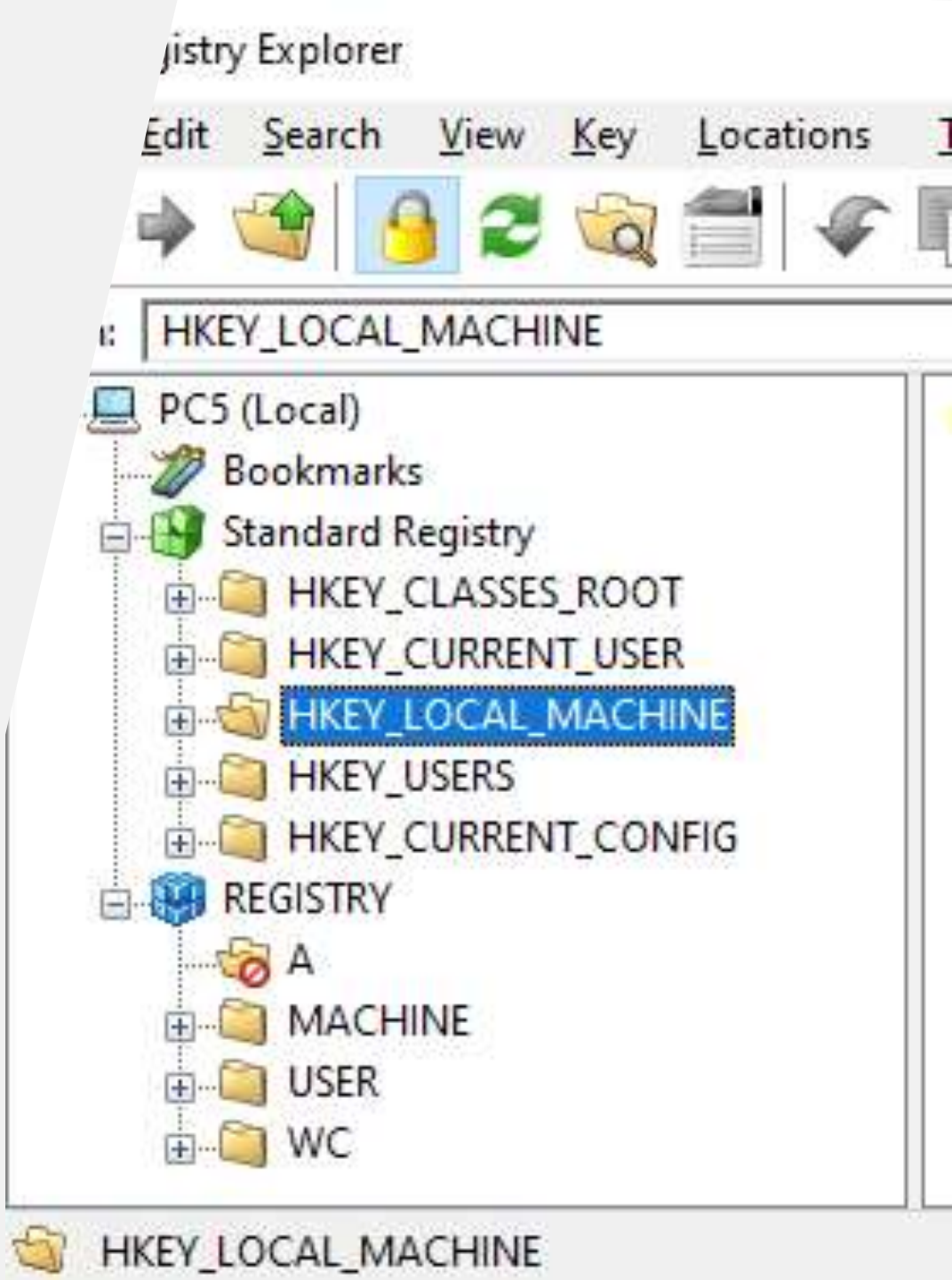
Why the registry?

It contains a ton of useful artefacts.

Forensic investigators love this data source.

Defenders utilize this source too little, while this is an amazingly rich source of valuable data.

Very strong source for behavioral detections, hard to avoid.



ATT&CK techniques referenced

There are 86 techniques that reference the Registry data source.

This data source has 4 components, which are referenced as follows:

- Windows Registry Key Modification (58x)
- Windows Registry Key Creation (17x)
- Windows Registry Key Access (7x)
- Windows Registry Key Deletion (4x)

It is very likely you'll be able to cover way more techniques than the ones currently mapped with this data source.


Registry Technique examples

- COM Hijacking (T1546.015)
- Service Creation (T1543.003, T1569.002, T1574.010, T1574.011)
- EDR Tampering (T1562.001)
- All kinds of discovery (T1007, T1016,T1033, T1082, T1201,T1518 and more)
- Persistence via custom keyboard layouts (T1547.?)
- Adding a local user via the registry (T1136.001)


Registry Enumeration examples

Applocker Policy Enumeration

HKLM\SOFTWARE\Policies\Microsoft\Wind

Audit, WEF and Sysmon Settings

HKEY_LOCAL_MACHINE\Software\Micros
HKEY_LOCAL_MACHINE\Software\Policies
HKLM\SYSTEM\CurrentControlSet\Services

RDP Cached Connections and settings

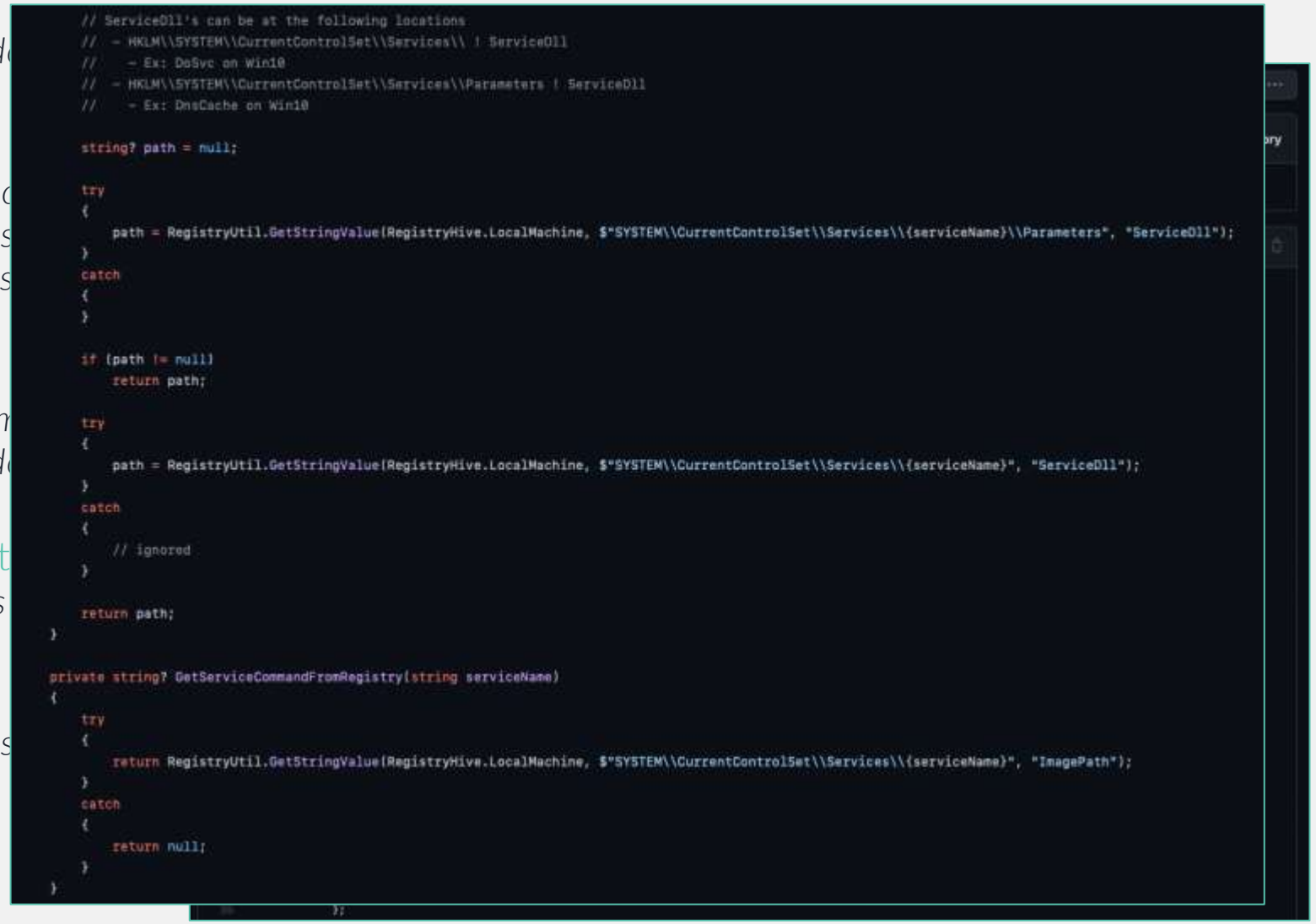
[HKC|HKCU] AND \Software\Microsoft\Term
HKLM\SOFTWARE\Policies\Microsoft\Wind

Defender Attack Surface Reduction sett

HKLM\Software\Policies\Microsoft\Windows

LAPS Setting enumeration

HKEY_LOCAL_MACHINE\Software\Policies



```
// ServiceDll's can be at the following locations
//  - HKLM\\SYSTEM\\CurrentControlSet\\Services\\ ! ServiceDll
//    - Ex: DoSvc on Win10
//  - HKLM\\SYSTEM\\CurrentControlSet\\Services\\Parameters ! ServiceDll
//    - Ex: DnsCache on Win10

string? path = null;

try
{
    path = RegistryUtil.GetStringValue(RegistryHive.LocalMachine, $"SYSTEM\\CurrentControlSet\\Services\\{serviceName}\\Parameters", "ServiceDll");
}
catch
{
}

if (path != null)
    return path;

try
{
    path = RegistryUtil.GetStringValue(RegistryHive.LocalMachine, $"SYSTEM\\CurrentControlSet\\Services\\{serviceName}", "ServiceDll");
}
catch
{
    // ignored
}

return path;
}

private string? GetServiceCommandFromRegistry(string serviceName)
{
try
{
    return RegistryUtil.GetStringValue(RegistryHive.LocalMachine, $"SYSTEM\\CurrentControlSet\\Services\\{serviceName}", "ImagePath");
}
catch
{
    return null;
}
}
```

Service Creation Example (T1569.002/T1574.011)

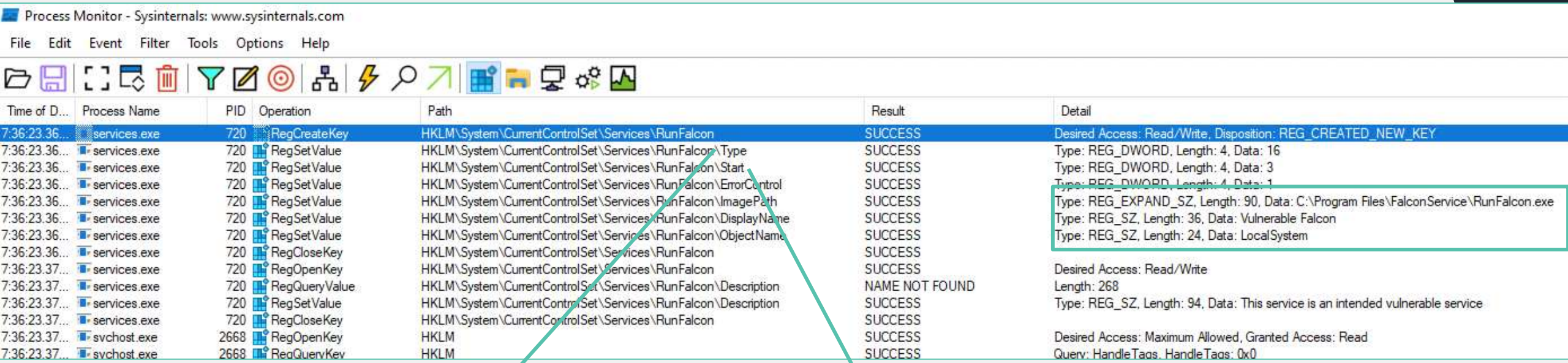
```
PS D:\Lab data\Day3\exercise-d3e3> .\Install-Service.ps1

Status   Name               DisplayName
------   ----               -----------
Stopped  RunFalcon          Vulnerable Falcon

PSPath            : Microsoft.PowerShell.Core\FileSystem::C:\Program Files\FalconService\
PSParentPath      : Microsoft.PowerShell.Core\FileSystem::C:\Program Files
PSChildName       : FalconService
PSDrive           : C
PSProvider        : Microsoft.PowerShell.Core\FileSystem
PSIsContainer     : True
Name              : FalconService
FullName          : C:\Program Files\FalconService
Parent            : Program Files
Exists            : True
```

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of D...	Process Name	PID	Operation	Path	Result	Detail
7:36:23.36...	services.exe	720	RegCreateKey	HKLM\System\CurrentControlSet\Services\RunFalcon	SUCCESS	Desired Access: Read/Write, Disposition: REG_CREATED_NEW_KEY
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\Type	SUCCESS	Type: REG_DWORD, Length: 4, Data: 16
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\Start	SUCCESS	Type: REG_DWORD, Length: 4, Data: 3
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\ErrorControl	SUCCESS	Type: REG_DWORD, Length: 4, Data: 1
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\ImagePath	SUCCESS	Type: REG_EXPAND_SZ, Length: 90, Data: C:\Program Files\FalconService\RunFalcon.exe
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\DisplayName	SUCCESS	Type: REG_SZ, Length: 36, Data: Vulnerable Falcon
7:36:23.36...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\ObjectName	SUCCESS	Type: REG_SZ, Length: 24, Data: LocalSystem
7:36:23.36...	services.exe	720	RegCloseKey	HKLM\System\CurrentControlSet\Services\RunFalcon	SUCCESS	
7:36:23.37...	services.exe	720	RegOpenKey	HKLM\System\CurrentControlSet\Services\RunFalcon	SUCCESS	Desired Access: Read/Write
7:36:23.37...	services.exe	720	RegQueryValue	HKLM\System\CurrentControlSet\Services\RunFalcon\Description	NAME NOT FOUND	Length: 268
7:36:23.37...	services.exe	720	RegSetValue	HKLM\System\CurrentControlSet\Services\RunFalcon\Description	SUCCESS	Type: REG_SZ, Length: 94, Data: This service is an intended vulnerable service
7:36:23.37...	services.exe	720	RegCloseKey	HKLM\System\CurrentControlSet\Services\RunFalcon	SUCCESS	
7:36:23.37...	svchost.exe	2668	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
7:36:23.37...	svchost.exe	2668	RegQueryKey	HKLM	SUCCESS	Query: HandleTags, HandleTags: 0x0

Service Creation Example (T1569.002/T1574.011)



ServiceType	
Adapter	4
FileSystemDriver	2
InteractiveProcess	256
KernelDriver	1
RecognizerDriver	8
Win32OwnProcess	16
Win32ShareProcess	32

ServiceStartMode	
Automatic	2
Boot	0
Disabled	4
Manual	3
System	1

Adding a local user via the registry

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
7:03:5...	CreateHiddenA...	13944	RegOpenKey	HKLM\SAM\SAM\Domains\Account\Users	SUCCESS	Desired Access: All Access
7:03:5...	CreateHiddenA...	13944	RegOpenKey	HKLM\SAM\SAM\Domains\Account\Users\Names	SUCCESS	Desired Access: All Access
7:03:5...	lsass.exe	752	RegOpenKey	HKLM\SAM\SAM\DOMAINS\Account\Users\Names\HellowATTCKCon$	NAME NOT FOUND	Desired Access: Read
7:03:5...	lsass.exe	752	RegOpenKey	HKLM\SAM\SAM\DOMAINS\Account\Users	SUCCESS	Desired Access: Read
7:03:5...	lsass.exe	752	RegQueryValue	HKLM\SAM\SAM\Domains\Account\Users\(Default)	SUCCESS	Type: REG_RESOURCE_REQUIREMENTS_LIST, Length: 0
7:03:5...	lsass.exe	752	RegCloseKey	HKLM\SAM\SAM\Domains\Account\Users	SUCCESS	
7:03:5...	lsass.exe	752	RegCreateKey	HKLM\SAM\SAM\DOMAINS\Account\Users	SUCCESS	Desired Access: Write, Disposition: REG_OPENED_EXISTING_KEY
7:03:5...	lsass.exe	752	RegSetValue	HKLM\SAM\SAM\Domains\Account\Users\(Default)	SUCCESS	Type: REG_QWORD, Length: 0
7:03:5...	lsass.exe	752	RegCloseKey	HKLM\SAM\SAM\Domains\Account\Users	SUCCESS	
7:03:5...	lsass.exe	752	RegCreateKey	HKLM\SAM\SAM\DOMAINS\Account\Users\Names\HellowATTCKCon$	SUCCESS	Desired Access: Write, Disposition: REG_CREATED_NEW_KEY
7:03:5...	lsass.exe	752	RegSetValue	HKLM\SAM\SAM\Domains\Account\Users\Names\HellowATTCKCon$\(Default)	SUCCESS	Type: <Unknown: 1008>, Length: 0
7:03:5...	lsass.exe	752	RegCloseKey	HKLM\SAM\SAM\Domains\Account\Users\Names\HellowATTCKCon$	SUCCESS	

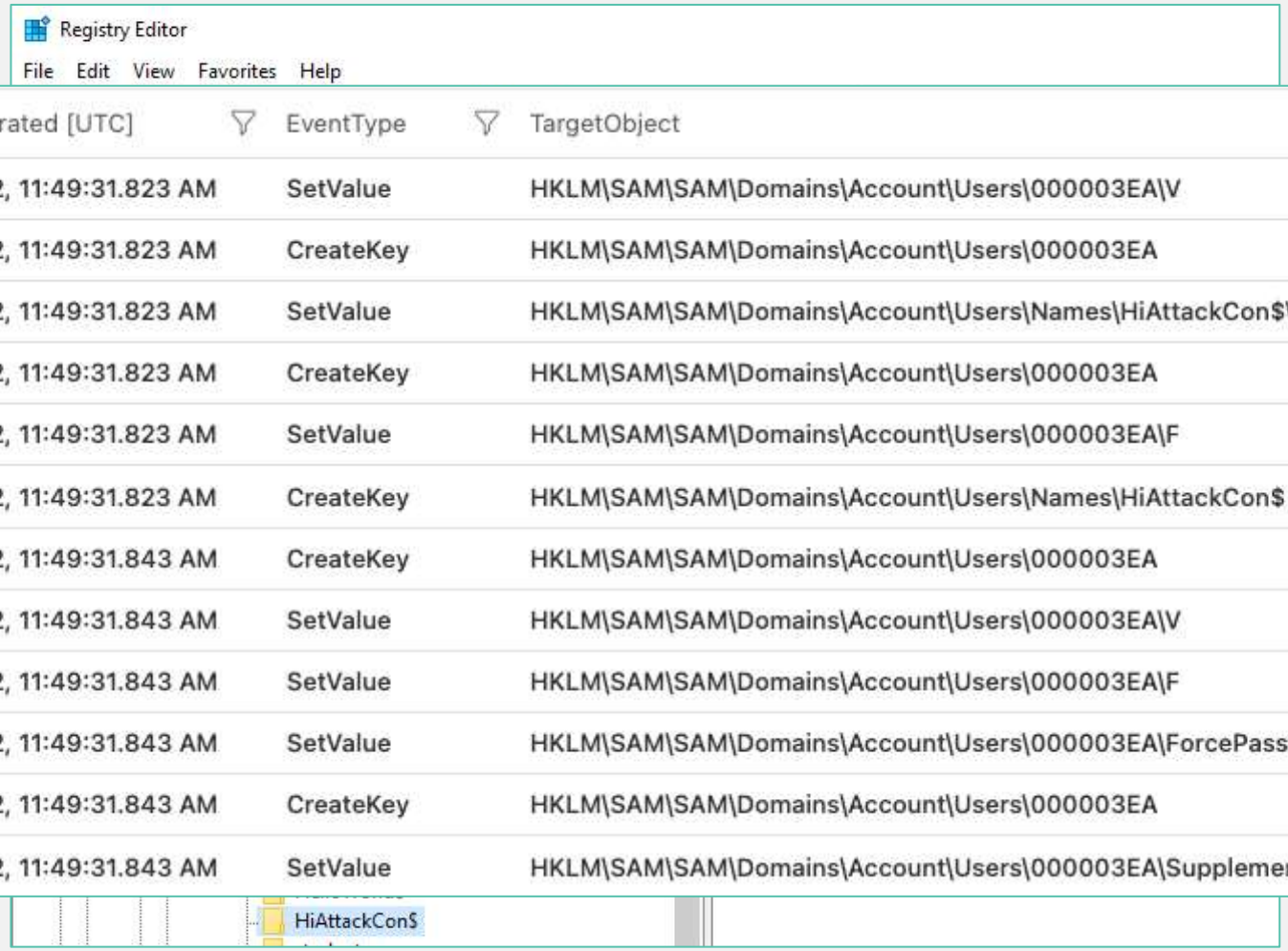
Process Monitor X v2.0 Beta 1

File Edit View Search Event Monitor Options Tab Help

Events

#	Time	Event	PID	Process Name	TID	Details
103655 *	03/24/22 18:58:44.037065	Registry/QueryValue	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665488; Status: 0xC0000034; Index: 2; KeyHandle: 0xFFFF9800519054C0; KeyName: DirectoryServiceExtPt;
103656 *	03/24/22 18:58:44.037073	Registry/Close	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665507; Status: 0x00000000; Index: 0; KeyHandle: 0xFFFF9800519054C0;
103657 *	03/24/22 18:58:44.037108	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665533; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Builtin\Groups\Names\HelloATTCKcon$;
103658 *	03/24/22 18:58:44.037122	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665550; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Builtin\Aliases\Names\HelloATTCKcon$;
103659 *	03/24/22 18:58:44.037141	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665562; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Builtin\Users\Names\HelloATTCKcon$;
103660 *	03/24/22 18:58:44.037154	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665581; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Account\Groups\Names\HelloATTCKcon$;
103661 *	03/24/22 18:58:44.037165	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665594; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Account\Aliases\Names\HelloATTCKcon$;
103662 *	03/24/22 18:58:44.037176	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665605; Status: 0xC0000034; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Account\Users\Names\HelloATTCKcon$;
103663 *	03/24/22 18:58:44.037188	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665618; Status: 0x00000000; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Account\Users;
103664 *	03/24/22 18:58:44.037211	Registry/QueryValue	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665627; Status: 0x00000000; Index: 2; KeyHandle: 0xFFFF980043EB8DF0;
103665 *	03/24/22 18:58:44.037251	Registry/Close	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665685; Status: 0x00000000; Index: 0; KeyHandle: 0xFFFF980043EB8DF0;
103666 *	03/24/22 18:58:44.037272	Registry/Open	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665700; Status: 0x00000000; Index: 0; KeyHandle: 0xFFFF980042599530; KeyName: DOMAINS\Account\Groups\00000201;
103667 *	03/24/22 18:58:44.037293	Registry/QueryValue	752 (0x2F0)	lsass.exe	7984 (0x1F30)	InitialTime: 03/24/22 18:58:37.665712; Status: 0x00000000; Index: 2; KeyHandle: 0xFFFF98004D20AE00; KeyName: C;

Adding a local user via the registry



TimeGenerated [UTC]	EventType	TargetObject	Details	Image
3/24/2022, 11:49:31.823 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\V	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.823 AM	CreateKey	HKLM\SAM\SAM\Domains\Account\Users\000003EA	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.823 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\Names\HiAttackCon$\(Default)	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.823 AM	CreateKey	HKLM\SAM\SAM\Domains\Account\Users\000003EA	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.823 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\F	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.823 AM	CreateKey	HKLM\SAM\SAM\Domains\Account\Users\Names\HiAttackCon$	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	CreateKey	HKLM\SAM\SAM\Domains\Account\Users\000003EA	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\V	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\F	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\ForcePasswordReset	Binary Data	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	CreateKey	HKLM\SAM\SAM\Domains\Account\Users\000003EA	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
3/24/2022, 11:49:31.843 AM	SetValue	HKLM\SAM\SAM\Domains\Account\Users\000003EA\SupplementalCredentials	Binary Data	C:\Windows\system32\lsass.exe

How do you get this data?

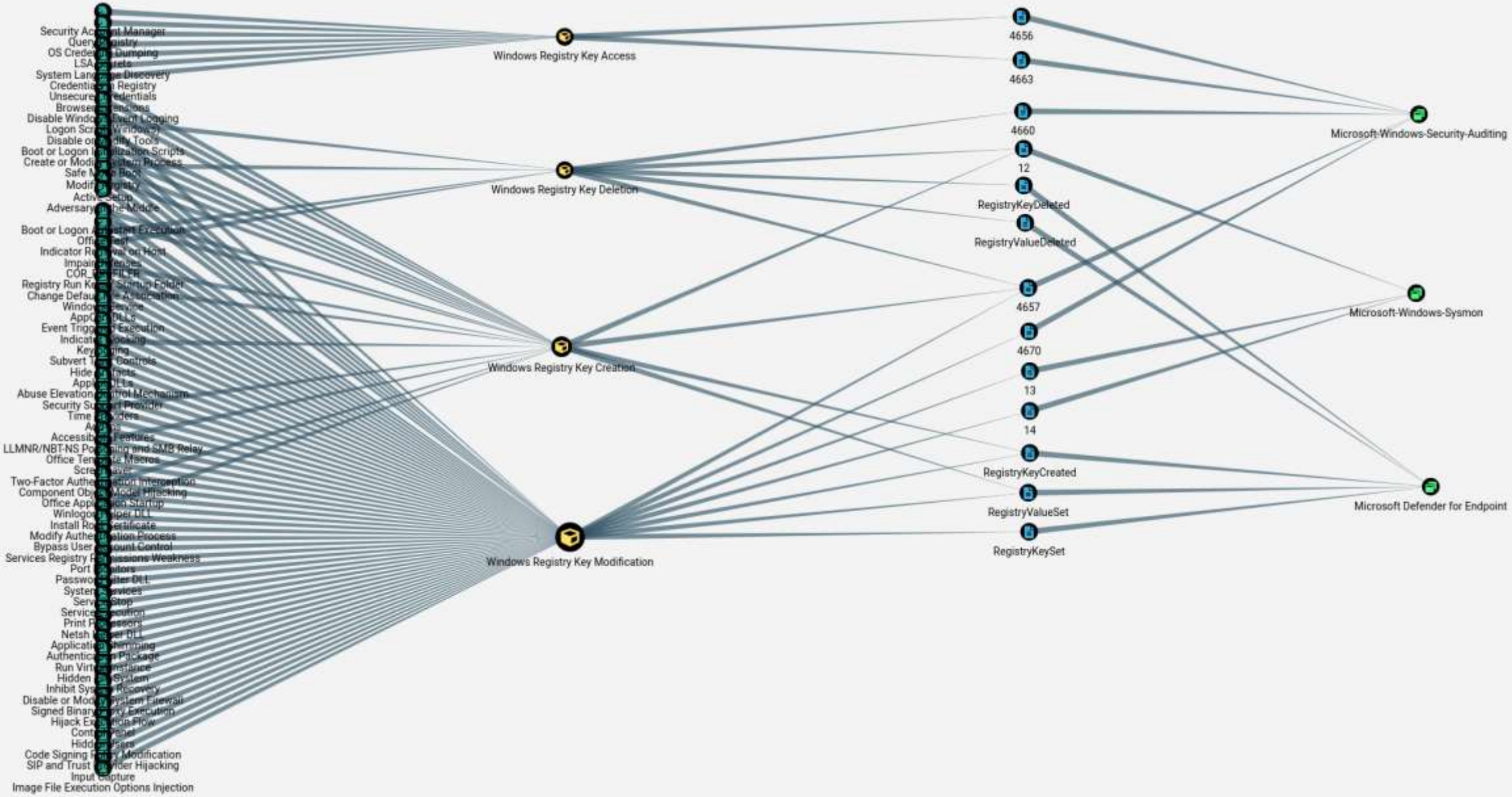
- Windows native, possible but complicated via SACLs (read, create, modify and delete)
- Sysmon (create, modify and delete)
- EDR Solutions (most often: create, modify and delete)

While researching / hunting:

- ETW (Microsoft-Windows-Kernel-Registry) (read, create, modify and delete)
- State based or live monitoring via tools; Procmon, ProcMonV2, Autoruns, osquery, Velociraptor etc.


https://github.com/OTRF/Set-AuditRule

ATT&CK techniques referenced and their telemetry



Process Access

Component number 18 in ATT&CK in references

Objectives

What is Process Access?

- Process
- Objects
- Handles
- Access

How can Defenders leverage this telemetry?

ATT&CK techniques referenced

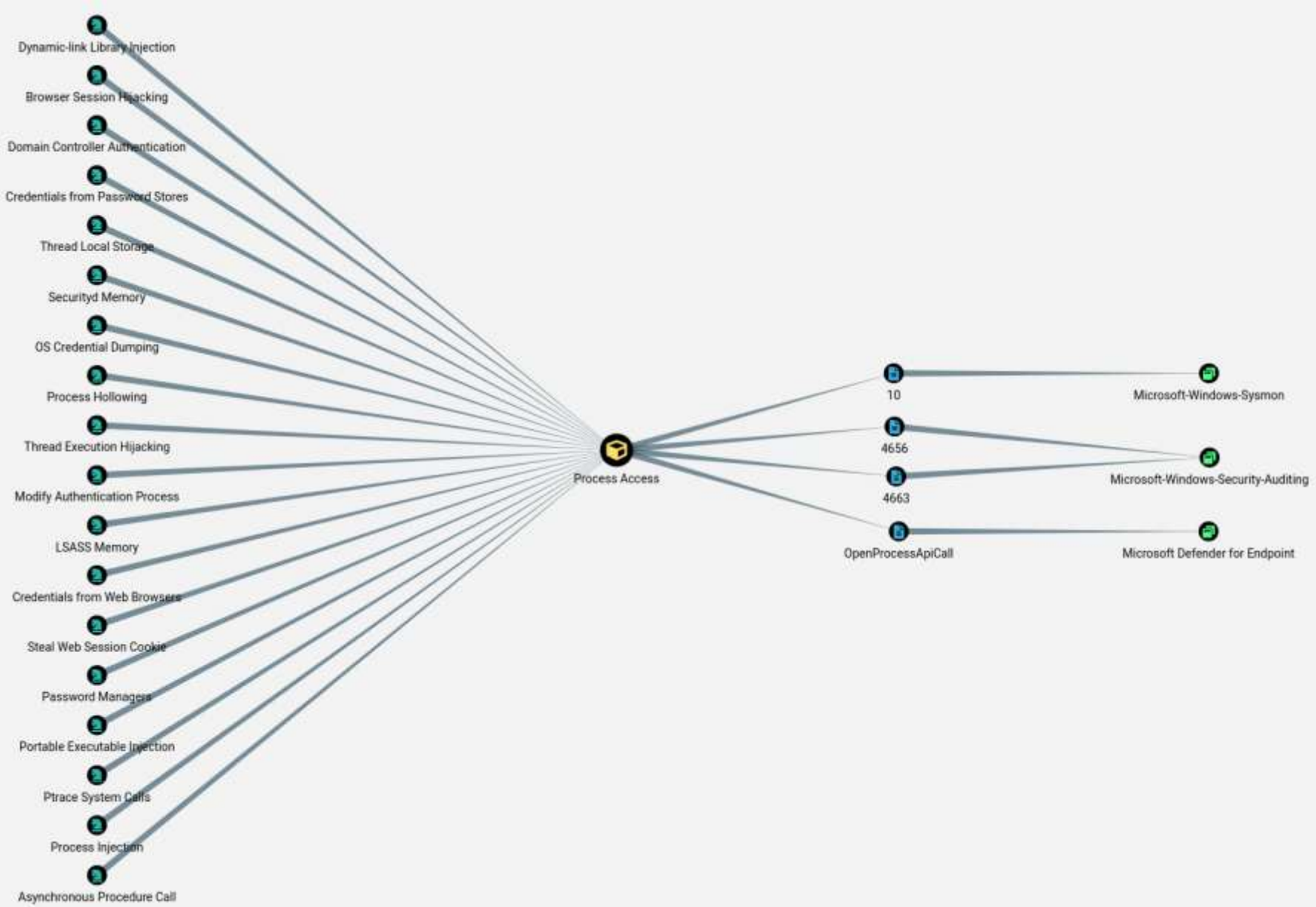
There are 18 sub-techniques/techniques that reference the Process Access data component.

Process Access is a part of the Process data source:

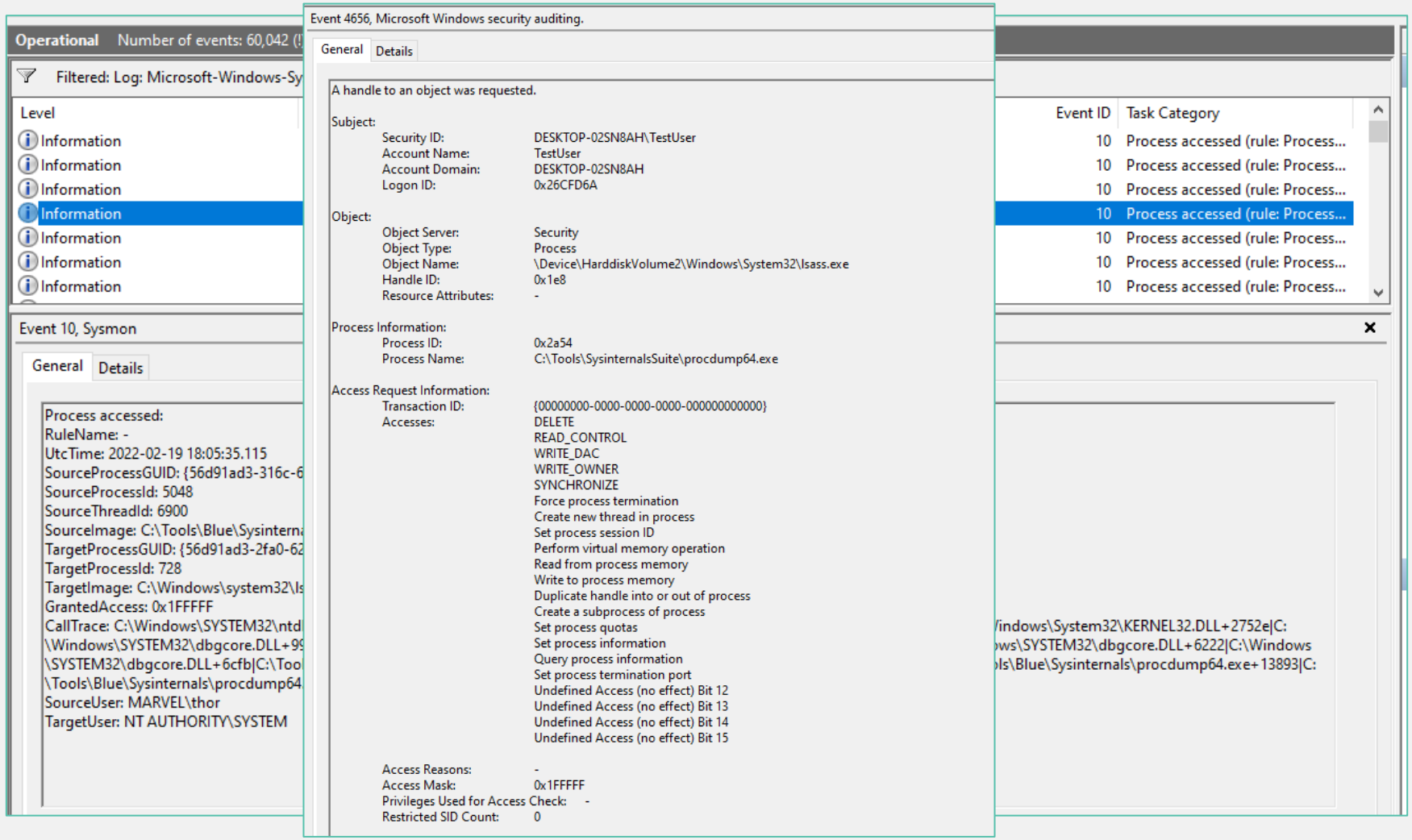
- Process: OS API Execution
- **Process: Process Access**
- Process: Process Creation
- Process: Process Metadata

It is very likely you'll be able to cover way more techniques than the ones currently mapped with this data source.


ATT&CK techniques referenced and their telemetry



What is Process Access?



What is Process Access?

Processes

- A container that hosts resources for a running instance of a program
- Backed by **EPROCESS** kernel structure
- **Process** type kernel object

Objects

- Data structures that need to be shared/protected and reflect some resource
- Processes, Files, Registry, etc
- Requires programs in user-mode to obtain a handle for access

Handles

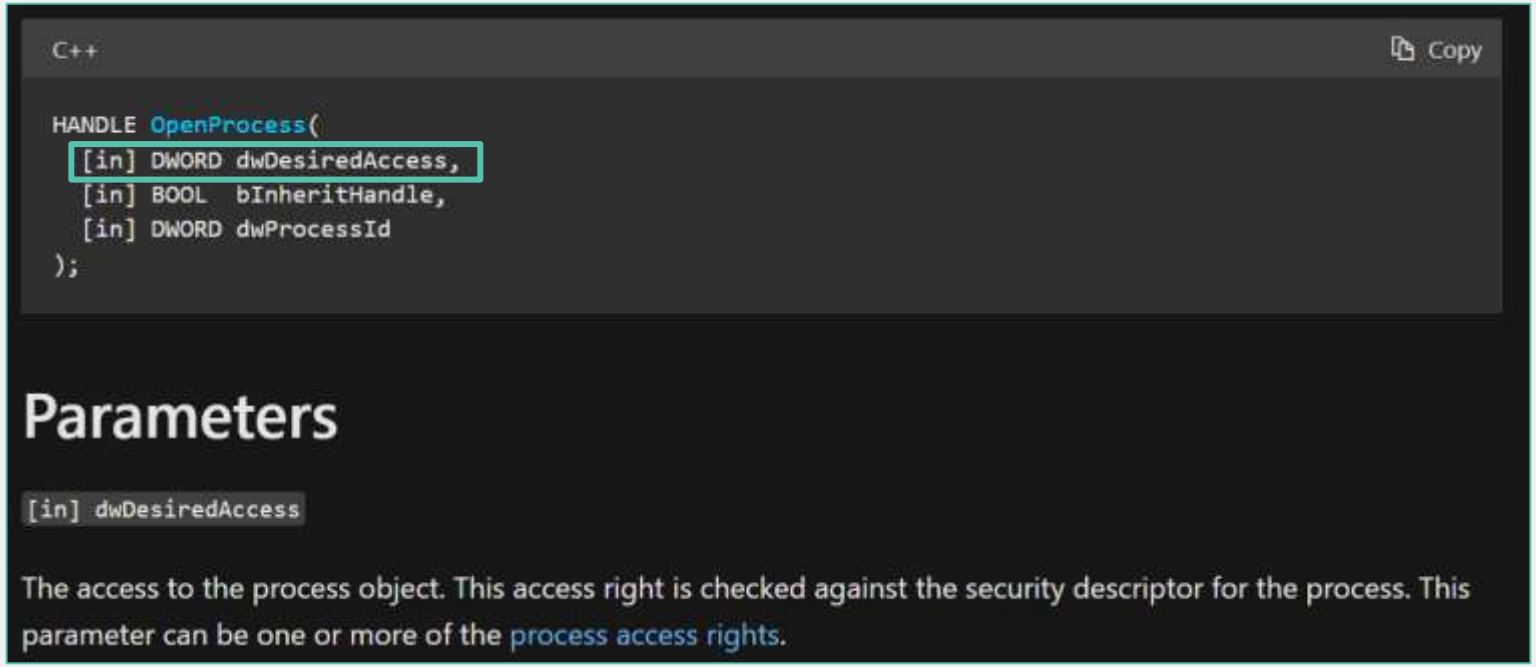
- Allows objects to be **shared** across processes
- An entry within an internal table
- Used to regulate access to a target object
- Once a handle is obtained, the source user's access is limited to what they requested


OpenProcess

Access

- Each object type has a set of access rights that allows the requestor to perform a set of actions

OpenProcess



C++ Copy

```
HANDLE OpenProcess(
  [in] DWORD dwDesiredAccess,
  [in] BOOL  bInheritHandle,
  [in] DWORD dwProcessId
);
```

Parameters

`[in] dwDesiredAccess`

The access to the process object. This access right is checked against the security descriptor for the process. This parameter can be one or more of the process access rights.

Object Access Rights

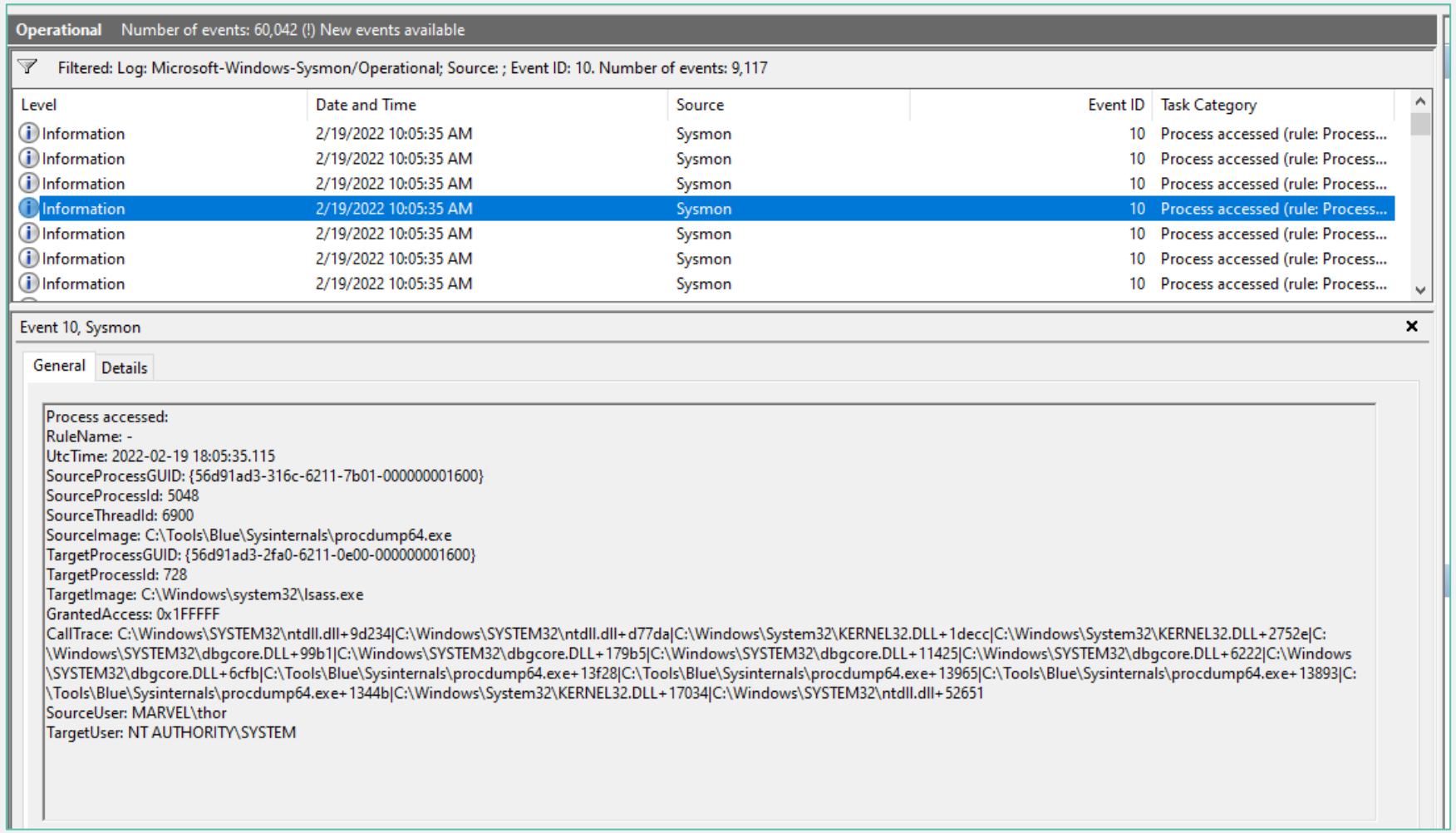
Process Access Rights

PROCESS_CREATE_PROCESS (0x0080)	Required to use this process as the parent process with PROC_THREAD_ATTRIBUTE_PARENT_PROCESS.
PROCESS_CREATE_THREAD (0x0002)	Required to create a thread in the process.
PROCESS_DUP_HANDLE (0x0040)	Required to duplicate a handle using DuplicateHandle.
PROCESS_QUERY_INFORMATION (0x0400)	Required to retrieve certain information about a process, such as its token, exit code, and priority class (see OpenProcessToken).

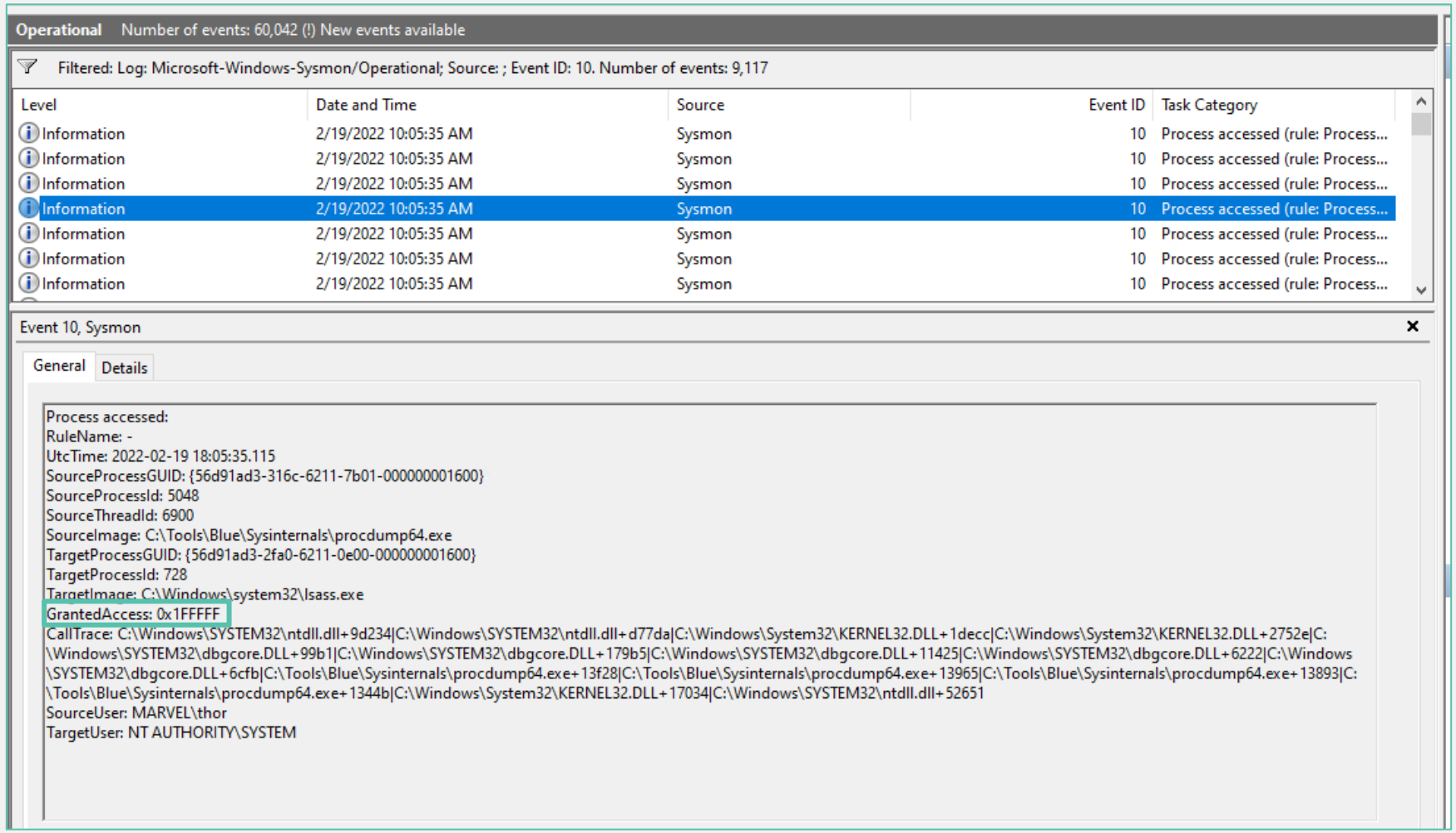
Registry Access Rights

KEY_ALL_ACCESS (0xF003F)	Combines the STANDARD_RIGHTS_REQUIRED, KEY_QUERY_VALUE, KEY_SET_VALUE, KEY_CREATE_SUB_KEY, KEY_ENUMERATE_SUB_KEYS, KEY_NOTIFY, and KEY_CREATE_LINK access rights.
KEY_CREATE_LINK (0x0020)	Reserved for system use.
KEY_CREATE_SUB_KEY (0x0004)	Required to create a subkey of a registry key.
KEY_ENUMERATE_SUB_KEYS (0x0008)	Required to enumerate the subkeys of a registry key.


How can Defenders leverage this telemetry?



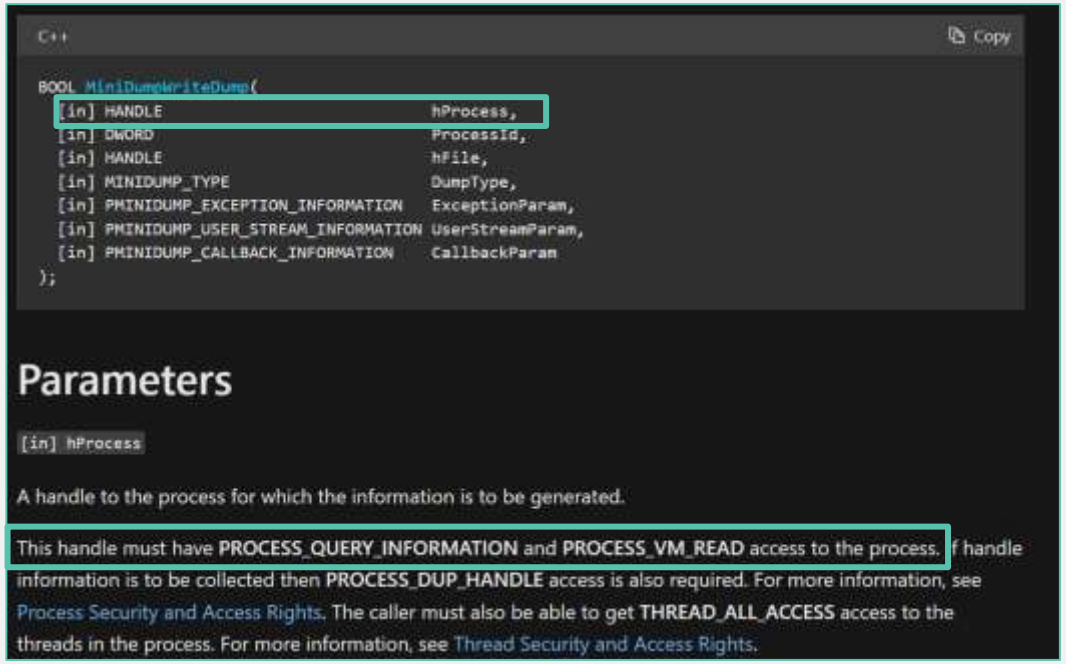
Determining Rights



Determining Rights

Process wants to implement some function (*MiniDumpWriteDump*) on another process

- Identify Function



C++ Copy

```
BOOL MiniDumpWriteDump(
  [in] HANDLE                            hProcess,
  [in] DWORD                             ProcessId,
  [in] HANDLE                            hFile,
  [in] MINIDUMP_TYPE                     DumpType,
  [in] PMINIDUMP_EXCEPTION_INFORMATION   ExceptionParam,
  [in] PMINIDUMP_USER_STREAM_INFORMATION UserStreamParam,
  [in] PMINIDUMP_CALLBACK_INFORMATION    CallbackParam
);
```

Parameters

`[in] hProcess`

A handle to the process for which the information is to be generated.

This handle must have **PROCESS_QUERY_INFORMATION** and **PROCESS_VM_READ** access to the process. f handle information is to be collected then **PROCESS_DUP_HANDLE** access is also required. For more information, see Process Security and Access Rights. The caller must also be able to get **THREAD_ALL_ACCESS** access to the threads in the process. For more information, see Thread Security and Access Rights.

Identify Access Needed

- PROCESS_QUERY_INFORMATION (0X0400) / PROCESS_QUERY_LIMITED_INFORMATION (0X1000) / PROCESS_VM_READ (0X0010) == 0x1410
- Could grant higher, but not less.

Bitwise Operations + Access Rights?

Bitwise Operations

- AND, OR, XOR, XAND
- We only care about **AND** today :)
- Can be used within Sysmon ID 10's **GrantedAccess** flag to see if access was requested.
- Checking the bitmask for when 1's match.

Identify if an access exists within the access requested

- Requested Right (*0x1FFFFF/PROCESS_ALL_ACCESS*) to process
- Using **MiniDumpWriteDump(0x1410)**
- Does **((0x1FFFFF AND 0x1410) == 0X1410)?**

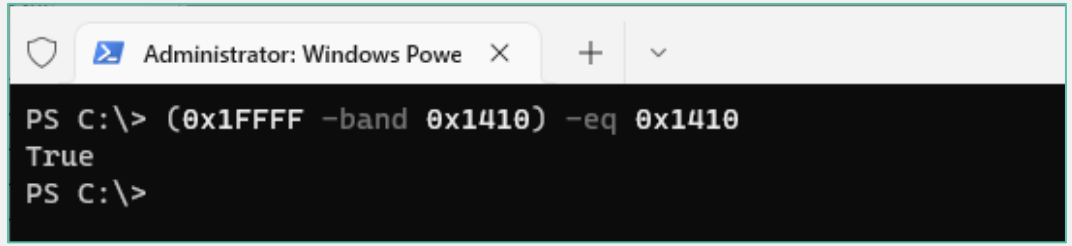
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0

(*0x1FFFFF*)
(0x1410)
(AND)
(0x1410)

- Does **((0x1FFFFF AND 0x1410) == 0X1410)? YES!**

Bitwise Operations + Access Rights?

Easy to do with PowerShell:



Easy to do with some analytical platforms:

Jupyter:

```
AND (a.process_granted_access & 5136) == 5136
```

Kusto:

```
DeviceEvents
| where ActionType  == "OpenProcessApiCall"
and FileName == "lsass.exe"
| extend DesiredAccess = extractjson("$.DesiredAccess", AdditionalFields)
| extend toint(DesiredAccess)
| where binary_and(DesiredAccess,5136) == 5136
```

Example – Dumping LSASS

MiniDumpWriteDump:

```
Process_Access_Lsass_Minimum_Rights_For_Minidump = spark.sql(
'''
SELECT
    a.host_name,
    a.process_name,
    a.process_target_name,
    a.process_granted_access
FROM sysmon_events a
JOIN sysmon_events b
ON a.process_name = b.process_name
AND b.event_id = 1
WHERE a.event_id = 10
AND a.process_target_name = "lsass.exe"
AND (a.process_granted_access & 5136) == 5136
'''
).show(10,False)
```

```
+------------------+-------------------------+-------------------+----------------------+
|host_name         |process_name             |process_target_name|process_granted_access|
+------------------+-------------------------+-------------------+----------------------+
|win10.marvel.local|procdump.exe             |lsass.exe          |2097151               |
```

5136 = 0x1410

https://docs.microsoft.com/en-us/windows/win32/api/minidumpapiset/nf-minidumpapiset-minidumpwritedump

Example – Dumping LSASS

ReadProcessMemory:

```
Process_Access_Lsass_Broad = spark.sql(
'''
SELECT
    a.host_name,
    a.process_name,
    a.process_target_name,
    a.process_granted_access
FROM sysmon_events a
JOIN sysmon_events b
ON a.process_name = b.process_name
AND b.event_id = 1
WHERE a.event_id = 10
AND a.process_target_name = "lsass.exe"
AND (a.process_granted_access & 4112) == 4112
'''
).show(10,False)
```

```
+------------------+------------+------------------+----------------------+
|host_name         |process_name|process_target_name|process_granted_access|
+------------------+------------+------------------+----------------------+
|win10.marvel.local|procdump.exe|lsass.exe          |4112                  |
```

4112 = 0x1010

https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-readprocessmemory

Example – Process Injection

Reflective DLL Injection:

```
ReflectiveDLL_ProcessInjection = spark.sql(
'''
SELECT
    b.process_path,
    b.process_target_name,
    b.process_target_id,
    b.thread_new_id,
    a.process_id,
    a.process_granted_access
FROM sysmon_events b
JOIN sysmon_events a
ON a.process_guid = b.process_guid
AND a.event_id = 10
AND (a.process_granted_access & 5178) == 5178

WHERE b.event_id = 8
AND NOT b.process_name = "csrss.exe"
'''
).show(10,False)
```

```
+-----------------------------------------------------+-----------------+----------------+-------------+----------+---------------------+
|process_path                                         |process_target_name|process_target_id|thread_new_id|process_id|process_granted_access|
+-----------------------------------------------------+-----------------+----------------+-------------+----------+---------------------+
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |3124            |7940         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |3124            |7940         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7388         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7388         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |5376         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |5376         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7640         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7640         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7332         |5452      |2047999              |
|c:\windows\system32\windowspowershell\v1.0\powershell.exe|notepad.exe      |7924            |7332         |5452      |2047999              |
+-----------------------------------------------------+-----------------+----------------+-------------+----------+---------------------+
```

5178 = 0x143A (CreateRemoteThread)


https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread

Wrapping up

- Remember to look at the bigger picture
- Understand what you are detecting and HOW you are detecting it
- Be aware of your assumptions about attacker techniques and your own visibility
- Focus on the most resilient data source / data component for detections
- Know your tools, understand their strengths and weaknesses

And remember,

Understanding your telemetry is the key to unlocking its true potential.

These projects are a huge asset for a detection engineer:

- Windows-API-to-SysmonEvents (https://github.com/jsecurity101/Windows-API-To-Sysmon-Events)
- Sysmon-modular (https://github.com/olafhartong/sysmon-modular)
- OSSEM Project (https://github.com/OTRF/OSSEM)

- Sysinternals Tools (https://docs.microsoft.com/en-us/sysinternals/downloads/)
- Pavel Yosifovich Tools (https://github.com/zodiacon/AllTools)
- API Monitor (http://www.rohitab.com/apimonitor)
- FRIDA (https://Frida.re)
- Ghidra (https://ghidra-sre.org)


Thank you! Questions ?

olaf@falconforce.nl
jonny.johnson@redcanary.com

https://falconforce.nl
https://redcanary.com

@olafhartong
@falconforceteam
@jsecurity101
@redcanary

https://linkedin.com/in/olafhartong
https://www.linkedin.com/in/jonathan-johnson-7aa937135/