

Hunting for Adversary's Schedule

After successfully getting their foothold into their desired enterprise, **Do you think adversary want to loose their access because they accidentally hit CTRL + C?** Lets *schedule* some time to hunt one of adversaries favorite **Persistence** technique.

Today's Schedule

- A Window into Shcheduled Task
- Threat Hunting - Its quirks and features
- Lets Get to know the Data
- Detectioin Engineering
- What we learned - A recap

A Window INTO - Scheduled Task

What are scheduled tasks?

According to MSDN "A scheduled task is basically a codeunit that runs logic in a background session at a specific time."

Some main components Triggers, Actions, Principals, Settings, Registry Information and Data. Detailed objects available can be found at [Task Schedule Objects](#)

TASK

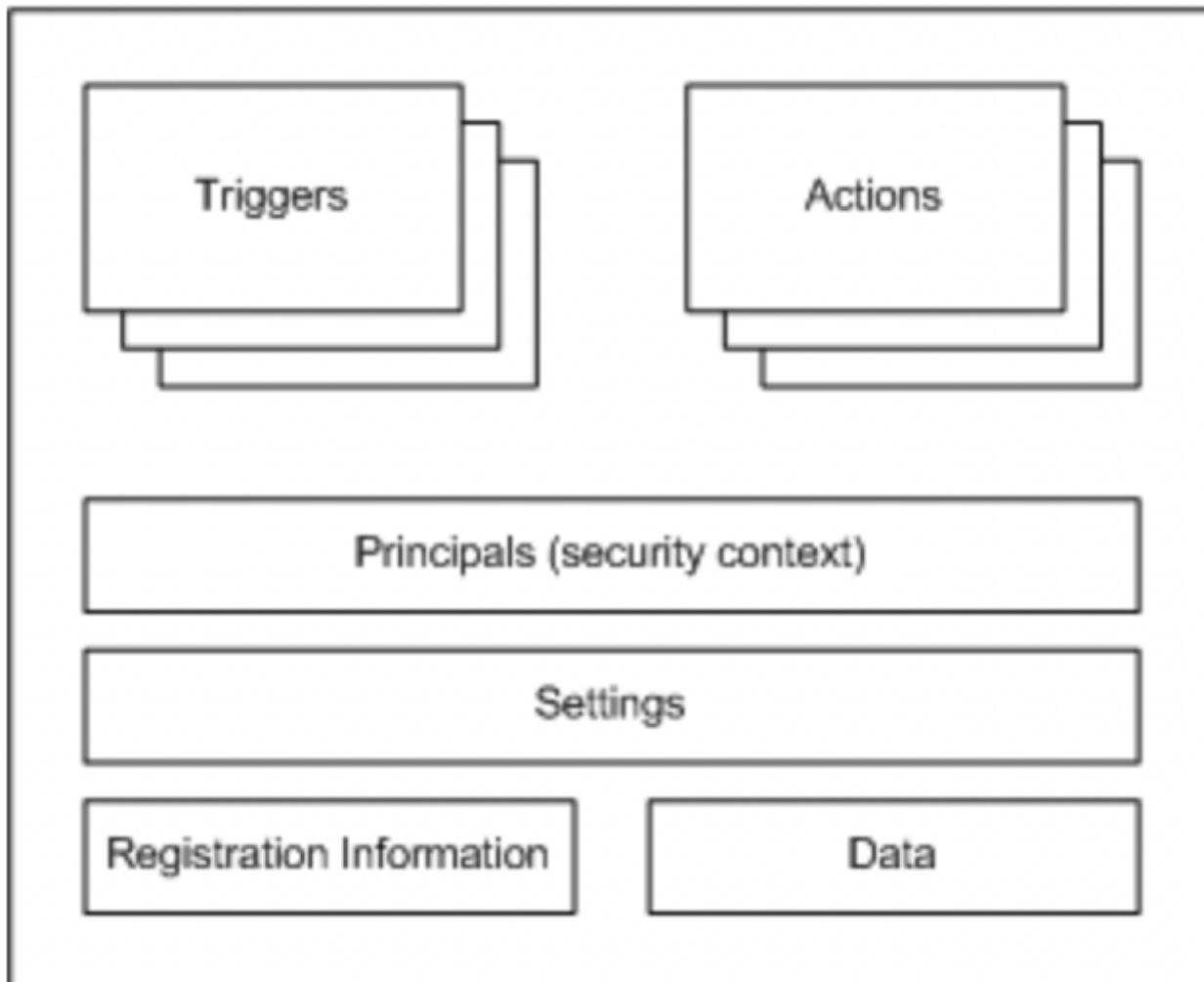


Image courtesy [Microsoft Docs](#)

Who can create scheduled tasks?

To create/view/run/change scheduled tasks you **must be** an administrator or **member of the administrator's group** AND the task has **necessary permissions**. Additionally, you can grant users with SeBatchLogonRight permissions. Also, from MSDN " Schedule Task Wizard does not verify the password that you type for the user account that the process will run under. Make sure that you type the correct password."

Why attackers use scheduled tasks?

Scheduled Task is one of the common persistence techniques used in Intrusion operations and Red Teaming/Pentesting activities. Once a perpetrator/red teamer/pentester established their foothold, they want to **maintain** their access surviving reboots, network issues, and even pulling the cords.

Which stage of Mandiant Lifecycle/MITRE/Kill chain Scheduled task is used?

In Mandiant Lifecycle, scheduled tasks revolve around the cycle of Maintain Presence and Escalate Privileges. In MITRE lingo, it is spanned across three different Tactics (Persistence, Execution, Privilege Escalation) depending on what a perpetrator wants to achieve. For mighty Cyber Kill Chain, this can be in Exploitation, installation, and Command and Control.

Inspector Gadget - Threat Hunting



Image Courtesy [Stickpng](#)

Before we learn about Threat Hunting, let's define it

What is Threat Hunting?

Threat Hunting is a process of being proactive of unveiling unknown-knowns and unknown-unknowns to better our security posture

In Threat Hunting, hypothesis creation is generally a first step to guide us in the process

Hypothesis Creation

"Hypotheses focus the research interest on the most important questions and thus generate a structure which then leads to the decision of which empiric tools to choose for research" [1\(\)](https://www.acad-write.org/mentor/how-to-write-a-hypothesis/). With this statement in mind, we will formulate our hypothesis using who, what, where, when, why and How. [2\(\)](https://youtu.be/vZgI00UcATw?t=26314).

In threat Hunting, we create Broad and Narrow Hypothesis (This is not a standard or a law but it provides a nice flow into starting our hunts). The Broad hypothesis can be used as our epic topic for the week/month. It is broad for a reason. The narrow hypothesis is what we want to hunt for today. You can think of the broad hypothesis as a Tactic and the narrow hypothesis as a tool/technique/procedure.

Broad Hypothesis

Adversaries use persistence mechanisms to stay connected.

Narrow hypothesis

For creating our narrow hypothesis, let's answer some questions

1. Who - Inside Threats/Adversaries/pentesters/read teamers. (answers **who** is our **person of interest**)
2. What - Scheduled tasks (various persistence mechanisms, you can use MITRE to get known ones) (answers **what** we would like to **hunt**)
3. Where - most of the artifacts be related to endpoint (EDR, sysmon, windows event logs) (answers **where** do find the **data** to prove/disprove our **hypothesis**)
4. When - We can hunt for all retained logs (answers the **timeframe** we will **focus** our hunt)
5. Why - To check for persistence in our environment (answers **why** are we **doing** this)
6. How - to identify a way of modeling our searches and interpret their results

With that in mind, our narrow hypothesis could be along the lines of

Adversaries might **create/change scheduled tasks** on local/remote endpoints using GUI/command line tools to **execute** additional/new executables/hta files/js files for **maintaining persistence** in our environment.

Familiarize yourself with available data

Having a working hypothesis is great BUT we need some data to perform this hunt.

How to determine what data is required?

There are several answers to the question depending on Time, Team/org capabilities among other things. The easiest and most tested way is to take a look what our dear friends at MITRE did. we go to [MITRE](#) and check the section named "Detection" to give us the Data components required to hunt for.

We see that we may need

- command execution and their arguments
- file creation/modification events
- process and scheduled creation events.

Let's check if we have any of those in our environment. using splunk.

query

```
index=* | dedup index | table index
```

Above query will

- query all indeces with wilcard *
- deduplicate them using **dedup** command
 - asking to present it in tabular form.

The screenshot shows a Splunk search interface. The search bar contains the query: `index=* | dedup index | table index`. The search results show 7 events found before 17/07/2022 21:06:20.000. The results are presented in a table with columns for index, source, type, host, offset, _id, and _index. The 'Statistics' tab is selected. The left sidebar shows a list of indices: index, zeek, osquery, sysmon, wineventlogs, hmail, velociraptor, and main. The 'Events (7)' button is highlighted with a red box. The top right corner shows 'All time' and a search icon. The bottom navigation includes 'Events (7)', 'Patterns', 'Statistics (7)', 'Visualization', '10 Per Page', 'Format', and 'Preview'.

We can see endpoint (osquery, windows event logs, sysmon) logs that are required for our hypothesis. But we need specific ones like command line arguments, process creation events, etc. So our next step is to figure out if we have those events in sysmon and windows events and examine fields. For this, let's take look at a sample log for sysmon and windows events.

Sysmon sample log with query

```
index=sysmon
```

```

[ [-]
  @timestamp: 2022-03-12T22:52:54.263Z
  @version: 1
  agent: { [-]
    ephemeral_id: 89aebf72-73f7-4535-899b-ee8d39c57953
    hostname: files
    id: 9809b5ad-f9bd-4cd8-a7be-abb90a8a4c39
    name: files
    type: winlogbeat
    version: 7.16.2
  }
  ecs: { [-]
    version: 1.12.0
  }
  event: { [-]
    action: File created (rule: FileCreate) ━━━━━━ Sysmon Tag/
    category: [ [-] Task Category
      file
    ]
    code: 11 ━━━━━━ Sysmon ID
    created: 2022-03-12T22:52:55.589Z
    kind: event
    module: sysmon
    provider: Microsoft-Windows-Sysmon ━━━━━━ Log provider
    type: [ [-]
      creation
    ]
  }
  file: { [-]
    directory: C:\Windows\ServiceProfiles\LocalService\AppData\Local
    extension: dat
    name: lastalive0.dat
    path: C:\Windows\ServiceProfiles\LocalService\AppData\Local\lastalive0.dat
  }
  host: { [-]
    architecture: x86_64
    hostname: files
    id: 2be492d7-af67-4e72-a1db-9c074112495c
    ip: [ [-]
      fe80::5bd:f44d:3be6:38a2
      172.16.50.110
      fe80::5efe:ac10:326e
    ]
    mac: [ [-]
      06:e5:dd:67:f2:a4
      00:00:00:00:00:00:e0
    ]
    name: files.magnuntempus.financial
    os: { [-]
      build: 14393.4770
      family: windows
      kernel: 10.0.14393.4770 (rs1_release.211101-1440)
      name: Windows Server 2016 Datacenter
      platform: windows
      type: windows
      version: 10.0
    }
  }
  log: { [-]
    level: information
  }
  message: File created:
RuleName: technique_id=T1047,technique_name=File System Permissions Weakness
UtcTime: 2022-03-12 22:52:54.263
ProcessGuid: {2BE492D7-E6DA-622C-1300-000000001702}
ProcessId: 992
Image: C:\Windows\System32\svchost.exe
TargetFilename: C:\Windows\ServiceProfiles\LocalService\AppData\Local\lastalive0.dat
CreationUtcTime: 2022-03-12 18:30:50.825
User: NT AUTHORITY\LOCAL SERVICE
  process: { [-]
    entity_id: {2BE492D7-E6DA-622C-1300-000000001702}
    executable: C:\Windows\System32\svchost.exe
    name: svchost.exe
    pid: 992
  }
  rule: { [-]
    name: technique_id=T1047,technique_name=File System Permissions Weakness
  } ← From sysmon config file
  tags: [ [-]
    sysmon
    beats_input_codec_plain_applied
  ]
  winlog: { [-]
    api: wineventlog
    channel: Microsoft-Windows-Sysmon/Operational
    computer_name: files.magnuntempus.financial
    event_data: { [+]
    }
    event_id: 11
    opcode: Info
    process: { [+]
    }
    provider_guid: {5770385F-C22A-43E0-BF4C-06F5698FFBD9}
    provider_name: Microsoft-Windows-Sysmon
    record_id: 84363
    task: File created (rule: FileCreate)
    -----
  }

```

Sysmon Tag/
Task Category

Sysmon ID

Log provider

Information related to hosts

how
we see in windows
event logs

From sysmon config file

```
user: 1 L+J
}
version: 2
}
```

From the screenshot, we see event.action and event.code interesting as [sysmon](#) documentation states them as events.

One caveat in going with this method is that we do not know all sysmon logs will have same/similar fields. So, let see what field that may appear in all the logs and test it.

The screenshot shows a Splunk search interface. The search bar contains the query "1 index=sysmon". Below the search bar, a message indicates "1,515,911 events (before 17/07/2022 22:23:28.000) No Event Sampling". The interface includes standard Splunk navigation buttons like Job, Fast Mode, and a search icon.

Events (1,515,911) Patterns Statistics Visualization

Total number of sysmon logs

The screenshot shows a Splunk search interface. The search bar contains the query "1 index=sysmon NOT event.code=". Below the search bar, a message indicates "689 events (before 17/07/2022 22:26:15.000) No Event Sampling". The interface includes standard Splunk navigation buttons like Job, Fast Mode, and a search icon.

Total number of logs without event id

99.999% of the logs have event id, so, it is good to work with this data. We can check the rest of the data for edge cases later in our threat hunting process if needed.

Now, lets check what types of events we have in sysmon

query

```
index=sysmon | dedup event.action | table event.action ,event.code
```

Above query will

- Use sysmon Index
- Deduplicate even.action field
- Presenting event.action and event.id in tabular form

The screenshot shows a Splunk search interface. The search bar contains the query "1 index=sysmon | dedup event.action | table event.action ,event.code". Below the search bar, a message indicates "19 events (before 17/07/2022 22:36:45.000) No Event Sampling". The interface includes standard Splunk navigation buttons like Job, Fast Mode, and a search icon.

Events Patterns Statistics (19) Visualization

10 Per Page ▾ Format Preview ▾ < Prev 1 2 Next >

event.action	Sysmon Tasks/Tags	Sysmon Event IDs
File created (rule: FileCreate)		11
Dns query (rule: DnsQuery)		22
Registry value set (rule: RegistryEvent)		13
Registry object added or deleted (rule: RegistryEvent)		12
Image loaded (rule: ImageLoad)		7
Network connection detected (rule: NetworkConnect)		3
Process Create (rule: ProcessCreate)		1
Pipe Created (rule: PipeEvent)		17
Process terminated (rule: ProcessTerminate)		5
Process accessed (rule: ProcessAccess)		10

There are several windows event IDs, let's see what is relevant to us by checking [Microsoft's Auditing](#) and searching for scheduled tasks in the sidebar, we will see events produced when a certain action is performed on/by scheduled tasks.

Event 4698 S: A scheduled task was created.
Security auditing > Advanced security audit policies

Event 4699 S: A scheduled task was deleted.
Security auditing > Advanced security audit policies

Event 4700 S: A scheduled task was enabled.
Security auditing > Advanced security audit policies

Event 4701 S: A scheduled task was disabled.
Security auditing > Advanced security audit policies

Event 4702 S: A scheduled task was updated.
Security auditing > Advanced security audit policies

As we did with sysmon, let's check out sample windows event log to see what fields exist with the query

index=wineventlogs

```
12/03/2022  { [-] 
22:52:41.000  @timestamp: 2022-03-12T22:52:39.833Z
@version: 1
agent: { [-]
ephemeral_id: 8b92aba7-db0e-494f-80bb-4caed3aa2298
hostname: wkst03
id: ede521c-9b85-4b9c-a9df-65d46bd0af04
name: wkst03
type: winlogbeat
version: 7.16.2
}
ecs: { [-]
version: 1.12.0
}
event: { [-]
code: 130
created: 2022-03-12T22:52:40.917Z
kind: event
provider: Microsoft-Windows-Time-Service
}
host: { [-]
architecture: x86_64
hostname: wkst03
id: 0522759f-f26f-4ec1-8da4-636e3f1eb358
ip: [ [+]
]
mac: [ +*]
]
name: wkst03.magnuntempus.financial
os: { [-]
}
}
log: { [-]
level: warning
}
message: NtpClient was unable to set a domain peer to use as a time source because of failure in establishing a trust relationship between this computer and the 'magnuntempus.financial' domain in order to
securely synchronize time. NtpClient will try again in 15 minutes and double the reattempt interval thereafter. The error was: The trust relationship between this workstation and the primary domain failed.
(0x800706FD)
tags: [ [+]
]
winlog: { [-]
api: wineventlog
channel: System
computer_name: wkst03.magnuntempus.financial
event_data: { [-]
}
event_id: 130
opcode: Info
process: { [+]
}
provider_guid: {06EDCFEB-0FD0-4E53-ACCA-A6FB8BF818CB}
provider_name: Microsoft-Windows-Time-Service
record_id: 109737
user: { [-]
domain: NT AUTHORITY
identifier: S-1-5-19
name: LOCAL SERVICE
type: Well Known Group
}
}
```

These look similar to sysmon data we checked earlier with event IDs, event provider, message and other details.

Now that we know how our data looks like, lets check windows event logs related to scheduled tasks that we saw before.

query

```
index=wineventlogs event.code IN (4698,4699,4700,4701,4702)
```

Above query will

```
Search for windowseventlogs index
```

- uses IN operator to check if event IDs 4698,4699,4700,4701,4702 present in event.code field.
you can also do event.code=4698 event.code=4699 so on. Both will get you same results

No events. That's odd... isn't it?

There is another channel that monitors scheduled task actions called "MicrosoftWindowsTaskScheduler/Operational". Let's check if we have that

query

```
index=wineventlogs event.provider="task" | dedup event.provider | table event.provider
```

Above query will

Search for windowseventlogs index

- Check for a keyword task in event.provider feild.
- Deduplicates all the event.provider values (if we have more than one)
- Presets in a tabular form

The screenshot shows the Kibana interface with a search bar containing the query: `index=wineventlogs event.provider="*task*" | dedup event.provider | table event.provider`. Below the search bar, it says "1 event before 18/07/2022 12:17:34.000" and "No Event Sampling". The results table has columns for "event.provider". A red box highlights the first row, which contains the value "Microsoft-Windows-TaskScheduler". A red arrow points from the text "The event provider we observed according to artifacts and Microsoft documentation" to this row.

We see "Microsoft-Windows-TaskScheduler/Operational" in the logs, but checking logs we observed a message

To help optimize for performance, Task Scheduler has automatically disabled logging. To re-enable logging, please use Event Viewer.

The screenshot shows the Kibana interface with a search bar containing the query: `index=wineventlogs event.provider="*task*" | dedup event.provider | table event.provider`. Below the search bar, it says "1 event before 18/07/2022 12:17:34.000" and "No Event Sampling". The results table has columns for "Time" and "Event". The "Event" column shows a JSON object with a "message" field containing the text: "To help optimize for performance, Task Scheduler has automatically disabled logging. To re-enable logging, please use Event Viewer.". A red box highlights this message.

Which means, we cannot use windows event logs for this (at least for now). This is a very important step and we will circle back in ## Detection Coverage and Log Value section

Lets turn data into actionable queries and results

Based on our analysis, we determined that we cannot rely on windows event logs or "Microsoft-Windows-TaskScheduler/Operational" channel events. The best option here is sysmon data.

Ok great, but what should we search in sysmon data?

To answer that question we need to know the artifacts left behind when scheduled tasks are created. Without going into too many details, below we can see those artifacts. These you can get from past Incidents, DFIR analysts, Blog posts, MITRE references, research papers, etc.

1. Windows events 4698,4699,4700,4701,4702 we researched before
2. "Microsoft-Windows-TaskScheduler/Operational" channel logs
3. File creations in C:\Windows\System32\Tasks folder (sysmon event 11) with svchost.exe as creation process
4. Registry CreateKey (name of the Task), DeleteKey, SetValue (ID, Index, SD) (sysmon events 12,13,14) svchost.exe is the Image and TargetObject is the path
5. Image load events for taskschd.dll (sysmon event 7)
6. Commandline arguments

We will build our queries based on the available data and fields in it.

Referring to [sysmon documentation](#), we can see event.code 11 is related to file creation event

query

```
index=sysmon event.code=11
```

Above query will

Search for sysmon index

- Check for logs with event.code 11.

Process.executable and process.path seems to be the fields we can search to fit our 3rd point above. Let's construct a query

query

```
index=sysmon event.code=11 "process.executable"="C:\Windows\system32\svchost.exe"  
"file.path"="C:\Windows\System32\Tasks\*" | transaction file.path
```

```
| table file.path,host.name
```

Above query will

Search for sysmon index

- Check for logs with event.code 11.
 - filter events that has svchost.exe as process
 - filter events that are only created in C:\Windows\System32\Tasks* path
 - stack results on file.path
 - present them in tabular form

```

1 index=syomon event.code=11 "process.executable"="C:\Windows\system32\svchost.exe" "file.path"="C:\Windows\System32\Tasks\*\* | transaction file.path
2 | table file.path,host.name

```

75 events before 18/07/2022 10:06:40 (0.000) No Event Sampling ▾

Events Patterns Statistics (75) Visualization

10 Per Page ▾ Format Preview ▾

file.path ▾ host.name ▾

file.path	host.name
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1161	wkst14.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1164	wkst15.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1147	wkst09.magnuntempus.financial
C:\Windows\System32\Tasks\Microsoft\Office\Office Performance Monitor	rdp01.magnuntempus.financial
	wkst01.magnuntempus.financial
	wkst02.magnuntempus.financial
	wkst03.magnuntempus.financial
	wkst05.magnuntempus.financial
	wkst06.magnuntempus.financial
	wkst07.magnuntempus.financial
	wkst08.magnuntempus.financial
	wkst09.magnuntempus.financial
	wkst10.magnuntempus.financial
	wkst11.magnuntempus.financial
	wkst12.magnuntempus.financial
	wkst13.magnuntempus.financial
	wkst14.magnuntempus.financial
	wkst15.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1144	wkst18.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1129	wkst03.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1126	wkst01.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2370586174-1517003462-1142029260-1125	wkst01.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-2285072431-2485292186-3246622615-500	wkst01.magnuntempus.financial
C:\Windows\System32\Tasks\OneDrive Reporting Task-S-1-5-21-3452670134-2458530845-825625581-500	wkst13.magnuntempus.financial

75 events are returned for our query and are easy to sift through.

This is where the true fun of Threat Hunting begins.

We need to interpret the results to find that needle in the haystack.

In @UKDaniel_Card words "**you don't need high privs to search the scheduled task history log.. but... there's a shit ton of ms/windows stuff in here so its hard to spot stuff..._**"

For this, we need a good understanding of our environment's known-normal along with the normal behavior of the Operating System (OS) we are investigating. Also, one of the common themes we can take is

"commonly observed events across multiple endpoints could be common OR something really bad is happening"

don't remember the person said it but experienced it first hand

This is not to say there won't be anything malicious in them BUT it will help us significantly reduce our dataset and gives a starting point. Some of the questions that help us in getting to this stage are

- Did we get the results we are expecting?
- What software do we install/approved on the system?
- Are there any anomalies in the data we received?
- How to narrow our investigation net?.

Let's say, CCleaner application is not approved in our environment but found in our results, we can follow up with the user or next steps according to our policies.

After filtering such results our refined query is

```

index=syomon event.code=11 "process.executable"="C:\Windows\system32\svchost.exe" AND
"file.path"="C:\Windows\System32\Tasks\" AND "file.path"!="C:\Windows\System32\Tasks\Microsoft\" AND
"file.path"!="C:\Windows\System32\Tasks>CreateExplorerShellUnelevatedTask"
"file.path"!="C:\Windows\System32\Tasks\GoogleUpdateTaskUserS"
"file.path"!="C:\Windows\System32\Tasks\MicrosoftEdgeUpdateTaskMachineCore"
"file.path"!="C:\Windows\System32\Tasks\MicrosoftEdgeUpdateTaskMachineUA"
"file.path"!="C:\Windows\System32\Tasks\Mozilla" "file.path"!="C:\Windows\System32\Tasks\Mozilla"
"file.path"!="C:\Windows\System32\Tasks\OneDrive" "file.path"!="C:\Windows\System32\Tasks\npcapwatchdog"
"file.path"!="C:\Windows\System32\Tasks\AVG" "file.path"!="C:\Windows\System32\Tasks\CCleaner*" | transaction
file.path

```

| table file.path,host.name

Above query will

Search for sysmon index

- Check for logs with event.code 11.
 - filter events that has svchost.exe as process
 - using AND operator to sepcify we want additional checks for file.path
 - Filtering some of the events from file.path based on our "commonly observed" methodology
 - stacking entries with file.path
 - Present in tabular form

```
1 index=sysmon event.code=11 "process.executable"="C:\Windows\system32\svchost.exe" "file.path"="C:\Windows\System32\Tasks\*\*" AND "file.path"!="C:\Windows\System32\Tasks\Microsoft\*\*" "file.path"!="C:\Windows\System32\Tasks\CreateExplorersShellInElevatedTask" "file.path"!="C:\Windows\System32\Tasks\GoogleUpdateTaskUserSA" "file.path"!="C:\Windows\System32\Tasks\MicrosoftEdgeUpdateTaskMachineUA" "file.path"!="C:\Windows\System32\Tasks\OneDrive" "file.path"!="C:\Windows\System32\Tasks\Mozilla" "file.path"!="C:\Windows\System32\Tasks\Mozilla\*\*" "file.path"!="C:\Windows\System32\Tasks\OneDrive" "file.path"!="C:\Windows\System32\Tasks\npcapwatchdog" "file.path"!="C:\Windows\System32\Tasks\AVG" "file.path"!="C:\Windows\System32\Tasks\CCleaner" | transaction file.path
2 | table file.path,host.name
```

1 event before 18/07/2022 10:10:47.000 No Event Sampling ▾

Events (1) Patterns Statistics (1) Visualization

10 Per Page ▾ Format Preview ▾

file.path	hostname
C:\Windows\System32\Tasks\Daily MagnumTempus IT Cleanup	wkst01.magnumtempus.financial wkst02.magnumtempus.financial

We are left with one task running on two computers. Great... But we will check for additional artifacts mentioned above to see if we can get to the same conclusion.

Let's check Registry events with event IDs found in [sysmon documentation](#)

query

```
index=sysmon "event.code" IN (12,13,14) "process.executable"="C:\Windows\system32\svchost.exe"
"registry.path"="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\*"
| transaction registry.path
| table registry.path,host.name
```

Above query will

Search for sysmon index

- Check for logs with event.code 12, 13, 14.
 - filter events that has svchost.exe as process
 - filter events with registry path
 - stacking entries with registry.path
 - Present in tabular form

Index=sysmon "event.code" IN (12,13,14) "process.executable"="C:\Windows\system32\svchost.exe" "registry.path"="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree*"
 | transaction registry.path
 | table registry.path,host.name

439 events (before 18/07/2022 10:14:03.000) No Event Sampling ▾

Events (439) Patterns Statistics (439) Visualization

10 Per Page ▾ Format Preview ▾

registry.path ▾ host.name ▾

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\Windows\UpdateOrchestrator\SD

dc.magnutempus.financial
 dc02.magnutempus.financial
 files.magnutempus.financial
 rdp01.magnutempus.financial
 wkst01.magnutempus.financial
 wkst02.magnutempus.financial
 wkst03
 wkst03.magnutempus.financial
 wkst04-1.magnutempus.financial
 wkst04.magnutempus.financial
 wkst05.magnutempus.financial
 wkst06.magnutempus.financial
 wkst07.magnutempus.financial
 wkst08.magnutempus.financial
 wkst09.magnutempus.financial
 wkst10.magnutempus.financial
 wkst11.magnutempus.financial
 wkst12.magnutempus.financial
 wkst13.magnutempus.financial
 wkst14.magnutempus.financial
 wkst15.magnutempus.financial

dc.magnutempus.financial
 dc02.magnutempus.financial
 files.magnutempus.financial
 rdp01.magnutempus.financial
 wkst01.magnutempus.financial
 wkst02.magnutempus.financial
 wkst03
 wkst03.magnutempus.financial
 wkst04-1.magnutempus.financial
 wkst04.magnutempus.financial
 wkst05.magnutempus.financial
 wkst06.magnutempus.financial
 wkst07.magnutempus.financial
 wkst08.magnutempus.financial
 wkst09.magnutempus.financial
 wkst10.magnutempus.financial

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\Windows\UpdateOrchestrator

439 events. A little tough to sift through but still manageable. If you check the screenshot, we see path ending with SD.

By now, you understood that, majority of Threat Hunting is research.

So, let's take a detour to see what these values are without going to much into the weeds. Scheduled Task registry "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree" contains key/value pairs (keys and subkeys) and we create a task "Daily Task" in the below example, 4 Registry keys are created under "Daily Task" key, (Default), ID, Index, SD (you can check this by creating a registry key and using RegEdit navigate to the path mentioned above).

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily Task

Name	Type	Data
(Default)	REG_SZ	(value not set)
ID	REG_SZ	(CA0208EF-B372-4369-A61A-F31DC61789DA)
Index	REG_DWORD	0x00000003 (3)
SD	REG_BINARY	01 00 04 88 00 00 98 00 00 00 00 00 00 14 00 00 02 00 74 00 04 00 00 00 10 18 00 9f 01 1f 00 01 02 00 00 00 00 05 20 00

Sub Keys

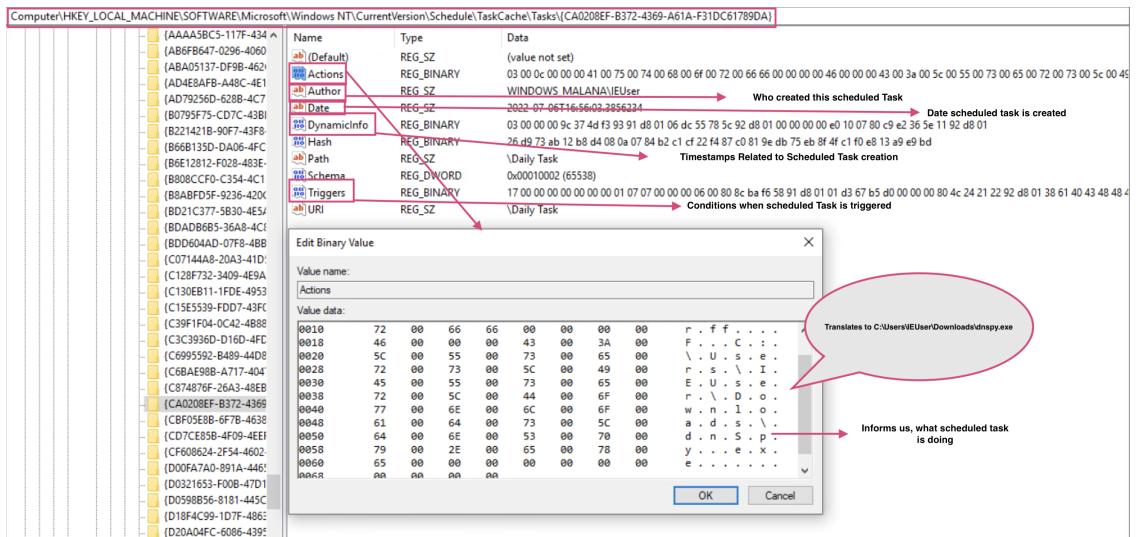
Task Schedule GUID

Data associated with Sub Keys

Name of the Registry Task

Detailed description: This screenshot shows a Windows Registry tree for the 'Daily Task' key under 'HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree'. The 'Daily Task' key has four subkeys: '(Default)', 'ID', 'Index', and 'SD'. The 'ID' subkey is highlighted with a red box and an arrow points to its value '(CA0208EF-B372-4369-A61A-F31DC61789DA)', which is labeled 'Task Schedule GUID'. The 'Index' subkey is also highlighted with a red box and an arrow points to its value '0x00000003 (3)', which is labeled 'Data associated with Sub Keys'. The 'SD' subkey is highlighted with a red box and an arrow points to its value '01 00 04 88 00 00 98 00 00 00 00 00 00 14 00 00 02 00 74 00 04 00 00 00 10 18 00 9f 01 1f 00 01 02 00 00 00 00 05 20 00', which is labeled 'Name of the Registry Task'.

Among these, ID field is important as it gives us a waypoint to navigate to it HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Task{CA0208EF-B372-4369-A61A-F31DC61789DA} (The last part if the GUID found in ID field). This will answer the question of "What's the scheduled Task is doing". Don't we see familiar names from the [Task Schedule Objects](#).



Ok from this, we can deduce that to check the name of the scheduled task, we can filter out (Default), ID, Index and SD.

query

```
index=sysmon "event.code" IN (12,13,14) "process.executable"="C:\Windows\system32\svchost.exe"
"registry.path"="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\|SD"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\|Index"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\|Id"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\|SD"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\|Index"
transaction registry.path | table registry.path,host.name
```

Above query will

Search for sysmon index

- Check for logs with event.code 12,13,14.
 - filter events that has svchost.exe as process
 - filtering some of the events from registry.path with sd, id, index subkeys
 - stacking entries with registry.path
 - Present in tabular form

Great, we have 83 events, better than 439 right?

By following exact steps (common methodology & known-normal of the environment) from above, we reduced our results with

```
index=sysmon "event.code" IN (12,13,14) "process.executable"="C:\Windows\system32\svchost.exe"
"registry.path"="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree" AND
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\OneDrive"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Mozilla"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\AVG"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\CCleaner"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\GoogleUpdate"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Schedule\TaskCache\Tree>CreateExplorerShell\UnelevatedTask"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\MicrosoftEdgeUpdateTaskMachine"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\npcapwatchdog"
"registry.path"!="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Amazon Ec2
Launch - Instance Initialization" "registry.path"!="HKLM\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft"
| transaction registry.path
| table registry.path,host.name
```

Above query will

- Search for sysmon index
- Check for logs with event.code 12,13,14.
 - filter events that has svchost.exe as process
 - using AND operator to sepcify we want additional checks for registry.path
 - Filtering some of the events from registry.path based on our "commonly observed" methodology
 - stacking entries with registry.path
 - Present in tabular form

```
1 index=symson "event_code" IN (12,13,14) "process.executable"]=="C:\Windows\system32\svchost.exe" "registry.path"]=="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\*" AND "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\OneDrive\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Mozilla\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\CLClearer\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\GoogleUpdate\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\CreateExplorerShellInElevatedTask\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\MicrosoftEdgeUpdateTaskMachine\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\ncpwatchdog\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\AmazonEc2Launch_Initialization\*" "registry.path"!="HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\*"
2 | transaction registry.path
3 | table registry.path, host.name

✓ 4 events [before 18/07/2022 11:19:40.000] No Event Sampling ▾ Job ▾ ||| Verbose Mode ▾
Events (4) Patterns Statistics (4) Visualization
10 Per Page ▾ Format Preview ▾
host.name ▾
registry.path ▾
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\SD
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\id
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\Index
wks01.magnumtempus.financial
wks02.magnumtempus.financial
wks01.magnumtempus.financial
wks02.magnumtempus.financial
wks01.magnumtempus.financial
wks02.magnumtempus.financial
wks01.magnumtempus.financial
wks02.magnumtempus.financial
wks01.magnumtempus.financial
wks02.magnumtempus.financial
Same Scheduled Task name AND same workstations
```

We got the same Task name on the same hosts.

By following above steps, we can say that a scheduled task named "Daily MagnumTempus IT Cleanup" is running on hosts wkst01.magnumtempus.financial and wkst02.magnumtempus.financial.

What is it doing though

We now know the name of the scheduled task and the workstations, let's search for that name to see what other events we can see for this.

query

host.hostname IN (wkst02, wkst01) AND "Daily MagnumTempus IT Cleanup"

Above query will

Search for host.hostname for two workstations wkst01 and wkst01 we observed before

- search for only events with keyword "Daily MagnumTempus IT Cleanup"

New Search

host.hostname IN (wkst02, wkst01) AND "Daily MagnumTempus IT Cleanup"

✓ 10 events before 18/07/2022 11:26:05:000 No Event Sampling

Events (10) Patterns Statistics Visualization

Format Timeline • Zoom Out + Zoom to Selection X Deselect

All time ▾ Job ▾ Verbose Mode ▾

1 hour per column

List ▾ Format 20 Per Page ▾

< Hide Fields □ All Fields

SELECTED FIELDS

- agent.hostname 2
- host 1
- source 1
- sysmon 1
- #winlog.event_id 3

INTERESTING FIELDS

- #@timestamp 3
- #@version 1
- #agent.event_id 2
- #agent.name 2
- #agent.type 1
- #sysmon 1
- #date_hour 2
- #date_index 1
- #date_minute 2
- #date_month 1
- #date_second 2
- #date_time 1
- #date_year 1
- #date_zone 1
- event.category 1
- event.event_id 3
- event.category 3
- event.event_id 2
- event.event_id 1
- event.event_id 1
- event.event_id 2
- file.directory 1
- file.filename 1
- file.path 1
- host.architecture 1
- hostname 2
- host.id 2
- host.ip 2
- host.name 3
- host.name 2
- #hosts.build 1
- hosts.family 1
- hosts.id 1
- hosts.name 1
- hosts.platform 1
- hosts.type 1
- #hosts.version 1
- #index 1
- alarm.level 1
- #message 10
- #process.event_id 2
- #process.executable 1
- #process.name 1
- #process.pid 2
- #process.parent 2
- #process.type 1
- #registry.data.string 1
- #registry.data.type 1
- #registry.hive 1

Event details (sysmon)

```

    > 12/02/2022 21:20:16.000 { [-]
      @timestamp: 2022-02-12T21:20:13.558Z
      @version: 1
      @source: sysmon
      @sysmon: 1
      @winlog.event_id: 3
      agent: { [-]
        ephemeral_id: 0b3d1ef6-3607-4855-8c26-5f442d35d52d
        hostname: wkst02
        id: d4ab8bf4-b9dd-99aa716c3e3
        name: wkst02
        type: winlogbeat
        version: 7.16.2
      }
      ecs: { [-]
        version: 1.12.0
      }
      event: { [-]
        action: Registry value set (rule: RegistryEvent)
        category: [ +]
        code: 3
        created: 2022-02-12T21:20:15.000Z
        kind: event
        module: sysmon
        provider: Microsoft-Windows-Sysmon
        type: [ -]
        change
      }
      host: { [-]
        architecture: x86_64
        hostname: wkst02
        id: d4ab8bf4-b9dd-99aa716c3e3
        ip: [ +]
        mac: [ -]
        os: { [-]
          build: 14393.4778
          kernel: 10.0.14393.4778 (rel.1_release_21H1101-1440)
          name: Windows Server 2016 Datacenter
          platform: windows
          type: windows
          version: 10.0
        }
        log: { [-]
          level: information
        }
        message: Registry value set.
        RuleName: Technique_id=T1053,Technique_name=Scheduled Task
        EventType: SetValue
        UserSid: S-1-5-21-20-13-558
        ProcessId: {D4A1B8F4-00B7-6208-1500-000000001002}
        ProcessId: 372
        ProcessName: C:\Windows\system32\svchost.exe
        Provider: {D4A1B8F4-00B7-6208-1500-000000001002}
        ProviderId: 372
        RuleName: Technique_id=T1053,Technique_name=Scheduled Task
        TaskCachePath: HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\Index
        TaskName: Daily MagnumTempus IT Cleanup
        TaskType: Index
      }
      rule: { [-]
        name: technique_id=T1053,technique_name=Scheduled Task
      }
      tags: [ -]
      sysmon
      beats_input_codec_plain_applied
    }
  
```

Full message for the event

```

    host: { [-]
      architecture: x86_64
      hostname: wkst02
      id: d4ab8bf4-b9dd-99aa716c3e3
      ip: [ +]
      mac: [ -]
      os: { [-]
        build: 14393.4778
        kernel: 10.0.14393.4778 (rel.1_release_21H1101-1440)
        name: Windows Server 2016 Datacenter
        platform: windows
        type: windows
        version: 10.0
      }
      log: { [-]
        level: information
      }
      message: Registry value set.
      RuleName: Technique_id=T1053,Technique_name=Scheduled Task
      EventType: SetValue
      UserSid: S-1-5-21-20-13-558
      ProcessId: {D4A1B8F4-00B7-6208-1500-000000001002}
      ProcessId: 372
      ProcessName: C:\Windows\system32\svchost.exe
      Provider: {D4A1B8F4-00B7-6208-1500-000000001002}
      ProviderId: 372
      RuleName: Technique_id=T1053,Technique_name=Scheduled Task
      TaskCachePath: HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\Index
      TaskName: Daily MagnumTempus IT Cleanup
      TaskType: Index
    }
    rule: { [-]
      name: technique_id=T1053,technique_name=Scheduled Task
    }
    tags: [ -]
    sysmon
    beats_input_codec_plain_applied
  }
  
```

One of the workstations

```

    host: { [-]
      architecture: x86_64
      hostname: wkst02
      id: d4ab8bf4-b9dd-99aa716c3e3
      ip: [ +]
      mac: [ -]
      os: { [-]
        build: 14393.4778
        kernel: 10.0.14393.4778 (rel.1_release_21H1101-1440)
        name: Windows Server 2016 Datacenter
        platform: windows
        type: windows
        version: 10.0
      }
      log: { [-]
        level: information
      }
      message: Registry value set.
      RuleName: Technique_id=T1053,Technique_name=Scheduled Task
      EventType: SetValue
      UserSid: S-1-5-21-20-13-558
      ProcessId: {D4A1B8F4-00B7-6208-1500-000000001002}
      ProcessId: 372
      ProcessName: C:\Windows\system32\svchost.exe
      Provider: {D4A1B8F4-00B7-6208-1500-000000001002}
      ProviderId: 372
      RuleName: Technique_id=T1053,Technique_name=Scheduled Task
      TaskCachePath: HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Daily MagnumTempus IT Cleanup\Index
      TaskName: Daily MagnumTempus IT Cleanup
      TaskType: Index
    }
    rule: { [-]
      name: technique_id=T1053,technique_name=Scheduled Task
    }
    tags: [ -]
    sysmon
    beats_input_codec_plain_applied
  }
  
```

We have 10 events BUT all of them are the ones we observed before AND no instances where we can see what it is doing.

Hmmm, strange.

We still have two more data components to check, Commandline arguments and process creation events.

There are several ways to create a scheduled task (GUI/Command tasks/Powershell/WMI). Though we will not be able to demonstrate all the ways, you can check [Scheduled Task creation by Schtasks](#), [Scheduled Task using At](#), [Scheduled Task using Powershell](#) and [Scheduled Task using WMI](#)

We will take some those events and search for keywords related to them. As we are checking for commadline arguments and process creation, we will check in the fileds process.command_line , process.executable and process.process.command_line

query:

```
((process.command_line="/c start /B C" OR "process.command_line"= "/create" OR "process.command_line"="/sc|onevent" OR "process.command_line"="New-ScheduledTaskAction -TaskName -CimSession" OR "process.command_line"="New-ScheduledTaskAction -Execute" OR "process.command_line"="New-ScheduledTaskTrigger" OR "process.command_line""register-scheduledTask") OR ("process.parent.command_line"=="C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" " AND "process.executable"="C:\Windows\System32\cmd.exe" AND "process.command_line"="/c start /B C*") ) AND "process.parent.executable"!="C:\Program Files\Microsoft Office\root\Integration\Integrator.exe"
```

Above query will

Search process.command_line for keywords related to scheduled task creation

- filtering events with Integrator.exe in process.parent.executable (fun exercise, check what it is and what it's used for)

The screenshot shows a Splunk search interface with the following details:

- Event 1:** A process starts `C:\Windows\System32\cmd.exe` with commandline `/c start /B C`.
 - Process Information:** Agent: technique_id=1093, technique_name=CommandLine Interface
 - User:** nunes123
 - Parent Process:** `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
- Event 2:** The `cmd.exe` process starts `C:\Windows\system2\cmd.exe` with commandline `/c start /B C`.
 - Process Information:** Agent: technique_id=1093, technique_name=CommandLine Interface
 - User:** nunes123
 - Parent Process:** `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`
- Event 3:** The `cmd.exe` process starts `C:\Windows\system2\cmd.exe` with commandline `/c start /B C`.
 - Process Information:** Agent: technique_id=1093, technique_name=CommandLine Interface
 - User:** nunes123
 - Parent Process:** `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`

We see that an executable called cleanup.exe starting via cmd.exe from powershell.exe being parent process. It is running on one of the hosts that we deemed to be suspicious (wkst01) AND cleanup.exe is in tmp folder. **A very strange combination in our org.**

Based on the Time this is executing, commandline arguments, process combination, folder location, AND workstation, we can with some degree of confidence say that this is malicious IF NOT at least suspicious. We can either start our IR collection process to determine other actions OR perform more log analysis to confirm.

Let's go a little further and see what this executable is doing by searching for the name cleanup.exe.

query:

```
cleanup.exe
```

Time	Event
12/02/2022 21:15:59.017	<pre>{ "@timestamp": "2022-02-12T21:15:59.017Z", "@version": 1, "agent": { "id": 3516, "name": "wkst02", "type": "windows" }, "ecs": { "version": 1 }, "event": { "action": "File created", "category": "File System", "code": 0, "kind": "File System" }, "file": { "path": "C:\Windows\Temp\cleanup.exe" }, "host": { "name": "wkst02" }, "log": { "source": "Windows" }, "message": "File created" } RuleName: - UtcTime: 2022-02-12 21:15:59.017 ProcessGuid: {D4A1B8F4-02E6-6208-7200-000000001002} ProcessId: 3516 Image: C:\Windows\Explorer.EXE TargetFilename: C:\Windows\Temp\cleanup.exe CreationUtcTime: 2022-02-12 21:15:59.017 User: MAGNUMTEMPUS\karen.metuens process: {} tags: [] winlog: {} }</pre> <p>Show as raw text</p> <p>agent.hostname = wkst02 host = 127.0.0.1:8088 source = http:sysmon-hec sourcetype = _json winlog.event_id = 11</p>

113 events. Great..

Let's see what events we see by checking event.action like we did before

query

```
cleanup.exe
| dedup event.action
| table event.action
```

Search for keyword cleanup.exe

- deduplicate tasks
 - present us in tabular form

```
1 cleanup.exe
2 | dedup event.action
3 | table event.action
```

✓ 8 events before 18/07/2022 12:01:26.000 No Event Sampling ▾

Events (8) Patterns Statistics (8) Visualization

10 Per Page ▾ Format Preview ▾

event.action ▾

- File created (rule: FileCreate)
- Process terminated (rule: ProcessTerminate)
- Process accessed (rule: ProcessAccess)
- Network connection detected (rule: NetworkConnect)
- Image loaded (rule: ImageLoad)
- Registry object added or deleted (rule: RegistryEvent)
- Process Create (rule: ProcessCreate)
- Dns query (rule: DnsQuery)

Sysmon Tasks we see for cleanup.exe

There are some VERY interesting sysmon Tasks here.

Go on an adventure to see what they are doing.

Detection Engineering:

With the help of different gadgets, the inspector (we) have succeeded in unveiling a malicious scheduled task. While our DFIR folks have their little field trip, let's see how we can use the hunt to better our detections and visibility.

In detection engineering we should ask ourselves: detections relevance and security value, challenges in consuming/using the telemetry, and its impact on our visibility/coverage.

One of the sites we can check is [strontic](#), especially for command-line utilities native to windows(if it is present in the site AND is native to windows). This gives me all the options that A tool can perform, its default path, Hash, signature, metadata, and get this, POSSIBLE MISUSE. That is so cool. We can check sigma rules, Malware IOCs, Atomic Red Team tests, and Yara rules

Depending on the maturity of the organization, you can use the site to create queries from known paths of misuse/attacks OR you can use it for testing your own detections.

Detection Coverage and Log Value

As we checked before, we do not have all the events where we can get information on Scheduled Tasks and their associated actions.

With our hunt, we identified that Windows event Logs (Security - 4698) and Microsoft-Windows-TaskScheduler/Operational.evtx are also helpful in performing hunts and creating detections. **These must be explicitly collected and are not on by default.** This exercise can be viewed as a "visibility coverage gap" and helped us identify two more additional data sources for better detection coverage.

Below events would trigger if we collected the above telemetry

User "W^{...}" registered Task Scheduler task "\MyApp"

Event 129, TaskScheduler

General Details

Task Scheduler launch task "\MyApp", instance "C:\Users\IEUser\Downloads\" with process ID 9788.

Event 129, TaskScheduler	
General	Details
<input checked="" type="radio"/> Friendly View	<input type="radio"/> XML View
Opcode	0
Keywords	0x8000000000000000
- TimeCreated	
[SystemTime]	[REDACTED]
EventRecordID	2283
Correlation	
- Execution	
[ProcessID]	1580
[ThreadID]	7252
Channel	Microsoft-Windows-TaskScheduler/Operational
Computer	[REDACTED]
- Security	
[UserID]	S-1-5-18
- EventData	
TaskName	\MyApp
Path	C:\Users\IEUser\Downloads\[REDACTED]
ProcessID	9788
Priority	16384

Event 200, TaskScheduler

General Details

Task Scheduler launched action "C:\Users\IEUser\Downloads\MyApp.exe" in instance "{1fa319f2-ffc1-4e36-9314-096e92f29ae7}" of task "\MyApp".

Event 201, TaskScheduler

General Details

Task Scheduler successfully completed task "\MyApp", instance "{1fa319f2-ffc1-4e36-9314-096e92f29ae7}", action "C:\Users\IEUser\Downloads\MyApp.exe" with return code 2147516570.

Most of the time detection engineering is a reactive space (based on the attacks we observe) and covering all variations of existing tools/techniques OR creation of new ones is not possible. So, revisiting a rule after a specified amount of time is recommended. Some of the questions we want to answer are,

- is the rule/s still valid?
- Is there any new research in the space and will that be covered by our existing rules?
- Is it time to retire the rule? maybe the technique or tool is not valid and such.

Advanced Scheduled Task Techniques and some ideas

Hidden Scheduled Tasks

[Hafnium](#) hides tasks from the schtasks.exe executable and Task Scheduler. It does this by deleting the associated Security Descriptor registry value that is created automatically upon task creation.

Security Descriptor registry value is used for determining the users allowed to run the task. This is a binding between the tasks we see when we query VS the tasks that are actually present. If we remove this key and its value, you will not see the task though it is present. This can only be performed by the SYSTEM user.

pro-tip

So, now think about where this step can be performed in the kill chain or MITRE. Because if it needs a SYSTEM account to perform the action, you should be able to see one of the PRIVESC techniques before you can see that. In the contrast, if you see this, that means the perpetrator already performed PrivEsc.

query

```
index=symon AND event.code=12 AND message="DeleteKey" AND  
"registry.path"="HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree*\SD"
```

Changing Scheduled Tasks with COM Handles

This technique is detailed in [Misc-PowerShell-Stuff](#)

Psuedo-code

Change in Scheduled task followed by registry delete/create of default key of the COM at
HKCR:\{CLSID{\#\#HANDLER_found_in_scheduled_task}\}\InprocServer32

Some other use cases could be

1. Scheduled Tasks making unusual communications

2. Scheduled Tasks downloading powershell cradles
3. Scheduled Tasks executing binaries from odd folders
4. Scheudled Task changes (parameters and scripts/executable/events)
5. Monitoring Deleted Scheduled tasks by checking Event ID 141 (Microsoft-Windows-TaskScheduler/Operational log) and Event ID 4699 (Security)
6. Changes for schedules inside |Microsoft\Windows| by unusual executable.

Some Known-attacks

1. APT28, Fancy Bear, Sofacy or STRONTIUM (Deleting Scheduled Tasks that can be retrived from Registry forensics)[MITRE Groups](#)
2. [Scheduled Task invoking MSBUILD.exe](#)
3. [IcedID Malware](#) - scheduling a task at logon
4. [Exploiting Environment Variables](#) in Scheduled Tasks for UAC Bypass
5. [Trickbot](#)
6. [Fentanyl](#) using Scheudled tasks to execute VB script

THREAT HUNTER TEMPLATE

Playbook Title: Hunting for Adversary's Schedule

Mitre Tactic: T1053

Mitre Sub Technique: T1053.005

Hypothesis:

Adversaries might create/change scheduled tasks on local/remote endpoints using GUI/command line tools to execute additional/new executables/hta files/js files for maintaining persistence in the environment

Proposed Detection Query:

Query to leverage sysmon data to hunt for scheduled tasks

```
index=sysmon AND event.code=11 AND "process.executable"="C:\\Windows\\System32\\svchost.exe" AND
"file.path"="C:\\Windows\\System32\\Tasks\\*"    "file.path"!="C:\\Windows\\System32\\Tasks\\Microsoft\\*"
"file.path"!="C:\\Windows\\System32\\Tasks\\CreateExplorerShellUnelevatedTask"
"file.path"!="C:\\Windows\\System32\\Tasks\\GoogleUpdateTaskUserS"
"file.path"!="C:\\Windows\\System32\\Tasks\\MicrosoftEdgeUpdateTaskMachineCore"
"file.path"!="C:\\Windows\\System32\\Tasks\\MicrosoftEdgeUpdateTaskMachineUA"
"file.path"!="C:\\Windows\\System32\\Tasks\\Mozilla"   "file.path"!="C:\\Windows\\System32\\Tasks\\Mozilla\\*"
"file.path"!="C:\\Windows\\System32\\Tasks\\OneDrive*"   "file.path"!="C:\\Windows\\System32\\Tasks\\npcapwatchdog"
"file.path"!="C:\\Windows\\System32\\Tasks\\AVG*"  "file.path"!="C:\\Windows\\System32\\Tasks\\CCleaner*" | transaction file.path
| table file.path,host.name
> ((process.command_line)="*c start /B C*" OR "process.command_line"="* /create*" OR "process.command_line"="*\\sc\\ onevent*")
OR ("process.parent.command_line"="\"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe\" " AND
"process.executable"="C:\\Windows\\System32\\cmd.exe" AND "process.command_line"="*c start /B C*") AND
"process.parent.executable"!="C:\\Program Files\\Microsoft Office\\root\\Integration\\Integrator.exe"
```

Simulation Details: Does not apply to us

Hunter Limitations/Observation Notes:

When searching for Scheduled tasks, we observed that we do not have all the data required, windows event logs, and "Microsoft-Windows-TaskScheduler/Operational" channel events. Also, we only see only one workstation wkst01 has cleanup.exe but no events for wkst02

Hunt Findings:

We observed a scheduled Task named "Daily MagnumTempus IT Cleanup" on computers wkst02 and wkst01

References:

1. <https://www.acad-write.org/mentor/how-to-write-a-hypothesis/>
2. <https://youtu.be/vZgl00UcATw?t=26314>
3. <https://attack.mitre.org/techniques/T1053/>
4. <https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4698>
5. <https://strontic.github.io/xcyclopedia/>
6. <https://github.com/enigma0x3/Misc-PowerShell-Stuff/blob/master/Get-ScheduledTaskComHandler.ps1>

7. <https://www.microsoft.com/security/blog/2022/04/12/tarrask-malware-uses-scheduled-tasks-for-defense-evasion/>
8. <https://twitter.com/jhencinski/status/1338187598599233537>
9. <https://www.fortinet.com/blog/threat-research/icedid-malware-analysis-part-two>
10. <https://www.tiraniddo.dev/2017/05/exploiting-environment-variables-in.html>
11. <https://securityintelligence.com/posts/new-malware-trickbot-anchordns-backdoor-upgrades-anchormail/>
12. <https://www.capesandbox.com/analysis/282242/>
13. <https://attack.mitre.org/groups/G0007/>
14. <https://docs.microsoft.com/en-us/troubleshoot/windows-server/system-management-components/schedule-server-process>
15. <https://redcanary.com/threat-detection-report/https://blog.menasec.net/2019/03/threat-hunting-25-scheduled-tasks-for.html>
16. <https://posts.bluraven.io/hunting-for-the-behavior-scheduled-tasks-9efe0b8ade40>
17. https://twitter.com/search?q=windows%20scheduled%20tasks&src=recent_search_click
18. <https://t.co/d54Gdkju5c>
19. <https://openCyberSecurityAlliance.org/huntbook-persistent-threat-discovery-kestrel/>
20. @jaredcatkinson - A person to follow
21. <https://www.ired.team/offensive-security/lateral-movement/wmi-via-newscheduledtask>