Visualizing processes and their parent/child relationships

You are a security analyst, combing through your data looking for signs of malicious activity. You spot a process which looks interesting to you, and you want to trace the activity or relationships of that process. At first, you try to brute force this by copying/pasting parent and child process IDs over and over again, but this turns out to be a time-consuming and frustrating task. You're concerned that the next time you need to do this in the middle of an ongoing security incident, you may miss one item that could have led you to something interesting. Splunk does a great job of ingesting process data, allowing you to search and correlate, but it's challenging to visualize parent/child relationships for this data, especially spanning multiple generations.

Required data

- Microsoft Sysmon
- Windows event logs

How to use Splunk software for this use case

First, you'll need to ensure you have completed some prerequisites:

1. Configure the Splunk Add-on for Microsoft Sysmon and Splunk Add-on for Microsoft Windows, together with the Windows Universal Forwarder, to capture process data.
2. Configure the Windows endpoints to capture the process-related events. Two types of data you can use for capturing new process creation events are:
   - Sysmon with Event Code 1 enabled (SwiftOnSecurity or Olaf Hartong’s Sysmon configs are both good places to start). If you want to use the Process tracking using an add-on solution described later, this data type is best.
   - Windows Security Event Logs with Event ID 4688, including command line in process creation events.

Step 1 - Process tracking using search

In this example, let’s assume you have a known malicious spreadsheet (Salaries.xls) in your environment, and you want to understand if it’s been opened. If it has, you’d also like to know what has transpired.

1. Look to see if the spreadsheet filename, salaries.xls, has been observed in any EventCode 1 events in Sysmon.
2. Run the following search. You can optimize it by specifying an index and adjusting the time range.

```
source="xmlwineventlog:microsoft-windows-sysmon/operational" EventCode=1
| search CommandLine="*Salaries.xls*"
```
## Search explanation

The table provides an explanation of what each part of this search achieves. You can adjust this query based on the specifics of your environment.

<table>
<thead>
<tr>
<th>Splunk Search</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>source=&quot;xmlwineventlog:microsoft-windows-sysmon/operational&quot; EventCode=1</td>
<td>Search Sysmon data for process creation events. This part of the search uses a source (not sourcetype) for Sysmon data.</td>
</tr>
<tr>
<td></td>
<td>search CommandLine=&quot;<strong>Salaries.xls</strong>&quot;</td>
</tr>
<tr>
<td></td>
<td>table _time host user ProcessId CommandLine</td>
</tr>
</tbody>
</table>

### Results

Process creation events referencing Salaries.xls are returned from this search. The search will also let you know the command line that triggered the process - for example, showing that Excel.exe is opening the spreadsheet.

Using the previous search, you can use the table command again to display more fields such as the parent process name, the parent process ID, parent process path, process path, and more. These fields can help you explore further and confirm that processes are running from expected locations.

```
| table parent_process_name parent_process_id parent_process_path process_path |
```

Process path information is a very valuable piece of information when looking for processes launching from places they shouldn’t be (temp directories, startup folders, etc). You can use the process ID returned from this search to begin hunting down the entire process tree, using the resulting process ID as the parent process ID in each new search, although this can be time consuming.

### Step 2 - Process tracking using an add-on

With this step of the process, you can use the PSTree for Splunk app to pass in parent and child fields and then create a table of the resulting process family structure.
The PSTree for Splunk app is not supported. Customers are solely responsible for ensuring proper functionality and version compatibility of unsupported apps and add-ons.

1. Install the PSTree for Splunk app, along with the Splunk Python SDK.
2. Run the following search. You can optimize it by specifying an index and adjusting the time range.

```
source="xmlwineventlog:microsoft-windows-sysmon/operational" EventCode=1 user=[username]
  | rex field=ParentImage "x5c(?<ParentName>[^x5c]+)$"
  | rex field=Image "x5c(?<ProcessName>[^x5c]+)$"
  | eval parent = ParentName." (".ParentProcessId."")"
  | eval child = ProcessName." (".ProcessId."")"
  | eval detail=strftime(_time,"%Y-%m-%d %H:%M:%S")." ".CommandLine
  | pstree child=child parent=parent detail=detail spaces=50
  | search tree=*Salaries.xls*
  | table tree
```

**Search explanation**

The table provides an explanation of what each part of this search achieves. You can adjust this query based on the specifics of your environment.

<table>
<thead>
<tr>
<th>Splunk Search</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>source=&quot;xmlwineventlog:microsoft-windows-sysmon/operational&quot; EventCode=1 user=[username]</td>
<td>Search Sysmon data for process creation events from a specified user.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Splunk Search</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>search parameter after the pstree command has completed its operations.</td>
</tr>
<tr>
<td></td>
<td>Create a table from the tree data.</td>
</tr>
</tbody>
</table>

**Results**

This search returns a table listing process traces several layers deep, with timestamps and command-line data included to assist you in your investigation.

**Next steps**

The content in this guide comes from a previously published blog, one of the thousands of Splunk resources available to help users succeed. In addition, these Splunk resources might help you understand and implement this use case:

- Blog: [Rex groks gibberish](#)