Detecting Attacker Behavior With Splunk Based On Analytics

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.

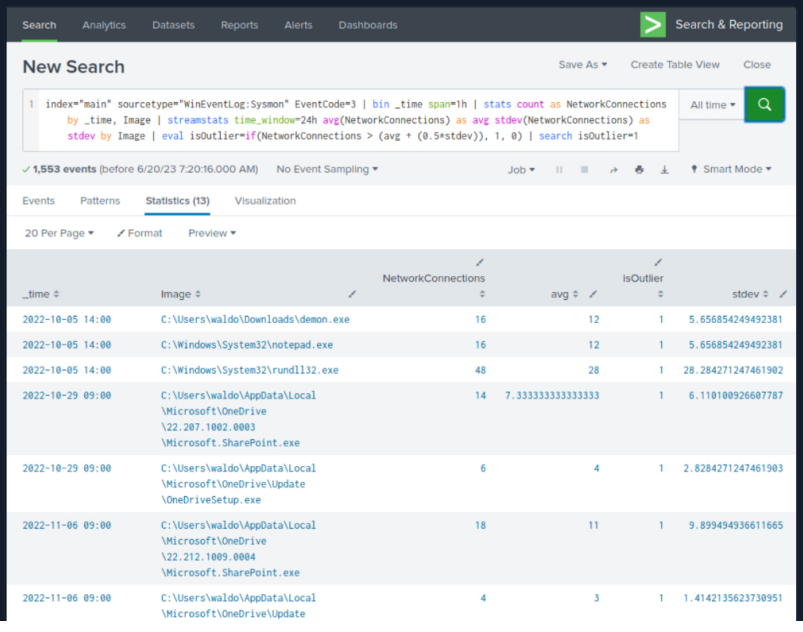
Detecting Attacker Behavior With Splunk Based On Analytics

```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=3 | bin _time span=1h | stats count as
```

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.



_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\Update\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\Update	4	3	1	1.4142135623730951

Go to Questions

Table of Contents

Splunk Fundamentals

Introduction To Splunk & SPL

Using Splunk Applications

Investigating With Splunk

Intrusion Detection With Splunk (Real-world Scenario)

Detecting Attacker Behavior With Splunk Based On TTPs

Detecting Attacker Behavior With Splunk Based On Analytics

Skills Assessment

Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

	\OneDriveSetup.exe				
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.

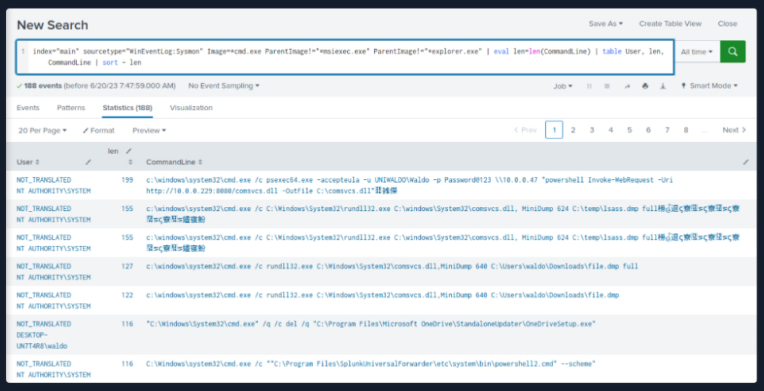
1. **Example: Detection Of Abnormally Long Commands**

   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)
```

   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" sourcetype="WinEventLog:Sysmon" Image=*cmd.exe ParentImage!="*msiexec.exe"
```

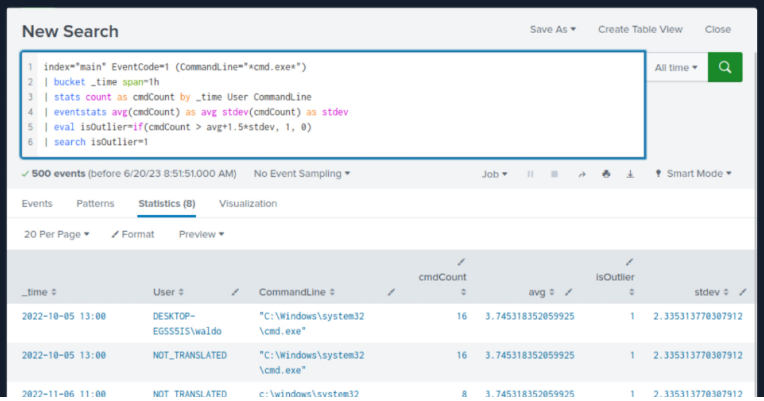


   Once again, we observe the recurrence of malicious activity that we previously identified during our investigation.

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.

```
index="main" EventCode=1 (CommandLine="*cmd.exe*") | bucket _time span=1h | stats count
```



		\cmd.exe /c dir				
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-EGS55IS\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**

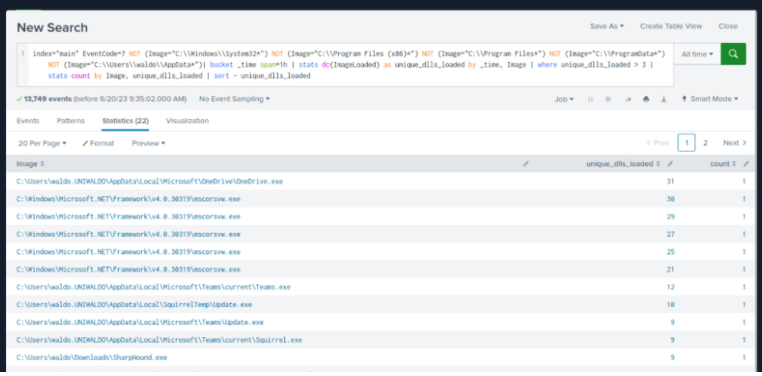
   It is not uncommon for malware to load multiple DLLs in rapid succession. The following SPL can assist in monitoring this behavior.

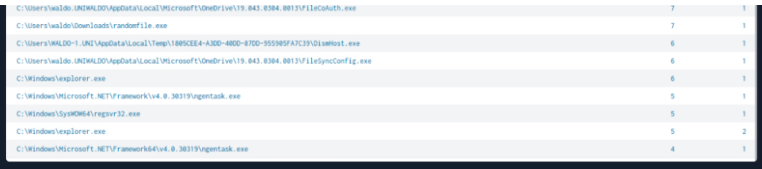
```
index="main" EventCode=7 | bucket _time span=1h | stats dc(ImageLoaded) as unique_dlls_l
```

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 Fi
```

   - `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`).





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

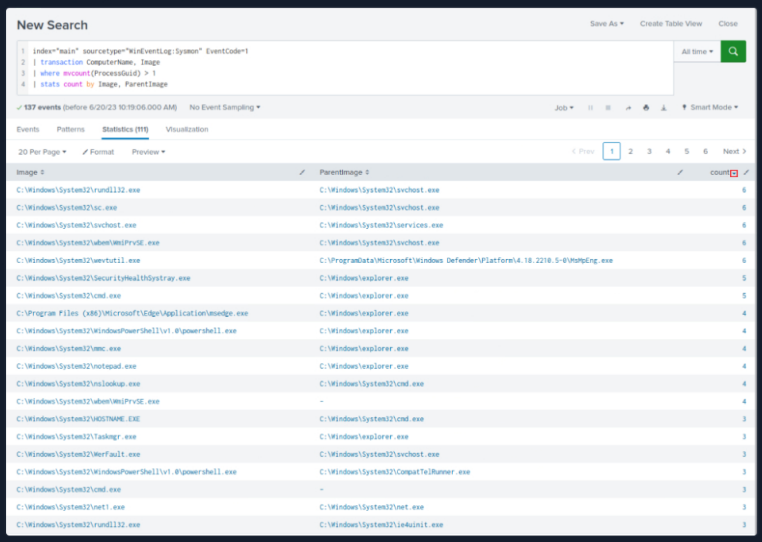
It's important to note that this behavior can also be exhibited by legitimate software in numerous cases, so context and additional investigation would be necessary to confirm malicious activity.

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.

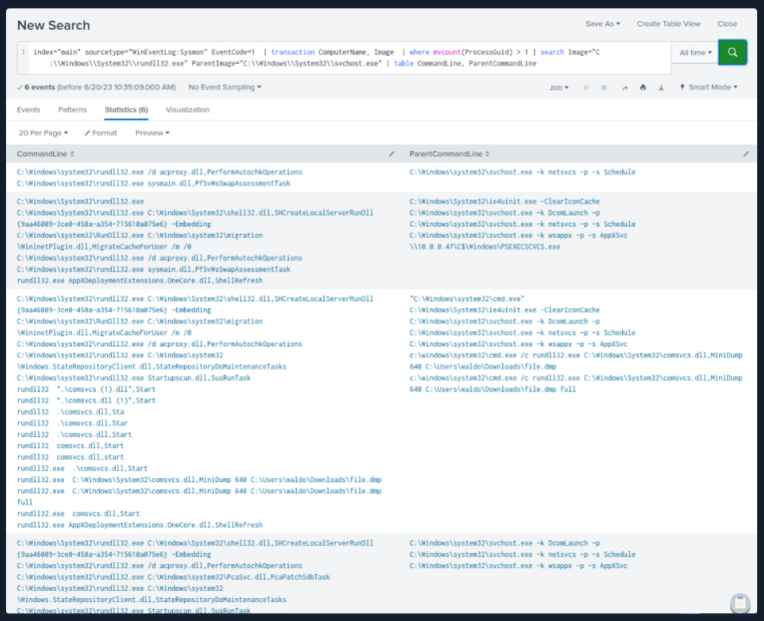
```
index="main" sourcetype="WinEventLog:Sysmon" EventCode=1 | transaction ComputerName, Ima
```

- `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`).



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, Im
```



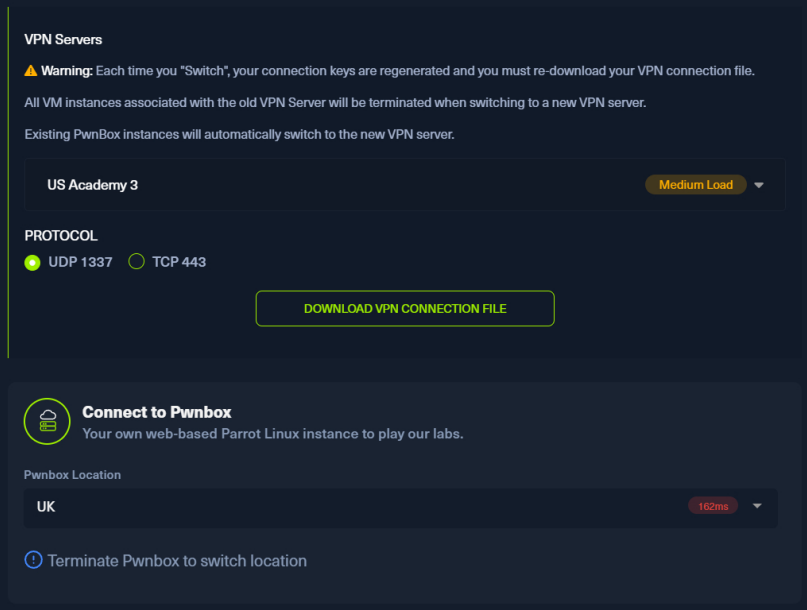
After careful scrutiny of the results, it becomes apparent that we not only identify the presence of previously identified suspicious commands but also new ones.

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.

Practical Exercises

Navigate to the bottom of this section and click on `Click here to spawn the target system!`

Now, navigate to `http://[Target IP]:8000`, open the `Search & Reporting` application, and answer the question below.



Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

+ 1 Navigate to http://[Target IP]:8000, open the "Search & Reporting" application, and find through an analytics-driven SPL search against all data the source process images that are creating an unusually high number of threads in other processes. Enter the outlier process name as your answer where the number of injected threads is greater than two standard deviations above the average. Answer format: _.exe

randomfile.exe

Submit

Previous Next

Mark Complete & Next