Splunk for Cybersecurity Issues (#6889)
Adversary Emulation Using Caldera (#6454)

Final Deliverable

Will VanderFeltz
SET Labs
Summer Quarter 2020

Table of Contents

Executive Summary

I have been given administrative privileges on Splunk. We would like to create a test bed that mimics our production environment as closely as possible, and have Splunk installed on each machine in the test bed forwarding data to a unique index that is separate from the production system. From there, we can run scheduled Caldera operations that we can analyze in Splunk. The School of Engineering and Technology is currently in year 2 of a 3 year agreement with Splunk. We pay for the right to ingest 2 GB of data per day. We need to get an inventory on how much data we are ingesting. At the moment, there is no way for us to view the daily/monthly license usage. We would like to know if there would be benefit to paying to collect more data, if we need to collect data, or if we This project will attempt to determine the utility of Splunk for indicators of compromise (IOCs) and evaluate the need for more information to be collected for reducing false positives/negatives.

Overview

Ideally, we would like to determine where the weaknesses are on a production system. Presumably, an adversary would take advantage of these weaknesses after a compromise. We could**Adversary Emulation Using Caldera (#6454)** manually run through every post-compromise technique in the Mitre ATT@CK framework (https://attack.mitre.org/), but this would take a long time and require a means of verifying a successful technique. It would require detailed and intimate knowledge of the framework.

We don't want to work on a production system, at least the first time we try something, in case it causes one or more parts of the system to fail. Instead, we create a duplicate or near-duplicate environment and use that as the testbed.

As mentioned, doing it manually would take a lot of effort. Caldera is an automated adversary emulation application that could tell us a lot about weaknesses in the test system, advising changes for the production system.

Consequently, this is the approach we will take: create a test system that mimics all or part of the production system, and then set up Caldera to run on the test system, and evaluate the results. From there, we can either make changes on the production system or highlight the ATT&CK framework for where we already handle all or part of a technique, or don't handle it at all, allowing prioritization of effort in reducing the attack surface. We can highlight techniques via the MITRE ATT&CK Navigator.

Splunk For Cybersecurity Issues (#6889)

Gathering, correlating and analyzing log files from various systems is a key aspect of determining if something anomalous has happened or if there are signs of successful infiltration by an unauthorized actor or exfiltration of data by an existing actor.

We don't have the resources to devote to spending a lot of time poring over log files to see if there are indications of compromise. In addition, we may already be compromised and the actor is sending data out or trying to contact other systems on the campus network, presumably looking for higher-value targets.
We don't even have the time to evaluate tools such as Splunk, which we have had on premise for the last two years of a 3-year agreement. It may be useful to us, but we don't know. It may be more useful if we paid more money for collecting more data from more systems.

Caldera: The Game Plan

What is it?

Measuring aspects of a network's security posture through penetration testing, red teams, and adversary emulation is resource-intensive. CALDERA offers an intelligent, automated red team system that can reduce resources needed by security teams for routine testing. We are going to use CALDERA to assess and look for vulnerabilities in our test bed, and if we find some; patch those vulnerabilities in our production environment. **We are using Caldera version 2.6.3**

Setup

A successful Caldera operation is made up of two parts: a server and an agent(s).The server is going to be the location in which you are managing the operation from. This is where the cloned Github repository will reside.

To clone the repository and set up your Caldera server, follow these steps:

For Windows

1.Install Git for Windows (Git Bash)
2.Install latest version of Python (to use pip)
3.Open Git and run the following command
```
Git clone --recursive https://github.com/mitre/caldera.git --branch 2.6.3
```
4.CD to the Caldera directory
5.Run the following command (when in the caldera directory)
```
Pip install -r requirements.txt
```
6.Start the Caldera server(when in the caldera directory)
```
Python server.py --fresh
```
7.Open Chrome(Must be Chrome) and navigate to Localhost:8888

For Linux

1.Open Terminal
2.Install latest version of Python (to use pip)
3.Run the following command
```
Git clone --recursive https://github.com/mitre/caldera.git --branch 2.6.3
```
4.CD to the Caldera directory
5.Run the following command (when in the caldera directory)
`Pip install -r requirements.txt` (Installs dependencies)
6.Start the Caldera server(when in the caldera directory)
```
Python server.py --fresh
```
7.Open Chrome(Must be Chrome) and navigate to Localhost:8888

How to Run an Operation

The second part of a successful Caldera operation is the agent(s). The default agent for Caldera is called 54ndc47 (Sandcat). 54ndc47 is a CAT (coordinated access trojan). Agents are processes which are deployed on compromised hosts and connect with the C2 server periodically for instructions. An agent connects to the server through a contact, which is a specific connection point (port 8888 is default) on the server.You can run one operation on multiple agents at the same time. Sandcats can be installed as a Powershell, CMD, Linux, or MacOS command. From the Caldera GUI dashboard on the server, navigate to the “Sandcat” option under the “Plugins” tab, select your operating system, and paste the command into the machine that you would like to run tests on. Ensure that you change the IP address in the command to match that of the Caldera server.

Splunk: The Game Plan

<u>What is it?</u>

Splunk software is used for searching, monitoring, and analyzing machine-generated data. It captures, indexes, and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards, and visualizations. Splunk is commonly used for application management, security and compliance, as well as business and web analytics. We will be using Splunk to search for indicators of compromise (IOCs) and evaluate the potential need for a change in terms of current usage license. We will install Splunk Universal Forwarders in a virtual test environment and monitor the machines while running automated tests against them to see how transparent the operations are in the logs. We will create a new index for these forwarders to send data to so that test data will not get mixed with the production environment.

<u>Setup</u>

Creating an Index

1. In Splunk Web, navigate to Settings > Indexes and click New.
2. To create a new index, enter:
--A name for the index. User-defined index names must consist of only numbers, lowercase letters, underscores, and hyphens. They cannot begin with an underscore or hyphen, or contain the word "kvstore".
--The index data type. For event data, click Events. This is the default data type.
--The path locations for index data storage:
-Home path. Leave blank for default `$SPLUNK_DB/<index_name>/db`
-Cold path. Leave blank for default `$SPLUNK_DB/<index_name>/colddb`
-Thawed path. Leave blank for default `$SPLUNK_DB/<index_name>/thaweddb`
--Enable/disable data integrity check.
--The maximum size of the entire index. Defaults to 500000MB.
--The maximum size of each index bucket. When setting the maximum size, use auto_high_volume for high volume indexes (such as the main index); otherwise, use auto.
--The frozen archive path. Set this field if you want to archive frozen buckets. For information on bucket archiving, see Archive indexed data.
--The app in which the index resides.
--The tsidx retention policy. See Reduce tsidx usage.
3. For more information on index settings, see Configure index storage.
--Click Save.

Installing a Splunk Universal Forwarder

Proper procedure for installing Windows Universal Forwarder
-Check license agreement
-Click "Customize Options"
-Click next (default path is fine)
-Click next (no SSL cert)
-Click next (local system)
-Select the event logs you would like to monitor
-Click next
-Create an admin user ID and credential (I used the same as my Splunk account)
-Click next
-Click next (deployment server)
-On the receiving indexer page, use splunk.d.insttech.washington.edu for the hostname, and port 18083 for the port
-Click next
-Click install

Give Splunk about 5 minutes to index everything, and then perform the following search in the Splunk web console:

```
Index=* host ="<Hostname>"
```

The events you selected during the install should show up in the search result. If not, change the index to "`_internal`" and look for errors in the splunkd log.

Forwarding Host Data to a Specific Index

By default, Splunk Forwarders forward data to the index="Main"

How To Forward Host Data to a Specific Index:
1.On the host machine that you are forwarding data from, navigate to:
`C:\Program Files\SplunkUniversalForwarder\etc\system\local`
2.Open the "inputs" file
3.Add a new line after "`host = <hostname>`" that reads:
`Index = <indexname>`
4.Save the file and navigate to the following directory in the CMD
C:\Program Files\SplunkUniversalForwarder\bin\
5.Run the command:
`Splunk restart`

This will restart the Splunk universal forwarder and apply the changes. From here on out, the data from that host should forward to the index that you specified in the inputs.conf file. If you see that it is still not forwarding, you can try going into:
`C:\Program Files\SplunkUniversalForwarder\etc\`
and deleting the "instance.cfg" file and then restart Splunk again.

Win:Event 4688

Win:Event 4688 -Process Creation
4688 - A new process has been created
-logs each program that is executed
-Who the program ran as
-Process that started this process
-Contains process launch path
-Logs go to Windows Security Log
-Logs any command line run in a Windows environment
-NEEDS to be enabled in all environments

How to enable Win:Event 4688
-Configured via Group Policy
-Computer Config > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > Detailed Tracking
-Select: Audit Process Creation, Select: Success (+Failure but more data collected) Select: OK

How to enable Command Line Process Auditing
-Computer Config>Policies>Administrative Templates>System>Audit Process Creation
-Select: Include command line in process creation events, Select: Enabled, Select: OK

Win:Event 4688-Interesting Behavior
-Processes not launching from where they should
-Svchost.exe should launch from `%SystemRoot%\System32\` NOT `C:\users\AppData\Roaming\Temp`
-SANS has a great DFIR Poster that visualizes this

Suspect Command Lines for Win:Event 4688
-Looking at Process Command Line Field (Instructions on the next page)
-Whoami (most real users KNOW their level of privilege in their environment)
-Netstat
-`C:\Windows\System32\svchost.exe -k netsvcs -p -s Schedule`
-Netsh advfirewall set allprofiles state off/runas int/SpecP[s
-Large chunks of obfuscated code (No legitimate user will be running scripted psh commands)

Limiting Splunk Noise on Win:Event 4688

I was having an issue where Splunk sub-processes took up a LOT of the Win:Event 4688 space in the logs and made it hard to find anything of value.

Solution: Create a Blacklist that uses a RegEx expression and grabs the Splunk Winprintmon.exe, regmon.exe (that were flooding the 4688 log), and does not forward them into Splunk. This removes almost all Splunk traffic from the log and allows for the Caldera operations and other vital information to come through cleanly.

Instructions:
--Navigate to: (cmd or GUI)
notepad C:\Program Files\SplunkUniversalForwarder\etc\apps\SplunkUniversalForwarder\local\inputs.conf
--Add the following under the last line in the stanza: `[WinEventLog://Security]`

```
blacklist1 = EventCode="4688" Message="(A new process has been created)(?s).*(splunk-winprintmon)"
blacklist2 = EventCode="4688" Message="(A new process has been created)(?s).*(splunk-regmon)"
```

--Save the inputs file after you add the blacklist events.
--You must restart Splunk to apply configuration changes

To restart Splunk:
--Navigate to the following directory in the CMD

```
C:\Program Files\SplunkUniversalForwarder\bin\
```

--Run the command:

```
Splunk restart
```

Initially, I had the issue of applying these changes in the wrong inputs.conf file. I was placing them inside of C:\Program Files\SplunkUniversalForwarder\etc\system\local\inputs.conf. This is the file I used to change the index configuration. This is incorrect.

The proper inputs.conf which is located under (As mentioned above:)

```
C:\ProgramFiles\SplunkUniversalForwarder\\etc\apps\SplunkUniversalForwarder\local\inputs.conf
```

You are able to create up to 9 blacklist items (however, you can combine filters if you need to.

Win:Event 4698

4698 - Scheduled Task Was Created
-Logs new scheduled tasks that are created
-Subject: Account Name = Who created
-Subject: Account Domain = Domain or Computer (if logged on locally)
-Subject: Logon ID = Can be correlated to 4624 for session info etc…
-Enabling 4698 also provides Event IDs: 4671, 4691, 5148-49, 4698-4702,5888-90
-Logs go to Windows Security Log

Scheduled Tasks may be one of the most commonly utilized persistence mechanisms

How to enable Win:Event 4698
-Configured via Group Policy
-Computer Config > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > Object Access
-Select: Audit Other Object Access Events, Select: Success (+Failure but more data collected)
Select: OK

-Big focuses are "Scheduled task created" and "Scheduled Task Updated"
-Gives account that created/updated the task
-One of the most utilized persistence methods because nobody usually checks these!
(Scheduled tasks for persistence)

Win:Event 4104

4104 - PowerShell Script Block Logging
-Script block auditing captures the full command or contents of the script
-Who executed the script
-When the script occured
-Regardless of how the Psh was executed it ends up here
-Mostly de-obfuscates the Script block
-Logs go to Application and Service Logs >Microsoft > Windows > PowerShell > Operational

How to enable Win:Event 4104
-Configured via Group Policy
-Computer Configuration > Policies >Administrative Templates > Windows Components > Windows PowerShell
-Select: Turn on Powershell Script Block Logging, and Select: Enabled, Select: Log script block invocation start/stop events

Win:Event 4104 - Interesting Behavior
Arguments with:
-obfuscation
-NoProfile (-nop)
-EncodeCommand, (-enc)
-WindowStyleHidden (-w hidden, -w 1)
-System.net.WebClient

Highly Recommended

Forwarding Operational PowerShell Logs to Splunk

Script Block Logging Into Splunk

A script block can be thought of as a collection of code that accomplishes a task. Script blocks can be as simple as a function or as full-featured as a script calling multiple cmdlets. Script block auditing captures the full command or contents of the script, who executed it, and when it occurred. Audits are recorded as event log entries in the Microsoft-Windows-PowerShell/Operational log regardless of how PowerShell was executed – from a command shell, the integrated scripting environment (ISE), or via custom hosting of PowerShell components. Event ID 4104 records the script block contents, but only the first time it is executed in an attempt to reduce log volume.

I detailed above how to enable Event 4104, but now we need to ensure that Splunk is monitoring this event. Event 4104 is not part of the Security logs, so we need to forward the Windows PowerShell Operational logs to Splunk.

To do this:
--Navigate and open the inputs config file located at

```
C:\ProgramFiles\SplunkUniversalForwarder\\etc\apps\SplunkUniversalForwarder\local\inputs.conf
```

--Press enter after the end of the last stanza in the file (ensuring a blank line between stanzas)
--Paste the following into the inputs.conf

```
[WinEventLog://Microsoft-Windows-Powershell/Operational]
checkpointInterval = 5
current_only = 0
disabled = 0
start_from = oldest
blacklist2 = EventCode="4105"
blacklist3 = EventCode="4106"
blacklist4 = EventCode="40961"
blacklist5 = EventCode="40962"
blacklist6 = EventCode="53504"
```

This enables Windows PowerShell Operational logs to be forwarded to Splunk. It also blacklists the events that take up a lot of space that we do not necessarily need to monitor. Event 4104 is robust enough that we can get the data that we need from it alone.

Splunk/Caldera: Bringing Everything Together

Now, I will demonstrate Splunk's ability to recognize Indicators of Compromise (IOCs) by running a Caldera operation and then analyzing the logs using Splunk.

Running a Caldera Operation (Windows Target)

(If you do not have Caldera installed, return to the Table of Contents and go to Caldera:Setup)

Step 1: Start Caldera Server

--Navigate to the Caldera directory in the cmd:

```
Cd caldera
```

--Start the Caldera server

```
Python server.py --fresh
```

Step 2: Navigate to localhost:8888 in Google Chrome

--This is how you access the Caldera GUI

Step 3: Go to the "Plugins" menu and click on "sandcat"

--Select which operating system your target machine is running. This is for the machine that you are going to run operations on.

--Copy the command

Step 4: Log onto the target machine

--Change the IP address from the default 0.0.0.0 to match the IP address of the Caldera Server

--(If you do not know the IP address of the server, go to it and use ipconfig/ifconfig to check.

--paste the sandcat command into the cmd/powershell/terminal (depending on OS) and execute

--It can be helpful to paste the sandcat command into a Google Doc so that it can be accessed easily

--Caldera is not designed to be a stealthy install, so it might trigger your antivirus.

--If you get errors trying to execute the command (this can happen a lot in pwsh), temporarily turn off Windows Defender and then turn it right back on after the command is executed

Step 5: Check the Caldera Server cmd/terminal

--You should see "Incoming beacon coming from ____"

--Your target machine is successfully communicating with your server

--This means that your agent is properly configured.You are ready to begin your operation.

Step 6: Time to Run an Operation!

--On your Caldera GUI, navigate to the "Campaigns" menu

--Click on the "Operations" Tab

--Click the "View" switch to change to "Add" an operation.

Step 7: Title your Operation

Step 8: Click Basic Options

--Select "Red" in the first drop down option

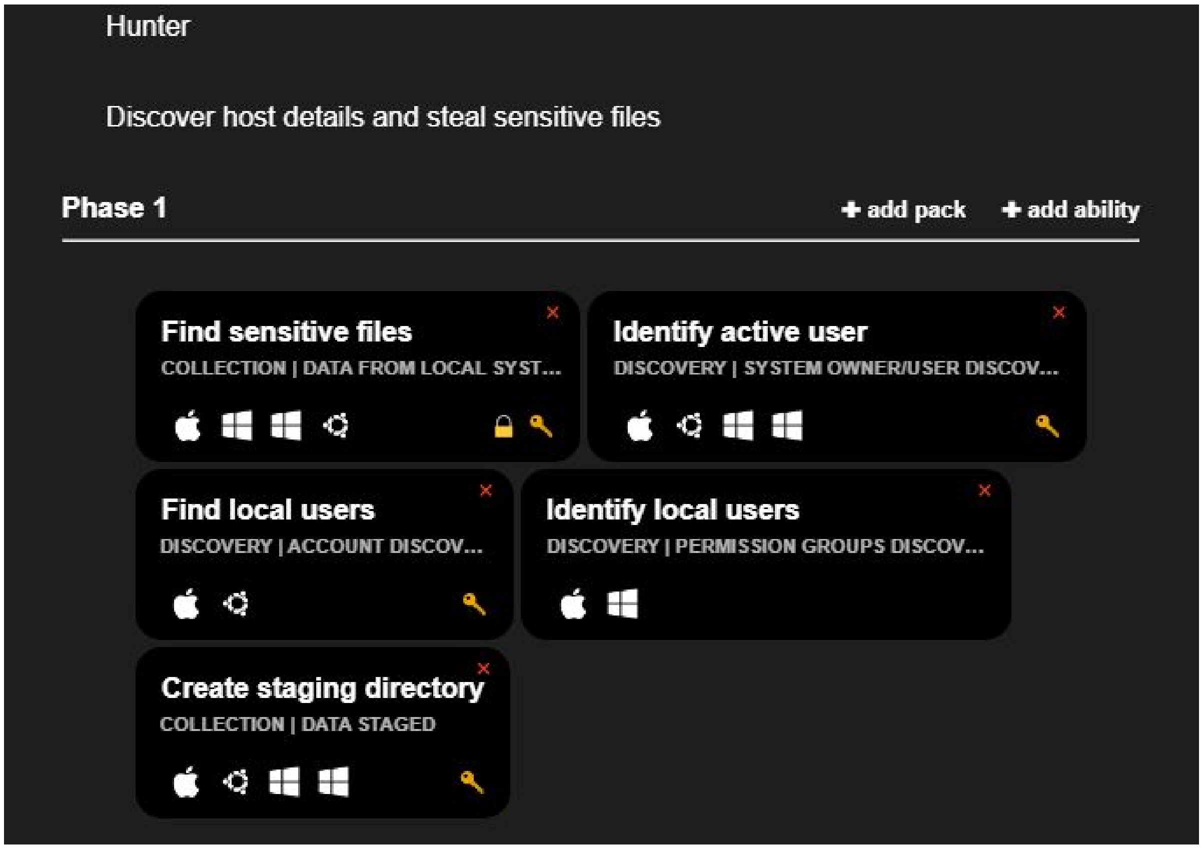
--Dropdown number 2 is where you select which Adversary you are going to use

--I will use "Hunter."

--For the 3rd drop down, select “Auto close operation”
--Leave the fourth dropdown on “Run immediately”
Step 9: Click “Start”
--Caldera will now execute the operation on the target machine
--Caldera runs in “Phases” by default
Step 10: Review your results!
--Once the operation is complete, you can review what “facts” Caldera has collected by clicking on the stars underneath the technique.

Hunter Adversary

--To see what tactics are in the “Hunter” adversary, see below


Hunter

Discover host details and steal sensitive files

Phase 1

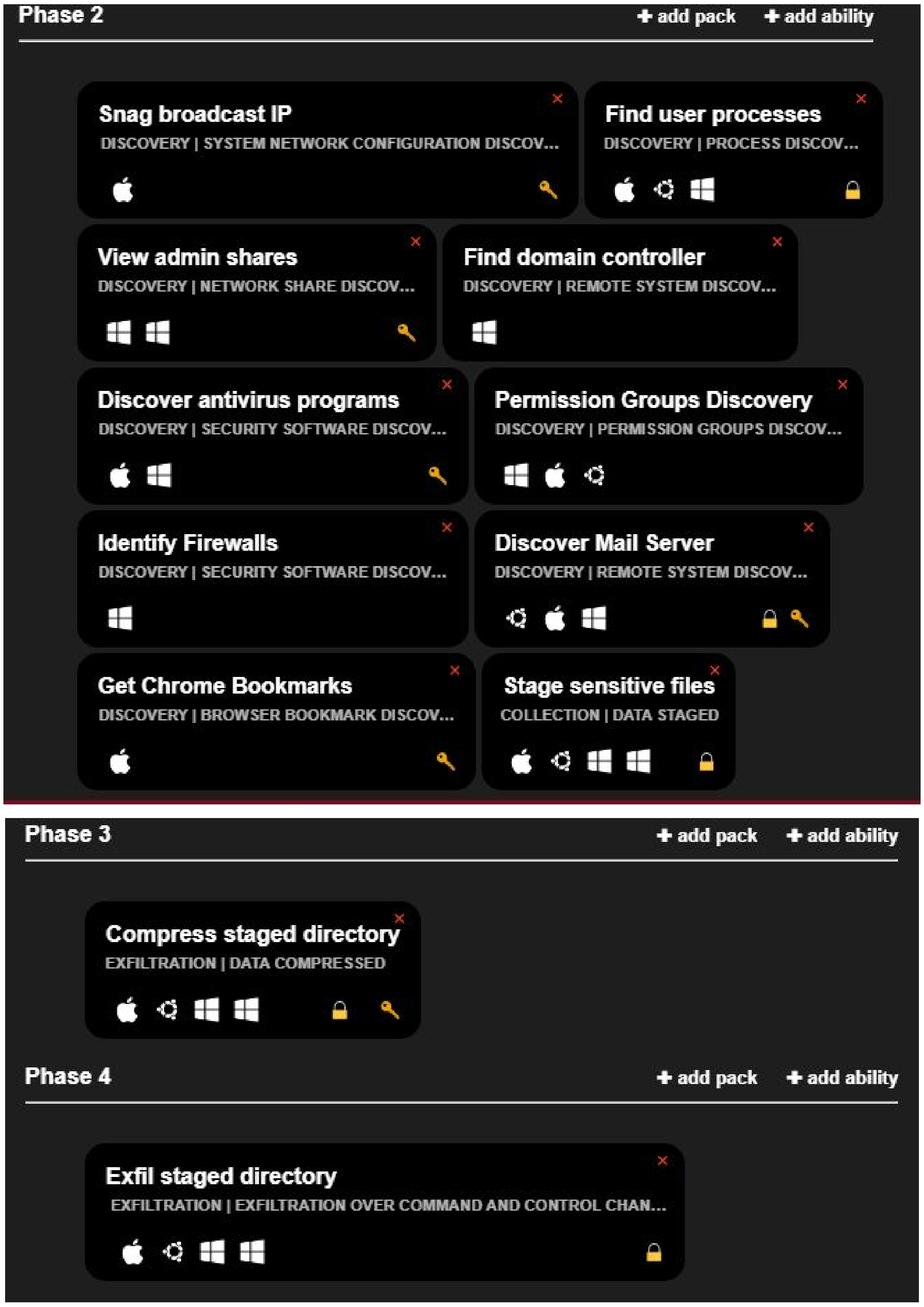
Find sensitive files
COLLECTION | DATA FROM LOCAL SYST...

Identify active user
DISCOVERY | SYSTEM OWNER/USER DISCOV...

Find local users
DISCOVERY | ACCOUNT DISCOV...

Identify local users
DISCOVERY | PERMISSION GROUPS DISCOV...

Create staging directory
COLLECTION | DATA STAGED



Phase 2 + add pack + add ability

Snag broadcast IP
DISCOVERY | SYSTEM NETWORK CONFIGURATION DISCOV...

Find user processes
DISCOVERY | PROCESS DISCOV...

View admin shares
DISCOVERY | NETWORK SHARE DISCOV...

Find domain controller
DISCOVERY | REMOTE SYSTEM DISCOV...

Discover antivirus programs
DISCOVERY | SECURITY SOFTWARE DISCOV...

Permission Groups Discovery
DISCOVERY | PERMISSION GROUPS DISCOV...

Identify Firewalls
DISCOVERY | SECURITY SOFTWARE DISCOV...

Discover Mail Server
DISCOVERY | REMOTE SYSTEM DISCOV...

Get Chrome Bookmarks
DISCOVERY | BROWSER BOOKMARK DISCOV...

Stage sensitive files
COLLECTION | DATA STAGED

Phase 3 + add pack + add ability

Compress staged directory
EXFILTRATION | DATA COMPRESSED

Phase 4 + add pack + add ability

Exfil staged directory
EXFILTRATION | EXFILTRATION OVER COMMAND AND CONTROL CHAN...

Analyzing the Caldera Operation in Splunk

Now that we have run a successful Caldera operation, we will use Splunk to see if we have visibility on what exactly it is that the Caldera operation accomplished.

Constructing the Initial Search

Isolate the Target Machine in Your Search

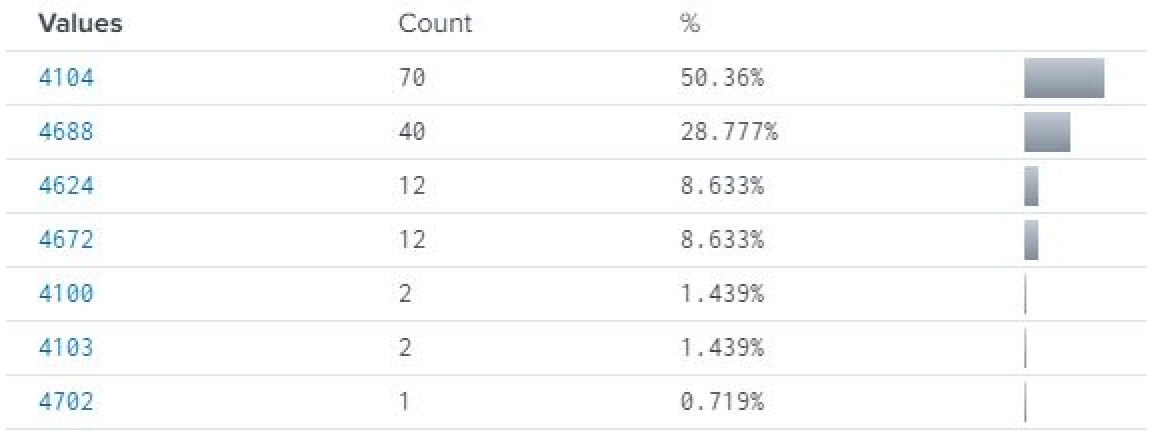
Isolate the target machine in your Splunk search so that you are only analyzing the events of the target machine. In my case, I have my target machine forwarding data to a specific index, so I will specify that index and host with the query:

```
index = "[IndexName]" host="[HostName]"
```

Set the Date/Time Range

Caldera does a good job with timestamps so that you know when an operation was started, and when it was completed. Change the Time Picker in Splunk so that it matches the window that the Caldera operation was running.

Win:Event Code Breakdown



Values	Count	%
4104	70	50.36%
4688	40	28.777%
4624	12	8.633%
4672	12	8.633%
4100	2	1.439%
4103	2	1.439%
4702	1	0.719%

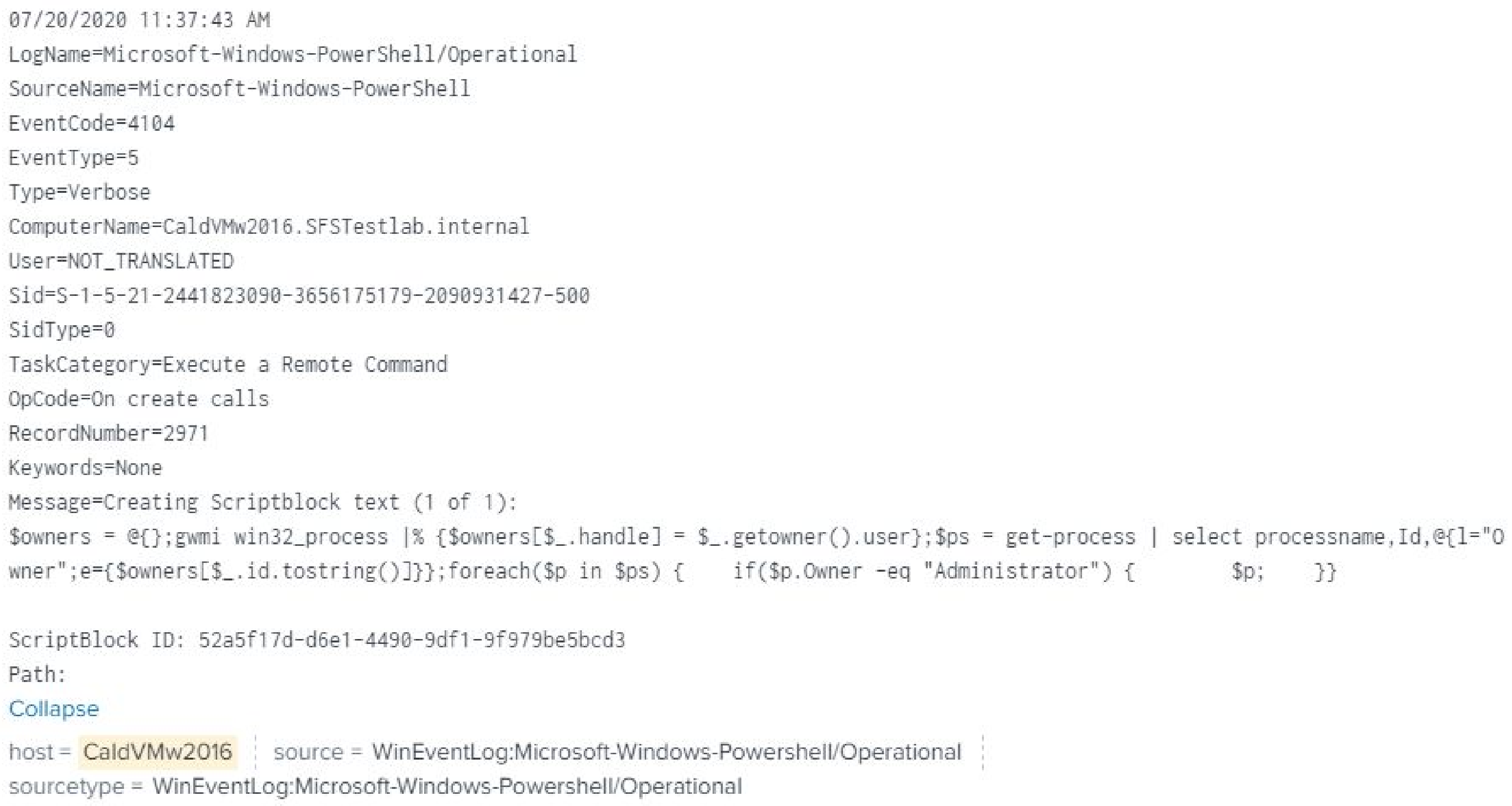
We can see that the majority of the events over the course of the Caldera operation were 4104. This means that a lot of Powershell operations were being run. Let’s take a deeper look and see what we can find.

Taking a Closer Look at Win:Event 4104

Use the query below to search through Event 4104 and sort by time.

```
index = "[IndexName]" host="[HostName]" EventCode="4104"
|sort _time
```

As I look through the Event 4104, almost every single one is related to the Caldera operation. Here is an example:



```
07/20/2020 11:37:43 AM
LogName=Microsoft-Windows-PowerShell/Operational
SourceName=Microsoft-Windows-PowerShell
EventCode=4104
EventType=5
Type=Verbose
ComputerName=CaldVMw2016.SFSTestlab.internal
User=NOT_TRANSLATED
Sid=S-1-5-21-2441823090-3656175179-2090931427-500
SidType=0
TaskCategory=Execute a Remote Command
OpCode=On create calls
RecordNumber=2971
Keywords=None
Message=Creating Scriptblock text (1 of 1):
$owners = @{};gwmi win32_process |% {$owners[$_.handle] = $_.getowner().user};$ps = get-process | select processname,Id,@{l="O
wner";e={$owners[$_.id.tostring()]}};foreach($p in $ps) {    if($p.Owner -eq "Administrator") {        $p;    }}

ScriptBlock ID: 52a5f17d-d6e1-4490-9df1-9f979be5bcd3
Path:
Collapse
host = CaldVMw2016 | source = WinEventLog:Microsoft-Windows-Powershell/Operational |
sourcetype = WinEventLog:Microsoft-Windows-Powershell/Operational
```

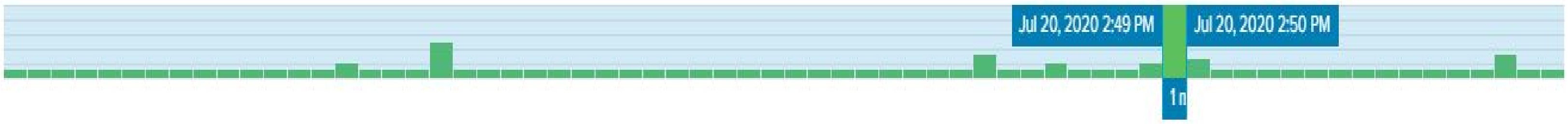
We can clearly see the script that Caldera ran in an effort to enumerate the current processes running on our target machine. This is why Script Block Logging is so useful.

Taking a Closer Look at Win:Event 4688

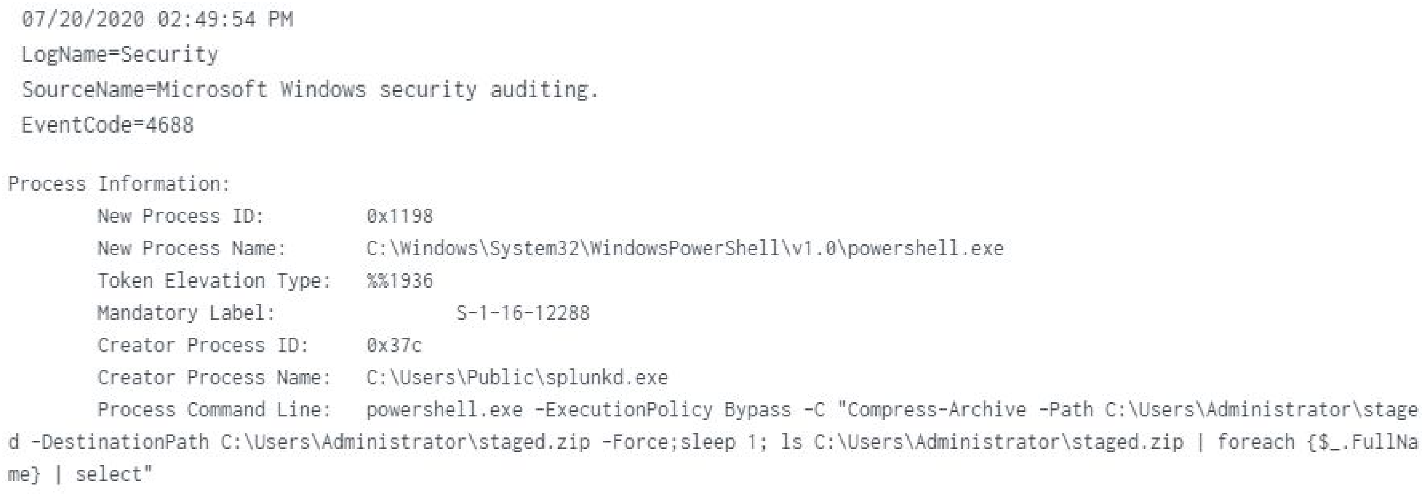
Use the query below to search through Event 4104. This will exclude the powershell commands that Splunk runs to collect/forward index data.

```
index="calderavm" EventCode = 4688  NOT
"Files\\SplunkUniversalForwarder"
```

You can look right under the search bar at the "Candlesticks" to see what moments in time had the highest volume of the event you are searching for (See below).



From the image above, we can see that the highest number of Event 4688 occurred between 2:49 and 2:50 PM. (I have my range set for 2-3 PM in the image above).



```
07/20/2020 02:49:54 PM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=4688

Process Information:
        New Process ID:         0x1198
        New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
        Token Elevation Type:   %%1936
        Mandatory Label:                S-1-16-12288
        Creator Process ID:     0x37c
        Creator Process Name:   C:\Users\Public\splunkd.exe
        Process Command Line:   powershell.exe -ExecutionPolicy Bypass -C "Compress-Archive -Path C:\Users\Administrator\stage
d -DestinationPath C:\Users\Administrator\staged.zip -Force;sleep 1; ls C:\Users\Administrator\staged.zip | foreach {$_.FullNa
me} | select"
```

Looking at the snippet of Event 4688 above, we can see that a new process named "`C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`" has been created. This looks just like a legitimate process. However, if we look at the Creator Process Name, we see "`C:\Users\Public\splunkd.exe.`" This is NOT an appropriate filepath for Splunkd.exe. It should not reside in the `C:\Users\Public\` directory. This tips us off to the fact that it is a Caldera event that we are looking at. It is very important to have a fundamental understanding of abnormal file paths so that you can recognize them whenever bad actors try to take advantage of them in your environment.

By looking at the Process command line (this is why enabling it is so important!), we can see the exact command that was run in powershell. It is Caldera extracting information from the staged.zip file that it created during its operation.

As we saw in the event log, Caldera was disguising itself as a splunkd.exe executable. Caldera is dynamic in what it disguises itself as, I have seen it guise as winword.exe, svchost.exe, and as we saw here, splunkd.exe. This is why knowing native file paths is important.

Splunk License Usage

What is it?

The School of Engineering and Technology is not the primary owner of the Splunk license for the University of Washington. We are licensed as a slave under the current agreement. What this means is that we do not have access to the raw license usage source that we would if we were the primary license owner. In order to get answers about license usage, we need to contact help@uw.edu. This makes it difficult for us to make accurate estimates when it comes to predicting how much data we are consuming in regards to our license. It also makes it harder to estimate how much more data we might require if we wanted to expand our Splunk environment.

Data Supplied by License Master

Here's the current usage for our pool (as of July 15, 2020):

```
"_time",volume(GB)
"2020-07-08","0.123"
"2020-07-09","0.126"
"2020-07-10","0.126"
"2020-07-11","0.126"
"2020-07-12","0.130"
"2020-07-13","0.127"
"2020-07-14","0.124"
"2020-07-15","0.126"
```

Tstats Command

Tstats is faster than stats since tstats only looks at the indexed metadata (the .tsidx, aka time series index files, in the buckets on the indexers) whereas stats is working off the data (in this case the raw events) before that command.

Since tstats can only look at the indexed metadata it can only search fields that are in the metadata. By default, this only includes index-time fields such as sourcetype, host, source, _time, etc. This is perfect in our case because we are working with host and sourcetype.

Determining Number of Hosts per Day Over a Timeframe

This query will show how many hosts were sending data to splunk each day (span=1d) based on what the user specifies in the Time Picker. (If you run it July 08-13 you will see some fluctuation in the number of hosts daily, that is because I was installing/uninstalling the universal forwarder trying to get things operational. I also changed the computer name of w2016 from default to "CaldVMw2016")

I am using tstats here for speed, you should change the search span in the Time Picker to last 7 days (last 30 days works too! It's a quick search)

```
| tstats count WHERE index=* sourcetype=* by _time, host
| timechart span=1d dc(host) AS unique_host_count
```

We are currently monitoring 24 hosts.

Finding the Number of Events indexed per Host per Day

1.Query for finding the number of events indexed per host per day (using _internal to include Splunk logs)

```
| tstats count as "event count" where index=_internal groupby host
```

2.Query for finding the TOTAL number of events in a given day (set day/time frame by using Time Picker)
Note You can also just look at the total number of events for search #1 (located right under the search bar)

```
| tstats count as "event count" where index=_internal groupby host
| stats sum("event count") as "Total Event Count"
```

3.Query for finding the number of events indexed per host per day (using "main" to exclude Splunk logs)

```
| tstats count as "event count" where index=main groupby host
```

4.Query for finding the number of events indexed per host per day (using "main" to exclude Splunk logs)
Note You can also just look at the total number of events for search #3 (located right under the search bar)

```
| tstats count as "event count" where index=main groupby host
| stats sum("event count") as "Total Event Count"
```

Correlation Between Number of Hosts and Amount of Data Ingested

Executive Summary (TLDR): The higher the number of events on a given day do not necessarily equate to a higher amount of data being ingested by Splunk.

I wanted to run the commands I referenced above over each day that we got license usage information and find out if a higher number of events corresponded to a higher number data ingested by Splunk. Here are the results.

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 8, 00:00 -July 8, 24:00)
Results: 5,469,230 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 9, 00:00 -July 9, 24:00)
Results: 5,514,268 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 10, 00:00 -July 10, 24:00)
Results: 5,517,109 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 11, 00:00 -July 11, 24:00)
Results: 5,451,421 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 12, 00:00 -July 12, 24:00)
Results: 5,451,307 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 13, 00:00 -July 13, 24:00)
Results: 5,231,978 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 14, 00:00 -July 14, 24:00)
Results: 5,104,584 total events (See right beneath search bar for total)

Query: `| tstats count as "event count" where index=_internal groupby host`
Timeframe: (July 15, 00:00 -July 15, 24:00)
Results: 5,351,560 total events (See right beneath search bar for total)