WHITE PAPER

Splunkmon — Taking Sysmon to the Next Level

KEY POINTS

- Sysmon can fill gaps in visibility into Windows processes and command line arguments
- Sysmon can be leveraged to identify threat actor reconnaissance, lateral movement, credential harvesting, and data collection
- Sysmon can identify network connections made by specific processes
- Sysmon events can be forwarded to Splunk (or other SIEMs) for searching, alerting, and dashboard visualizations

By Alec Randazzo, Thomas Aneiro, and James Espinosa

CRYPSIS™

www.crypsisgroup.com

Contents

ABOUT THE AUTHORS

Alec Randazzo Alec Randazzo is a Principal Consultant in the Crypsis Group's McLean, Virginia office. He has over four years of experience responding to data breaches on behalf of organizations across a variety of industries and ranging in size from small start-ups to Fortune 100 corporations.

Thomas Aneiro has more than three years of experience working in information technology and security operations. He leverages his operational expertise to work with clients to maximize their Splunk® instances by evaluating what data is most value in their environment.

James Espinosa is a Senior Consultant in the Crypsis Group's Chicago office. Mr. Espinosa has over four years of experience working in security operations and responding to data breaches on behalf of organizations ranging from start-ups to Fortune 500 corporations.

Introdution

The success of security teams when combating cyber threats relies on visibility into the environment they must defend. Security teams that do not have visibility into their endpoints may not see a threat actor moving in the environment, stealing credentials, deploying backdoors, or exfiltrating sensitive data. The goal of this white paper is to illustrate how security teams can close the visibility gap into Windows endpoints through the use of the free Sysinternals tool Sysmon[1] with Splunk®. Sysmon paired with Splunk can provide near real time visibility and alerting on the common actions targeted threat actors perform during an attack. Additionally, Sysmon paired with Splunk provides an excellent platform to proactively hunt for evidence of compromise in an environment.

This paper will provide the following:

- An Overview of Sysmon
- Tips for Pairing Sysmon with Splunk
- Scenarios showing what threat actors do and how Sysmon records their activity
- Splunk searches for hunting and alerting

Sysmon

Sysmon is a robust, free utility from the Sysinternals Suite capable of logging:

- Processes
- Process handles
- Network connections made by processes
- Driver loading
- Raw disk read and writes
- File creation timestamp modification
- Registry key creation, modification, deletion
- File creation

Sysmon logs these events in its own separate operational event log.[2] Sysmon is also highly configurable. Not only can we enable or disable features listed above, but we can also configure what each feature logs. For instance, if Sysmon is generating process creation events for a binary that we know is legitimate, we can configure Sysmon to drop all process creation events by that binary. This configurability and the logging of events to its own event log makes it a preferable alternative to Security event process auditing.

[1] https://technet.microsoft.com/en-us/sysinternals/sysmon

[2] The Sysmon operational event log is stored in “C:\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx”

Configuration

In its default setting, Sysmon logs more data than is necessary for hunting and alerting. We will focus on Process Creation and Network Connection events with a custom Sysmon configuration that weeds out a lot of noise. Crypsis provides a basic Sysmon configuration here. Note that the configuration will need to be modified further to your specific environment.

Installation

To install Sysmon, use the following command:

```
Sysmon.exe -i -h MD5,IMPHASH -n
```

After installation, load the custom configuration file with the following command:

```
Sysmon.exe -c sysmon.cfg
```

Upon installation, Sysmon will begin logging events to the operational event log "C:\Windows\System32\winevt\Logs\Microsoft-Windows-Sysmon%4Operational.evtx".

Event Examples

Event ID 1: Process Create

Figure 1 contains an example of a Sysmon "process create" event. Fields highlighted in Figure 1 are of particular interest from a hunting and alerting perspective.

```
Process Create:
UtcTime: 2016-12-04 21:19:16.392
ProcessGuid: {efbfc260-8854-5844-0000-001045448919}
ProcessId: 11920
Image: C:\PerfLogs\rar.exe
CommandLine: rar.exe  a -hp123qwe data.rar paystubs
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-1a43-5843-0000-0020963bbd18}
LogonId: 0x18BD3B96
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=5DC9AAF3B27B54CD7981F4D28187CBC7,IMPHASH=4397DAB9E39CC23F35F79B293DDAD7
80
ParentProcessGuid: {efbfc260-8782-5844-0000-00106b438119}
ParentProcessId: 15976
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "cmd.exe" /s /k pushd "C:\PerfLogs"
```

Figure 1: Example process creation Sysmon event

Fields of interest include:

- **Image:** The file path for the new process being spawned.
- **CommandLine:** Command line arguments passed to the new process being spawned.
- **CurrentDirectory:** The current working directory of the process. For instance, if a command prompt has its current working directory being "C:\Perflogs", then "C:\Perflogs" would be logged as the "CurrentDirectory". This field is very useful if there is a known threat using unique current working directories.
- **User:** The user how spawned the new process.
- **LoginId:** Login ID of the user who spawned the new process. This field is useful if you have suspicious activity and want to verify it by seeing what other processes were spawned during the login session.
- **Hashes:** Computed hashes of the file in the "Image" field. Hashes can be configured to log MD5, SHA1, SHA256, and IMPHASH.[3]
- **ParentProcessId:** Process ID of the parent process.
- **ParentImage:** File path of the parent process.

Event ID 3: Network Connection Detected

Figure 2 contains an example of a Sysmon "network connection detected" event. The highlighted fields are of interest from a hunting and alerting perspective.

```
Network connection detected:
UtcTime: 2016-10-17 18:37:13.497
ProcessGuid: {efbfc260-1a59-5805-0000-00108c2cc405}
ProcessId: 13740
Image: C:\windows\system32\printfltr.exe
User: NT AUTHORITY\SYSTEM
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.100.1.51
SourceHostname: DESKTOP-ASK78MS
SourcePort: 55064
SourcePortName:
DestinationIsIpv6: false
DestinationIp: 123.123.123.123
DestinationHostname:
DestinationPort: 443
DestinationPortName: https
```

Figure 2: Example network connection Sysmon event

[3] IMPHASH is a hash of the import table of a PE binary. This kind of hash is useful in finding malware variants.

Fields of interest include:

- **Image:** File path of the process that initiated the network connection. This field is extremely useful for all those instances where you have evidence of malicious network traffic but are unsure of what process is making the connections.
- **User:** User under whom the process is running under.
- **SourceHostname:** Hostname of the source system.
- **DestinationIp:** Destination IP address.
- **DestinationPort:** Destination port.

Sysmon and Splunk

Splunk is a great tool for gathering and dissecting large amounts of data in an organized statistical format, whereas Sysmon is a great tool for providing very granular insight into what is happening on a system. Combining the two provides an efficient means to optimize the best assets of both toolsets. Dashboards, saved searches, and alerts can be used to do much of the heavy lifting and take some of the more manual process out of hunting or operational monitoring. Personnel can be an organization's most over-taxed asset, creating automated and repeatable processes can allow the more senior members to hand down tasks, as well as knowledge, to the more junior analysts.

Adrian Hall created an excellent Splunk App[4] that's presently maintained by David Herrald that handles field extractions for Sysmon. We highly recommend leveraging this app to quickly get up to speed.

Creating Monitoring Dashboards

Visualizations are one of the easiest way for the human eye to detect anomalies. Graphs and maps give the brain a format in which to identify patterns and outliers with the least amount of processing required. Mapping network connections can show you where processes are connecting to. Alongside these we can use statistical data to correlate and support or disprove any conclusions that graphs and charts may suggest. For instance, a small business with primarily domestic dealings can create a visual mapping of processes logged by Sysmon performing network connections outside of the US as seen in Figure 3.

[4] https://splunkbase.splunk.com/app/1914/

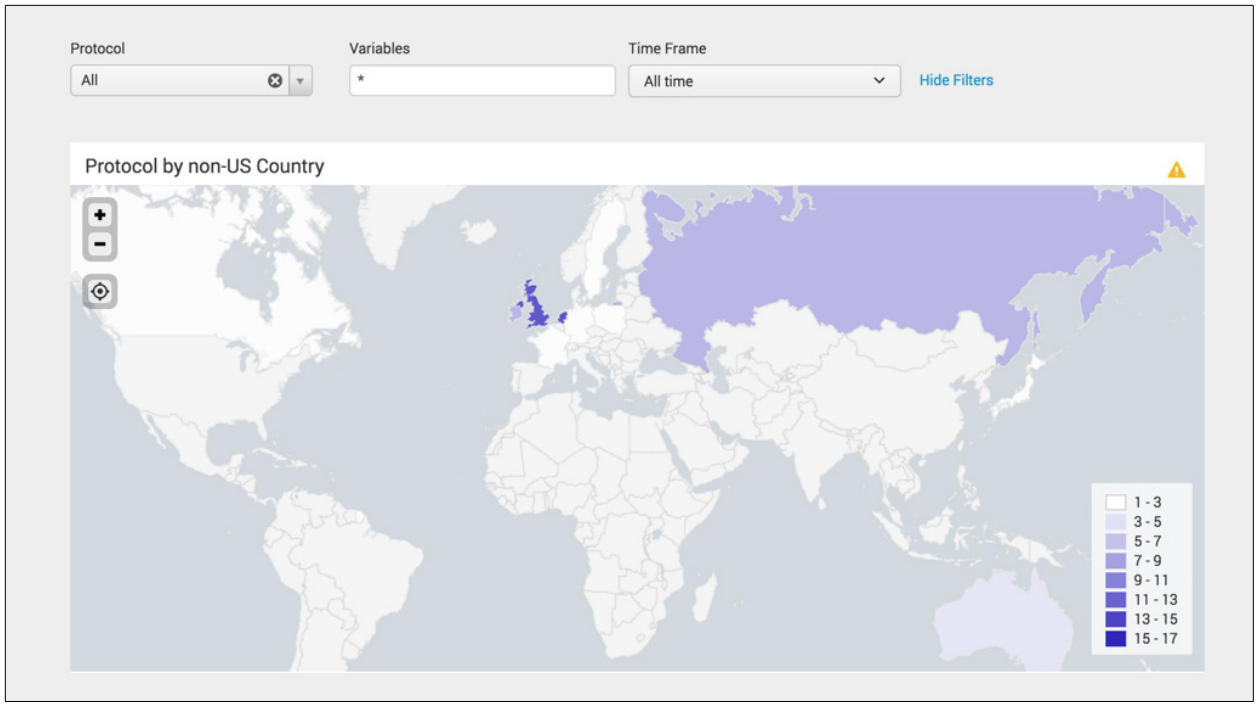

Figure 3: Visual mapping of process network connection Sysmon events

In support of visual mappings, statistical searches can be added, such as those shown in Figure 4. The statistical search will find processes opening network connections made at regular intervals. In the example shown, the parameters are looking for host and process image combinations that have a count over 5 times and a standard deviation of under 1x between connections. Ideally finding a standard deviation of 0 and average time to next run being an integer could find persistence mechanisms or C2 behavior. There will need to be a degree of review and filtering as this will also find benign connections.

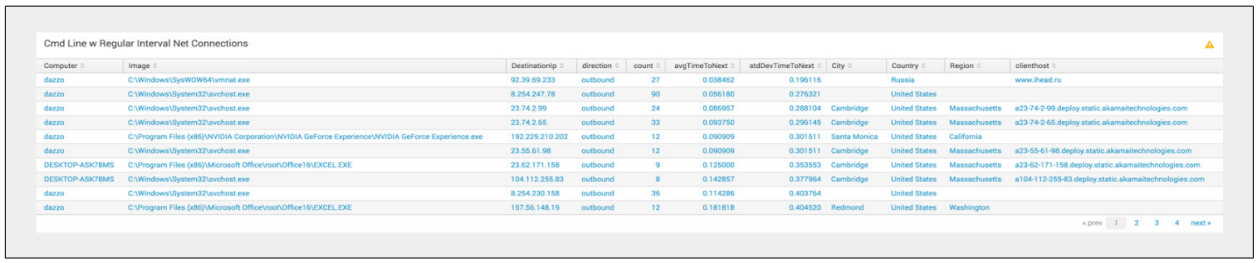

Figure 4: Statistical search of processes opening network connections

The following Splunk search can be used to perform this statistical search.

- index=sysmon EventCode=3 | streamstats window=1 global=f current=f last(_time) as next_ts by Computer Image DestinationIp | eval tm_to_next=next_ts-_time | stats values(direction) as direction count avg(tm_to_next) as avgTimeToNext stdev(tm_to_next) as stdDevTimeToNext by Computer Image DestinationIp | where stdDevTimeToNext < 1 AND count > 5 AND avgTimeToNext > 0| iplocation DestinationIp | lookup dnslookup clientip as DestinationIp | fields - lat lon | sort stdDevTimeToNext

Triggering Alerts from Splunk

Splunk allows for a variety of methods to trigger alerts, allowing for increased functionality and the ability to reach your target audience in the best means available. Splunk has ticketing abilities in its paid Enterprise Security app or can be connected via API to several other ticketing systems such as ServiceNow, Remedy, etc.

Alternatively, if an alert is urgent there are options for pushing alerts via other means of communications, such as email or chat. The email option is straightforward configure. Integrating with chat, such as Slack or Hipchat, require additional apps to be installed on the Splunk search head. Figure 5 shows an example alert configuration for Slack using the Splunk Slack app.[5]

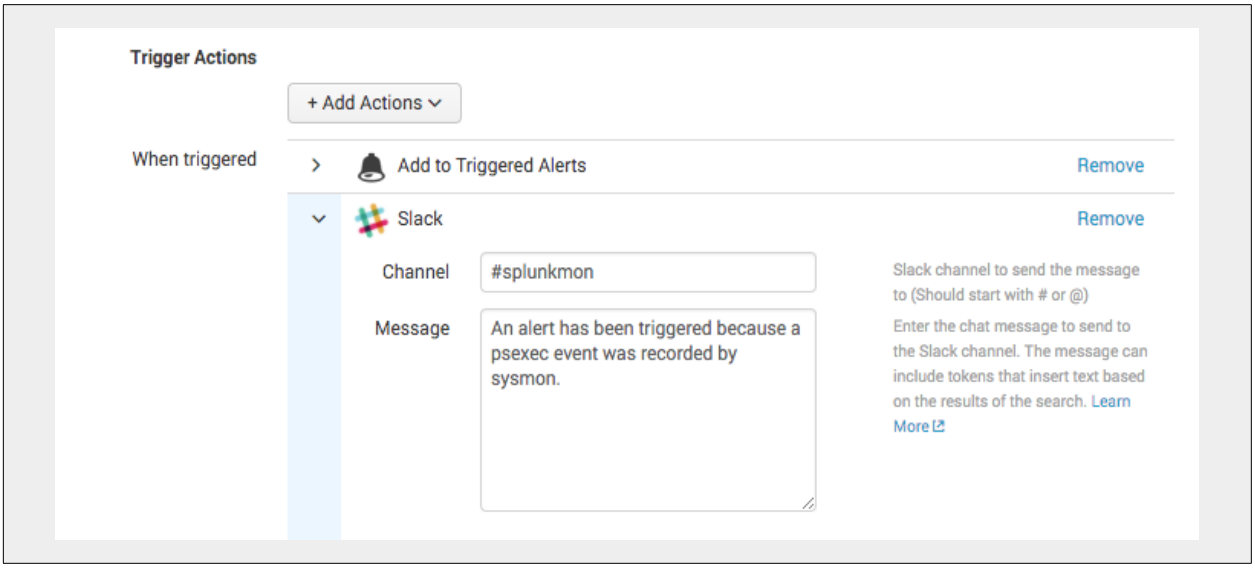

Figure 5: Creating a trigger to alert on Slack

Figure 6 shows what the triggered alert would look like in Slack if Splunk detected a PsExec process event.

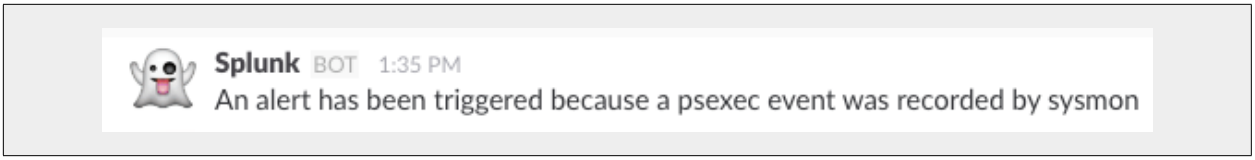

Figure 6: Example alert as seen in Slack

[5] https://splunkbase.splunk.com/app/2878/

Threat Actor Activity Scenarios

This section contains a variety of scenarios that walk through actions a threat actor may perform and how Sysmon would record them. At the end of each scenario there is a listing of Splunk searches used to detect the variety of actions the threat actor took in each scenario. Appendix A: Splunk Searches contains an aggregate list of Splunk searches for easier referencing.

Scenario: "net" Reconnaissance of Domain Admin Group

Threat actors will commonly use Windows "net"[6] commands to enumerate data from local systems and from the domain. The "net" commands are particularly useful for enumerating domain information such as:

- Users in a domain
- Groups in a domain
- Users in a group in a domain
- Computers in a domain

Threat actors will often perform numerous "net" queries in a short period of time to try and map out an environment early in the attack lifecycle. Given this knowledge, we can create Splunk searches that can be used to hunt for and alert on potential threat actor reconnaissance. We would want to especially look into instances of many various "net" commands from a single system in a short period of time. In our scenario, we will be focusing on what it would look like in Sysmon if a threat actor were to enumerate the users in the domain group "Domain Admin" using the command listed below.

Command

- net group "Domain Admin" /domain

Sysmon Events on Local System

Figure 7 contains the event logged by Sysmon of the "net" command. Note that in the Sysmon event, there are two spaces between "net" and "group" in the "CommandLine" field, which is standard when passing arguments to a new process. If we want to hunt for and alert on "net group" commands, we can create a search that looks for "CommandLine" contains "net group". False positives with this search may occur due to legitimate administrator activity; you may need to validate results by reviewing additional commands executed from the parent "cmd.exe" process from that system.

If we want to create a higher fidelity alert, we could search for instances where someone is enumerating the users in the organization's respective domain administrator and other privileged groups, which are a threat actor's prime targets for credential theft. This could be done in our example with a search for "CommandLine" contains "net group" and "CommandLine" contains "Domain Admin".

[6] https://support.microsoft.com/en-us/kb/556003

```
Process Create:
UtcTime: 2016-12-04 21:43:33.705
ProcessGuid: {efbfc260-8e05-5844-0000-0010f1ad9519}
ProcessId: 11220
Image: C:\Windows\System32\net.exe
CommandLine: net  group "Domain Admin" /domain
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-1a43-5843-0000-0020963bbd18}
LogonId: 0x18BD3B96
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=9B1E2A711EA151F766EA24389E2D4442,IMPHASH=C41B15F592DE4589047CE5119CE874
68
ParentProcessGuid: {efbfc260-8782-5844-0000-00106b438119}
ParentProcessId: 15976
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "cmd.exe" /s /k pushd "C:\PerfLogs"
```

Figure 7: Sysmon event generated by the "net group" command on the local system

Splunk Searches

The Splunk search examples below cover our above scenario and additional scenarios of other "net" commands.

- index=sysmon CommandLine="net group*" CommandLine="*/domain*"
- index=sysmon CommandLine="net group*" CommandLine="*Domain Admin*"
- index=sysmon CommandLine="net user*" CommandLine="*/domain*"
- index=sysmon CommandLine="net view*" CommandLine="*/domain*"

Scenario: Credential Harvesting with WMI and WCE

The following scenario showcases what Sysmon would record if a threat actor used Windows Management Instrumentation ("WMI") to execute the credential harvesting tool Windows Credential Editor ("WCE") on a remote system. The lessons learned from this scenario will be applicable to both the use of WMI and WCE by themselves. In order to execute WCE (renamed to "w.exe") on the remote system named SANDBOX, the threat actor will need to mount the remote system as a network share, copy "w.exe" to the remote system, and use WMI to execute WCE on the remote system. The three commands below will achieve this goal.

Commands

- net use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
- copy w.exe \\172.31.3.16\c$\PerfLogs
- wmic /NODE:172.31.3.16 /USER:"SANDBOX\Administrator" /PASSWORD:"PASSWORD" process call create "cmd /c C:\Perflogs\w.exe -w > C:\Perflogs\o.txt"

Sysmon Events on Local System

The “net use” command generated a Sysmon event with the command line arguments present in the “CommandLine” field as seen in Figure 8. Unfortunately, since “net use” commands are common in environments, it will not be preferable from a generic search and alert perspective. In our particular circumstance, if we knew that this “SANDBOX\Administrator” account was compromised, we could perform searches for “net use” commands leveraging that account. Additionally, if we know there is a compromise and the attacker used “C:\Perflogs” as their preferred directory, we can search for “cmd.exe” parent processes spawning new processes while the “CurrentDirectory” was “C:\Perflogs”.

```
Process Create:
UtcTime: 2016-11-28 13:36:06.076
ProcessGuid: {efbfc260-32c6-583c-0000-001019486014}
ProcessId: 11236
Image: C:\Windows\System32\net.exe
CommandLine: net  use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=9B1E2A711EA151F766EA24389E2D4442,IMPHASH=C41B15F592DE4589047CE5119CE874
68
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
```

Figure 8: Sysmon event generated by the “net use” command on the local system

The “copy” command did not generate a Sysmon event due to it being an internal “cmd.exe” command. The WMI command generated the Sysmon event in Figure 9. With this Sysmon event, there are two generic Splunk searches that can be used. First, if we want to find all instances of WMI being executed over the network, we can search for all instances where the “Image” contains “WMIC.exe” and the “CommandLine” contains “ /NODE:”. Second, we can do a search for instances of WMI being used to create a new process by searching for all instances of “Image” containing “WMIC.exe” and “CommandLine” containing “process call create”.

```
Process Craeate:
UtcTime: 2016-11-28 13:42:45.616
ProcessGuid: {efbfc260-3455-583c-0000-0010550e6814}
ProcessId: 19400
Image: C:\Windows\System32\wbem\WMIC.exe
CommandLine: wmic  /NODE:172.31.3.16 /USER:"SANDBOX\Administrator" /PASSWORD:"PASS
WORD" process call create "cmd /c C:\Perflogs\w.exe -w > C:\Perflogs\o.txt"
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=2CEE7F1AD77D8817E0F043E5E5ED1C83,IMPHASH=1B1A3F43BF37B5BFE60751F2EE2F32
6E
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 9: Sysmon event generated by the WMI command on the local system

Sysmon Events on Remote System

On the remote system, Sysmon recorded that the WMI process “WmiPrvSE.exe” passed the arguments “cmd /c C:\Perflogs\w.exe -w > C:\Perflogs\o.txt” to a new command prompt process as seen below. There are two pieces of information in the event in FIgure 10 that we can leverage. First, we can search for all instances of WMI spawning a new process by searching for “ParentImage’ contains “WmiPrvSE.exe”. Second, we can search for instances where a “-w” flag is passed and the writing of standard out to a file by searching for “Image” contains “cmd.exe” and “CommandLine” contains “ -w > ”.

```
Process Create:
UtcTime: 2016-11-28 13:28:04.228
ProcessGuid: {053d9c77-30e4-583c-0000-00101440ff2e}
ProcessId: 3892
Image: C:\Windows\System32\cmd.exe
CommandLine: cmd /c C:\Perflogs\w.exe -w > C:\Perflogs\o.txt
CurrentDirectory: C:\Windows\system32\
User: SANDBOX\Administrator
LogonGuid: {053d9c77-30e4-583c-0000-00203f3fff2e}
LogonId: 0x2eff3f3f
TerminalSessionId: 0
IntegrityLevel: High
Hashes: MD5=6960D29ABE74341FAB8300DB3E6F883D,IMPHASH=B1F6C2E6146D6A8FD41C6869A26958
36
ParentProcessGuid: {053d9c77-418a-5817-0000-001092f50100}
ParentProcessId: 1236
ParentImage: C:\Windows\System32\wbem\WmiPrvSE.exe
ParentCommandLine: C:\Windows\system32\wbem\wmiprvse.exe -secured -Embedding
```

Figure 10: Sysmon event generated by the WMI command on the remote system

The command prompt process then created the WCE process and passed the "-w" flag to it which dumped clear text passwords from memory as seen in Figure 11. From a hunting and alerting perspective, we can search for IMPHASHES that match the WCE IMPHASH to find WCE and potential variants.

```
Process Create:
UtcTime: 2016-11-28 13:28:04.244
ProcessGuid: {053d9c77-30e4-583c-0000-00103441ff2e}
ProcessId: 4604
Image: C:\PerfLogs\w.exe
CommandLine: C:\Perflogs\w.exe  -w
CurrentDirectory: C:\Windows\system32\
User: SANDBOX\Administrator
LogonGuid: {053d9c77-30e4-583c-0000-00203f3fff2e}
LogonId: 0x2eff3f3f
TerminalSessionId: 0
IntegrityLevel: High
Hashes: MD5=BE9387BF647993E501C5D78E49BD4AB5,IMPHASH=8AB93B061287C79F3088C5BC7E7D97
ED
ParentProcessGuid: {053d9c77-30e4-583c-0000-00101440ff2e}
ParentProcessId: 3892
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd /c C:\Perflogs\w.exe -w > C:\Perflogs\o.txt
```

Figure 11: Sysmon event generated upon execution of WCE on the remote system

The Windows binary "services.exe" spawned a process for the random character GUID binary as seen in Figure 12. We have a perfect combination of data to search with from the GUID name for the binary, the parameter "-S" being passed to it, and the "ParentImage" being "services.exe". In every instance that WCE executes, regardless of the flag passed for password dumping, this combination would occur. If we want a very high fidelity search signature for WCE, we can search for "ParentImage" contains "services.exe", and "CommandLine" contains a regex that matches on the GUID with "-S" being passed to it.

```
Process Create:
UtcTime: 2016-11-28 13:28:04.290
ProcessGuid: {053d9c77-30e4-583c-0000-00105e42ff2e}
ProcessId: 5876
Image: C:\Users\ADMINI~1\AppData\Local\Temp\a67f3685-9f2c-4c89-8b4f-27776a15a988.
exe
CommandLine: C:\Users\ADMINI~1\AppData\Local\Temp\a67f3685-9f2c-4c89-8b4f-
27776a15a988.exe -S
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {053d9c77-4176-5817-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=605560CA0624AABF9F53675257B9BE21,IMPHASH=E96A73C7BF33A464C510EDE582318B
F2
ParentProcessGuid: {053d9c77-4175-5817-0000-0010c8910000}
ParentProcessId: 468
ParentImage: C:\Windows\System32\services.exe
ParentCommandLine: C:\Windows\system32\services.exe
```

Figure 12: Sysmon event generated by the execution of WCE on the remote system

Splunk Searches

Listed below are Splunk searches that would cover the use of Windows Credential Editor and WMI by threat actors.

Window Credential Editor (“WCE”)

- index=sysmon ParentImage="*\\services.exe" | regex CommandLine="\\\\[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12}\.exe -S$"
- index=sysmon Image="*\\cmd.exe" CommandLine="* -w > *"
- index=sysmon Hashes="*IMPHASH=8AB93B061287C79F3088C5BC7E7D97ED*"

WMI

- index=sysmon Image="*\\WMIC.exe" CommandLine="*process call create*"
- index=sysmon ParentImage="*\\WmiPrvSE.exe" | stats values(user) dc(LogonGuid) count by Computer CommandLine
- index=sysmon Image="*\\WMIC.exe" CommandLine="*/NODE:*"

Scenario: Credential Harvesting with PsExec and gsecdump

The following scenario showcases what Sysmon would record if a threat actor used the Sysinternals tool PsExec (renamed to “p.exe”) to execute the credential harvesting tool gsecdump (renamed to “gsec.exe”) on a remote system. The lessons learned from this scenario will be applicable to both the use of PsExec and gsecdump by themselves. In order to execute “gsec.exe” on the remote system, a threat actor can use the following single command.

Command

- p.exe -s \\172.31.3.16 -u SANDBOX\Administrator -p PASSWORD -c "gsec.exe" -a

Sysmon Events on Local System

Upon sending the command, the command prompt created a new process for “p.exe” and passed all the arguments to that process as seen in Figure 13. From a hunting perspective, there are two searches that can be used. First, since the PsExec binary tends to be consistent between threat actors, we can search for just it’s IMPHASH. Second, since threat actors will often rename their tools to hide what they are, we could search for all instances of the PsExec IMPHASH where the “Image” field does not contain “PsExec.exe” to find all instances of renamed PsExec that were executed.

```
Process Create:
UtcTime: 2016-11-28 17:57:40.924
ProcessGuid: {efbfc260-7014-583c-0000-00100ab05516}
ProcessId: 13360
Image: C:\PerfLogs\p.exe
CommandLine: p.exe  -s \\172.31.3.16 -u SANDBOX\Administrator -p PASSWORD -c "gsec.
exe" -a
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=27304B246C7D5B4E149124D5F93C5B01,IMPHASH=C1E59519B5E5D84AF07AFA6F5A8625
F1
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 13: Sysmon event generated by PsExec on the local system

Sysmon Events on Remote System

PsExec copied a service binary named “PSEXESVC.EXE” over SMB to the remote system along with the gsecdump binary. The Windows binary “services.exe” then executed the “PSEXESVC.EXE” binary. The PsExec service binary “PSEXESVC.EXE” then executed “gsec.exe” as seen in Figure 14. If we wanted to search for all PsExec executions, we simply need to just search for “ParentImage” contains “PSEXESVC.EXE”. If PsExec is used legitimately by administrators in an environment, generic searches for PsExec execution will yield false positives.

```
Process Create:
UtcTime: 2016-11-28 17:43:16.087
ProcessGuid: {053d9c77-6cb4-583c-0000-00100f20542f}
ProcessId: 4172
Image: C:\Windows\gsec.exe
CommandLine: "gsec.exe" -a
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {053d9c77-4176-5817-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=57F222D8FBE0E290B4BF8EAA994AC641,IMPHASH=115914D47B7284D0CD16B1117458C2
6E
ParentProcessGuid: {053d9c77-6cb3-583c-0000-0010821e542f}
ParentProcessId: 2684
ParentImage: C:\Windows\PSEXESVC.exe
ParentCommandLine: C:\Windows\PSEXESVC.exe
```

Figure 14: Sysmon event generated by PsExec on the remote system

Upon execution, gsecdump dropped and executed a temporary binary that always follows the naming convention “g64-” followed by three random alphanumeric characters as seen in Figure 15. This temporary binary is an excellent term to key off of with a regex.

```
Process Create:
UtcTime: 2016-11-28 17:43:16.103
ProcessGuid: {053d9c77-6cb4-583c-0000-00103721542f}
ProcessId: 2224
Image: C:\Windows\Temp\g64-17a.exe
CommandLine: "gsec.exe" -a
CurrentDirectory: C:\Windows\system32\
User: NT AUTHORITY\SYSTEM
LogonGuid: {053d9c77-4176-5817-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: MD5=7E456D1136C832357909647A9EC66E2B,IMPHASH=336C7186B8744FA59DCBEDE7AED40C
38
ParentProcessGuid: {053d9c77-6cb4-583c-0000-00100f20542f}
ParentProcessId: 4172
ParentImage: C:\Windows\gsec.exe
ParentCommandLine: "gsec.exe" -a
```

Figure 15: Sysmon event generated upon the execution of gsecdump on the remote system

Splunk Searches

The Splunk searches below will find instances of PsExec and gsecdump usage by threat actors.

PsExec

- index=sysmon ParentImage="*\\PSEXESVC.EXE" | stats count by Image,Hashes
- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*"
- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*" NOT Image="*PsExec.exe"

gsecdump

- index=sysmon Image="*\\g64-*" | regex Image="\\\\g64-[a-z0-9]{3}\.exe$"

Scenario: Credential Harvesting with Scheduled Tasks and pwdump

The following scenario showcases what Sysmon would record if a threat actor were to use the Windows scheduled task binary "schtasks.exe" to execute the credential harvesting tool pwdump (renamed to "perflogs.exe") on a remote system. The commands listed below will copy the pwdump binary to a remote system, and use "schtasks.exe" to create, execute, and then delete the named scheduled task.

Commands

- net use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
- copy perflogs.exe \\172.31.3.16\c$\PerfLogs
- copy libeay32.dll \\172.31.3.16\c$\PerfLogs
- schtasks /Create /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD /SC ONLOGON /TN "GoogleUpdate" /TR "cmd /c C:\Perflogs\perflogs.exe > C:\Perflogs\pwd.txt"
- schtasks /Run /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD /TN "GoogleUpdate"
- schtasks /Delete /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD /TN "GoogleUpdate"

Sysmon Events on Local System

First the use of "net use" generated the Sysmon event seen in Figure 16. Unfortunately, from a generic hunting perspective, searching for "net use" commands will likely be noisy. In our circumstance, if we know that the "SANDBOX\Administrator" account was compromised, we could do searches for "net use" commands using that account. Additionally, if we know there was a compromise and the threat actor used "C:\Perflogs" as their preferred directory, searches can be done for "cmd.exe" spawning other processes while the "CurrentDirectory" was "C:\Perflogs".

```
Process Create:
UtcTime: 2016-11-28 16:02:09.801
ProcessGuid: {efbfc260-5501-583c-0000-0010b5a63f15}
ProcessId: 8372
Image: C:\Windows\System32\net.exe
CommandLine: net  use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=9B1E2A711EA151F766EA24389E2D4442,IMPHASH=C41B15F592DE4589047CE5119CE874
68
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 16: Sysmon event generated by the "net use" command on the local system

The commands for copying pwdump were absent in the Sysmon events since the "copy" command is an internal command of "cmd.exe". The next event as seen in Figure 17 is the creation of the "schtasks.exe" process that has the command line arguments to create a task on a remote system passed to it. From a searching perspective, searching for "CommandLine" contains "schtasks", "/Create" and "/s " would find all instances of "schtasks" being used to create a task over the network.

```
Process Create:
UtcTime: 2016-11-28 16:03:16.849
ProcessGuid: {efbfc260-5544-583c-0000-001067804415}
ProcessId: 16232
Image: C:\Windows\System32\schtasks.exe
CommandLine: schtasks  /Create /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD
/SC ONLOGON /TN "GoogleUpdate" /TR "cmd /c C:\Perflogs\perflogs.exe > C:\Perflogs\pwd.
txt"
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=EEB7A2162E4DBE32B56BEB84658483AE,IMPHASH=8AC94113AD25518D369E4EE37BEDAB
4F
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 17: Sysmon event generated upon execution of the task creation command on the local system

The Sysmon event in Figure 18 shows what would be recorded when "schtasks.exe" is used to run a task immediately over the network. A search for all instances where "CommandLine" contains "schtasks", "/Run", and "/s " would return all instances of tasks being forcibly run over the network.

```
Process Create:
UtcTime: 2016-11-28 16:03:29.633
ProcessGuid: {efbfc260-5551-583c-0000-00108d904415}
ProcessId: 10552
Image: C:\Windows\System32\schtasks.exe
CommandLine: schtasks  /Run /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD /TN
"GoogleUpdate"
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=EEB7A2162E4DBE32B56BEB84658483AE,IMPHASH=8AC94113AD25518D369E4EE37BEDAB
4F
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 18: Sysmon event generated upon execution of the task run command on the local system

The Sysmon event in Figure 19 shows what would be recorded when “schtasks.exe” is used to delete a task over the network. A search for all instances where “CommandLine” contains “schtasks”, “/Delete”, and “/s ” would return all instances of scheduled tasks being deleted over the network.

```
Process Create:a
UtcTime: 2016-11-28 16:03:38.192
ProcessGuid: {efbfc260-555a-583c-0000-0010fba04415}
ProcessId: 10652
Image: C:\Windows\System32\schtasks.exe
CommandLine: schtasks  /Delete /s 172.31.3.16 /U SANDBOX\Administrator /P PASSWORD
/TN "GoogleUpdate"
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-0284-5833-0000-0020c6a30d00}
LogonId: 0xDA3C6
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=EEB7A2162E4DBE32B56BEB84658483AE,IMPHASH=8AC94113AD25518D369E4EE37BEDAB
4F
ParentProcessGuid: {efbfc260-31b2-583c-0000-0010f0385714}
ParentProcessId: 8364
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 19: Sysmon event generated upon execution of the task deletion command on the local system

Sysmon Events on Remote System

On the remote system, the Windows binary “taskeng.exe” spawned a command shell and passed the arguments to it to execute pwdump as seen in Figure 20. If we were to search for all instances where “ParentImage” contains “taskeng.exe” we would see all instances of scheduled tasks executing on a system.

```
Process Create:
UtcTime: 2016-11-28 15:48:47.863
ProcessGuid: {053d9c77-51df-583c-0000-001014462f2f}
ProcessId: 3340
Image: C:\Windows\System32\cmd.exe
CommandLine: C:\Windows\system32\cmd.EXE /c C:\Perflogs\perflogs.exe > C:\Perflogs\
pwd.txt
CurrentDirectory: C:\Windows\system32\
User: SANDBOX\Administrator
LogonGuid: {053d9c77-41be-5817-0000-0020df400400}
LogonId: 0x440df
TerminalSessionId: 2
IntegrityLevel: High
Hashes: MD5=6960D29ABE74341FAB8300DB3E6F883D,IMPHASH=B1F6C2E6146D6A8FD41C6869A26958
36
ParentProcessGuid: {053d9c77-512f-583c-0000-0010d5212e2f}
ParentProcessId: 3524
ParentImage: C:\Windows\System32\taskeng.exe
ParentCommandLine: taskeng.exe {7295BF58-A368-4E97-8553-3ACD9E4BE26F} S-1-5-21-
610627558-401490655-2610694327-500:SANDBOX\Administrator:Interactive:[2]
```

Figure 20: Sysmon event generated upon execution of the scheduled task on the remote system

Unfortunately, the execution of pwdump on a system did not yield unique events like what we see with WCE and gsecdump as seen in Figure 21. From a generic hunting perspective, while not perfect, we can leverage the IMPHASH to find pwdump variants.

```
Process Create:
UtcTime: 2016-11-28 15:48:47.879
ProcessGuid: {053d9c77-51df-583c-0000-001046472f2f}
ProcessId: 4104
Image: C:\PerfLogs\perflogs.exe
CommandLine: C:\Perflogs\perflogs.exe
CurrentDirectory: C:\Windows\system32\
User: SANDBOX\Administrator
LogonGuid: {053d9c77-41be-5817-0000-0020df400400}
LogonId: 0x440df
TerminalSessionId: 2
IntegrityLevel: High
Hashes: MD5=D1337B9E8BAC0EE285492B89F895CADB,IMPHASH=B18A1401FF8F444056D29450FBC0A6
CE
ParentProcessGuid: {053d9c77-51df-583c-0000-001014462f2f}
ParentProcessId: 3340
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: C:\Windows\system32\cmd.EXE /c C:\Perflogs\perflogs.exe > C:\
Perflogs\pwd.txt
```

Figure 21: Sysmon event generated upon the execution of pwdump on the remote system

Splunk Searches

The Splunk searches below would find all instances of named scheduled tasks being interacted with over the network and instances of pdwump that are not packed or highly modified.

schtasks

- index=sysmon ParentImage="*\\taskeng.exe" | stats count by Computer,CommandLine
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Create*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Run*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Delete*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Change*" CommandLine="*/s *"

pwdump

- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*"

Scenario: Credential Harvesting with PowerShell and Mimikatz

The following scenario showcases what Sysmon would record if a threat actor used Windows PowerShell to execute the credential harvesting tool Mimikatz (renamed to "mimi.exe") on a remote system. The commands listed below can be used to copy the Mimikatz binaries to a remote system, and then PowerShell to execute Mimikatz.

Commands

- net use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
- copy mimi* \\172.31.3.16\c$\PerfLogs
- powershell Invoke-Command -ScriptBlock {C:\Perflogs\mimi.exe privileges::debug sekurlsa::logonpasswords exit} -ComputerName 172.31.3.16 -Credential SANDBOX\Administrator

Sysmon Events on Local System

First, "net use" generated the Sysmon event in Figure 22. Unfortunately, from a generic hunting perspective, searching for "net use" commands will likely be noisy. In our circumstance, if we know that the "SANDBOX\Administrator" account was compromised, we could perform searches for "net use" commands leveraging that account. Additionally, if we knew there was a compromise and the attacker used "C:\Perflogs" as their preferred directory, searches could be done for "cmd.exe" processes spawning processes while the "CurrentDirectory" was "C:\Perflogs".

```
Process Create:
UtcTime: 2016-12-08 15:33:38.781
ProcessGuid: {efbfc260-7d52-5849-0000-0010b210e805}
ProcessId: 6952
Image: C:\Windows\System32\net.exe
CommandLine: net  use \\172.31.3.16 PASSWORD /user:SANDBOX\Administrator
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-6f71-5849-0000-002058ea4205}
LogonId: 0x542EA58
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=9B1E2A711EA151F766EA24389E2D4442,IMPHASH=C41B15F592DE4589047CE5119CE874
68
ParentProcessGuid: {efbfc260-7d1a-5849-0000-001066dbe405}
ParentProcessId: 4644
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 22: Sysmon event generated by the "net use" command on the local system

Sysmon did not record the "copy" command due to it being an internal function of "cmd.exe". Upon execution of the PowerShell command, Sysmon recorded "cmd.exe" spawning the PowerShell process and passing all arguments to the new process as seen in Figure 23. From a hunting and alerting perspective, we can use "Image" contains "powershell.exe" and "CommandLine" contains "Invoke-Command" to identify instances of PowerShell remoting.

```
Process Create:
UtcTime: 2016-12-08 15:35:05.612
ProcessGuid: {efbfc260-7da9-5849-0000-00108861ec05}
ProcessId: 4492
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  Invoke-Command -ScriptBlock {C:\Perflogs\mimi.exe privi-
leges::debug sekurlsa::logonpasswords exit} -ComputerName 172.31.3.16 -Credential
SANDBOX\Administrator
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-6f71-5849-0000-002058ea4205}
LogonId: 0x542EA58
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=097CE5761C89434367598B34FE32893B,IMPHASH=CAEE994F79D85E47C06E5FA9CDEAE4
53
ParentProcessGuid: {efbfc260-7d1a-5849-0000-001066dbe405}
ParentProcessId: 4644
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\WINDOWS\system32\cmd.exe"
```

Figure 23: Sysmon event generated upon the execution of the PowerShell command on the local system

Sysmon Events on Remote System

The WinRM Remote PowerShell process "wsmprovhost.exe" on the remote host will spawn a process for Mimikatz with the command line arguments passed to it. From a hunting and alerting perspective, there are three fields we can work with as shown in Figure 24. First, we can create searches for the "CommandLine" field containing the unique strings "privileges::debug" or "sekurlsa::logonpasswords". Second, we could create a search for the Mimikatz IMPHASH. Third, if we wanted to hunt generically for all instances of PowerShell remoting being used, we could do a search for "ParentImage" contains "wsmprovhost.exe".

```
Process Create:
UtcTime: 2016-12-08 15:19:54.671
ProcessGuida: {053d9c77-7a1a-5849-0000-0010a8f5e605}
ProcessId: 4008
Image: C:\PerfLogs\mimi.exe
CommandLine: "C:\Perflogs\mimi.exe"  privileges::debug sekurlsa::logonpasswords exit
CurrentDirectory: C:\Users\Administrator\Documents\
User: SANDBOX\Administrator
LogonGuid: {053d9c77-c891-5848-0000-002015bc4c04}
LogonId: 0x44cbc15
TerminalSessionId: 0
IntegrityLevel: High
Hashes: MD5=0E1BFBBCBA4A9F32A67C697AC5E61E83,IMPHASH=3ED0D1F8A6FB0E255E1ED340B1A3EA
AB
ParentProcessGuid: {053d9c77-7a1a-5849-0000-00103fefe605}
ParentProcessId: 3228
ParentImage: C:\Windows\System32\wsmprovhost.exe
ParentCommandLine: C:\Windows\system32\wsmprovhost.exe -Embedding
```

Figure 24: Sysmon event generated by PowerShell on the remote system

Splunk Searches

The Splunk searches listed below would identify the activity from this scenario and other instances of PowerShell and Mimikatz usage.

PowerShell

- index=sysmon Image="*powershell.exe" CommandLine="*Invoke-Command*"
- index=sysmon ParentImage="*wsmprovhost.exe" | stats count by Computer,CommandLine
- index=sysmon Image="*powershell.exe" CommandLine="*-EncodedCommand*"
- index=sysmon Image="*powershell.exe" | stats count by Computer,CommandLine

Mimikatz

- index=sysmon Hashes="*IMPHASH=3ED0D1F8A6FB0E255E1ED340B1A3EAAB*"
- index=sysmon CommandLine="*privileges::debug*" OR CommandLine="*sekurlsa::*" OR CommandLine="*kerberos::*" OR CommandLine="*crypto::*" OR CommandLine="*lsadump::*" OR CommandLine="*process::*"

Scenario: Data Collection and Compression with WinRAR

The following scenario showcases what Sysmon would record if a threat actor used WinRAR's "rar.exe" (renamed to "party.jpg") to compress data. WinRAR's "rar.exe" is a favored utility by threat actors since it is a small stand alone binary. The command below can be used to compress the folder "paystubs" into the password protected WinRAR archive named "data.tmp".

Command

- party.jpg a -hp123qwe data.tmp paystubs

Sysmon Events on Local System

Sysmon recorded the event in Figure 25 upon the execution of WinRAR with the above command. This Sysmon event is rich with data that we can use for several searches. First, we can search for all instances where "CommandLine" contains " -hp" to find all instances where WinRAR is being used to encrypt its content and header information regardless of the file name of WinRAR. Additionally, we can search for " a " (note the two spaces in front of the "a") to find instances where WinRAR is being used to add files to an archive regardless of the file name of WinRAR. Lastly, since threat actors will often rename their tools to hide what they are, we could search for all instances of WinRAR's imphash where the "Image" does not contain "rar.exe" to find all instances of renamed "rar.exe".

```
Process Create:
UtcTime: 2016-12-04 21:30:56.771
ProcessGuid: {efbfc260-8b10-5844-0000-0010665f9019}
ProcessId: 1144
Image: C:\PerfLogs\party.jpg
CommandLine: party.jpg  a -hp123qwe data.tmp paystubs
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-1a43-5843-0000-0020963bbd18}
LogonId: 0x18BD3B96
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=5DC9AAF3B27B54CD7981F4D28187CBC7,IMPHASH=4397DAB9E39CC23F35F79B293DDAD7
80
ParentProcessGuid: {efbfc260-8782-5844-0000-00106b438119}
ParentProcessId: 15976
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "cmd.exe" /s /k pushd "C:\PerfLogs"
```

Figure 25: Sysmon event generated upon execution of the WinRAR command on the local system

Splunk Searches

The Splunk searches below can be used to identify WinRAR usage in an environment.

- index=sysmon CommandLine="* -hp*"
- index=sysmon Hashes="*IMPHASH=4397DAB9E39CC23F35F79B293DDAD780*"
- index=sysmon Hashes="*IMPHASH=4397DAB9E39CC23F35F79B293DDAD780*" NOT Image="*rar.exe*"
- index=sysmon CommandLine="* a *"

Scenario: Data Collection and Compression with 7-Zip

This scenario showcases the use of the data compression utility 7-Zip's "7z.exe" (renamed to "header.png") which is another tool favored by threat actors since it is a stand alone binary. The command below can be used to compress the folder "paystubs" into the 7z archive "data.7z".

Command

- header.png a -p123qwe data.7z paystubs

Sysmon Events on Local System

The Sysmon event in Figure 26 was created after the above command was executed. From a search perspective, there are three searches we could do. First, we could search for " a " (note the two spaces before "a") to find instances of 7-Zip being used to add files to an archive. Second, we could search for just the IMPHASH of 7-Zip to find instances of 7-Zip regardless of its file name. Last, since threat actors will often rename their tools to hide what they are, we could search for all instances of the 7-Zip IMPHASH and the "Image" not containing "7z.exe".

```
Process Create:
UtcTime: 2016-12-04 21:41:25.520
ProcessGuid: {efbfc260-8d85-5844-0000-00106f7f9419}
ProcessId: 3380
Image: C:\PerfLogs\header.png
CommandLine: header.png  a -r -p123qwe data.7z paystubs
CurrentDirectory: C:\PerfLogs\
User: CORP\Alec
LogonGuid: {efbfc260-1a43-5843-0000-0020963bbd18}
LogonId: 0x18BD3B96
TerminalSessionId: 2
IntegrityLevel: Medium
Hashes: MD5=3E797119E0FD64297CB82794B8D68EDD,IMPHASH=06CCDA30750899D24EC1383D46A36E
65
ParentProcessGuid: {efbfc260-8782-5844-0000-00106b438119}
ParentProcessId: 15976
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "cmd.exe" /s /k pushd "C:\PerfLogs"
```

Figure 26: Sysmon event generated upon execution of the 7-Zip command on the local system

Splunk Searches

The Splunk searches before would find instances of 7-Zip usage in an environment.

- index=sysmon Hashes="*IMPHASH=06CCDA30750899D24EC1383D46A36E65*"
- index=sysmon Hashes="*IMPHASH=06CCDA30750899D24EC1383D46A36E65*" NOT Image="*7z.exe*"
- index=sysmon CommandLine="* a *"

Conclusions

Sysmon is a very powerful and free tool that security teams can leverage for detection and response activities. While this paper does not cover all the possible commands and methodologies you may encounter, it is intended to provide a baseline for using Sysmon with Splunk. As with any detection techniques, we encourage everyone to do their own testing to see what Sysmon records during simulated attacks. We also encourage everyone to install Sysmon in their environment, regardless of whether Splunk or another log aggregator or SIEM is present. The recorded information is invaluable in responding to incidents as it provides a more complete picture of what happened during the incident.

Appendix A: Splunk Searches

This appendix contains an aggregate of Splunk searches for easy referencing.

Reconnaissance

“net” commands

- index=sysmon Image="*\\net.exe" (CommandLine="*net group*" OR CommandLine="*net localgroup*") | stats count by Computer,CommandLine
- index=sysmon Image="*\\net.exe" CommandLine="*net user" | stats count by Computer,CommandLine
- index=sysmon Image="*\\net.exe" CommandLine="*net view" | stats count by Computer,CommandLine

Credential Harvesting

Windows Credential Editor

- index=sysmon ParentImage="*\\services.exe" | regex CommandLine="\\\\[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12}\.exe -S$"
- index=sysmon Image="*\\cmd.exe" CommandLine="* -w > *"
- index=sysmon Hashes="*IMPHASH=8AB93B061287C79F3088C5BC7E7D97ED*"

Mimikatz

- index=sysmon CommandLine="*privileges::debug*" OR CommandLine="*sekurlsa::*" OR CommandLine="*kerberos::*" OR CommandLine="*crypto::*" OR CommandLine="*lsadump::*" OR CommandLine="*process::*"
- index=sysmon Hashes="*IMPHASH=3ED0D1F8A6FB0E255E1ED340B1A3EAAB*"

pwdump

- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*"

gsecdump

- index=sysmon Image="*\\g64-*" | regex Image="\\\\g64-[a-z0-9]{3}\.exe$"

Lateral Movement

WMI

- index=sysmon Image="*\\WMIC.exe" CommandLine="*process call create*"
- index=sysmon ParentImage="*\\WmiPrvSE.exe" | stats count by Computer,CommandLine
- index=sysmon Image="*\\WMIC.exe" CommandLine="*/NODE:*"

PowerShell

- index=sysmon Image="*powershell.exe" CommandLine="*Invoke-Command*"
- index=sysmon ParentImage="*wsmprovhost.exe" | stats count by Computer,CommandLine
- index=sysmon Image="*powershell.exe" CommandLine="*-EncodedCommand*"
- index=sysmon Image="*powershell.exe" | stats count by Computer,CommandLine

PsExec

- index=sysmon ParentImage="*\\PSEXESVC.EXE" | stats count by Image,Hashes
- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*"
- index=sysmon Hashes="*IMPHASH=B18A1401FF8F444056D29450FBC0A6CE*" NOT Image="*PsExec.exe"

Scheduled Tasks

- index=sysmon ParentImage="*\\taskeng.exe" | stats count by Computer,CommandLine
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Create*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Run*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Delete*" CommandLine="*/s *"
- index=sysmon Image="*\\schtasks.exe" CommandLine="*/Change*" CommandLine="*/s *"
- index=sysmon ParentImage="*\\at.exe" | stats count by Computer,CommandLine

Data Collection and Compression

WinRAR

- index=sysmon CommandLine="* -hp*"
- index=sysmon Hashes="*IMPHASH=4397DAB9E39CC23F35F79B293DDAD780*"
- index=sysmon Hashes="*IMPHASH=4397DAB9E39CC23F35F79B293DDAD780*" NOT Image="*rar.exe*"
- index=sysmon CommandLine="* a *"

7-Zip

- index=sysmon Hashes="*IMPHASH=06CCDA30750899D24EC1383D46A36E65*"
- index=sysmon Hashes="*IMPHASH=06CCDA30750899D24EC1383D46A36E65*" NOT Image="*7z.exe*"
- index=sysmon CommandLine="* a *"

Miscellaneous

Stacking Command Line Working Directories

- index=sysmon ParentImage="*\\cmd.exe" | stats count by CurrentDirectory

Stacking processes that don't have an executable extension

- index=sysmon EventCode="1" NOT Image="*.exe" NOT Image="System" NOT Image="<unknown process>" | stats count by Computer,CommandLine

Stacking processes with a single character file name

- index=sysmon | regex Image="\\\\.\....$" | stats count by Computer,CommandLine

Stacking binary network connections

- index=sysmon EventCode=3 | iplocation dest_ip | stats count by process,dest_ip,Country

Statistical search of processes opening network connections

- index=sysmon EventCode=3 | streamstats window=1 global=f current=f last(_time) as next_ts by Computer Image DestinationIp | eval tm_to_next=next_ts-_time | stats values(direction) as direction count avg(tm_to_next) as avgTimeToNext stdev(tm_to_next) as stdDevTimeToNext by Computer Image DestinationIp | where stdDevTimeToNext < 1 AND count > 5 AND avgTimeToNext > 0| iplocation DestinationIp | lookup dnslookup clientip as DestinationIp | fields - lat lon | sort stdDevTimeToNext

Contact Us to Learn More About How The Crypsis Group Can Help Your Organization

703.570.4103 | info@crypsisgroup.com