Splunk Use Cases

24 November 2021 - **72 mins read time**
Tags: Splunk

1- Windows Audit Log Tampering

Check for any tampering done to Windows audit logs.

```
index=__your_sysmon_index__ (sourcetype=wineventlog AND (EventCode=1102 OR EventCode=1100)) OR
(sourcetype=wineventlog AND EventCode=104)
| stats count by _time EventCode Message sourcetype host
```

2- Finding Large Web Uploads

Find large file uploads that could point to data exfiltration in your network.

```
index=__your_sysmon_index__ sourcetype=websense*
| where bytes_out > 35000000
| table _time src_ip bytes* uri
```

3- Detecting Recurring Malware on Host

Using anti-virus logs to detect if malware is recurring on a host after being removed.

```
index=__your_sysmon_index__ sourcetype=symantec:*
| stats count range(_time) as TimeRange by Risk_Name, Computer_Name
| where TimeRange>1800
| eval TimeRange_In_Hours = round(TimeRange/3600,2), TimeRange_In_Days = round(TimeRange/3600/24,2)
```

4-Detecting Brute Force Attacks

A brute-force attack consists of a multiple login attempts using many passwords by an unauthorized user/attacker with the hope of eventually guessing the correct password.

```
index=__your_sysmon_index__ sourcetype=winxsecurity user=* user!""
| stats count(eval(action="success")) as successes count(eval(action="failure")) as failures by user, ComputerName
| where successes>0 AND failures>100
```

Windows

```
index=windows source="WinEventLog:Security" EventCode=4625
| bin _time span=5m
| stats count by _time,user,host,src,action
| where count >= 5
```

Linux

```
index=linux source="/var/log/auth.log" "Failed password"
| bin _time span=5m
| stats count by _time,user,host,src,action
| where count >= 5
```

5- Detecting Unencrypted Web Communications

Find unencrypted web communications that could lead to a data breach.

```
index=__your_sysmon_index__ sourcetype=firewall_data dest_port!=443 app=workday*
| table _time user app bytes* src_ip dest_ip dest_port
```

6- Identifying Web Users By Country

Use IPs in your data to report and visualize user locations.

```
index=web sourcetype=access_combined
| iplocation clientip
| geostats dc(clientip) by Country
```

7- Identifying Slow Web Content

A slow loading web site can not only frustrate users, but can also hurt search rankings.

```
index=web sourcetype=ms:iis:auto OR sourcetype=apache: access
| stats avg(response_time) as art by uri_path
| eval "Average Response Time" = round(art,2)
| sort -"Average Response Time"
| table uri_path, "Average Response Time"
```

8- Finding New Local Admin Accounts

Often an attack will include the creation of a new user, followed by permissions being elevated to an admin level.

```
index=win_servers sourcetype=windows:security EventCode=4720 OR (EventCode=4732 Administrators)
| transaction Security_ID maxspan=180m
| search EventCode=4720 EventCode=4732
| table _time, EventCode, Recurity_ID, SamAccountName
```

9- Finding Interactive Logins From Service Accounts

Most service accounts should never interactively log into servers.

```
index=systems sourcetype=audit_logs user=svc_*
| stats earliest(_time) as earliest latest(_time) as latest by user, dest
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)
| convert ctime(earliest) ctime(Latest)
```

10- Log Volume Trending

Visualizing the number of events being logged by an application can provide a simple, yet powerful indicator of the state of your application, or changes in the behavior of your code or environment.

```
|tstats prestats=t count WHERE index=apps by host _time span=1m
|timechart partial=f span=1m count by host limit=0
```

11- Basic TOR Traffic Detection

Use firewall data to find TOR traffic on your network.

```
index=network sourcetype=firewall_data app=tor src_ip=*
| table _time src_ip src_port dest_ip dest_port bytes app
```

12- Measuring Storage I/O Latency

Quickly find I/O bottlenecks across your systems.

```
index=main sourcetype=iostat
| timechart avg(latency) by host
```

13- Measuring Storage Speed I/O Utilization by Host

It simple to track disk I/O, helping you quickly discover storage issues on your servers.

```
index=main sourcetype=iostat
| eval hostdevice=host+":"+Device
| timechart avg(total_ops) by hostdevice
```

14- Measuring Memory Utilization by Host

It's easy to track memory utilization of your systems using Splunk Enterprise.

```
index=main sourcetype=vmstat
| stats max(memUsedPct) as memused by host
| where memused>80
```

15- Rogue DNS detection

Look for DNS requests that are not destined for the dedicated DNS server.

```
index=security sourcetype=cp_log src_ip!=192.168.14.10 dest_ip!=192.168.14.10
protocol=53 action!=Drop
| where dest_ip="192.168.0.0/16" AND src_ip="192.168.0.0/16"
| stats count, values(dest_ip) by src_ip
```

16- Suspicious PowerShell Commands

Look for logs with commands that try to download external scripts/content or bypass PowerShell.

```
index=windows source="WinEventLog:Microsoft-Windows-PowerShell/Operational" EventCode=4104
AND ((ScriptBlockText=*-noni* *iex* *New-Object*) OR (ScriptBlockText=*-ep* *bypass* *-Enc*) OR
(ScriptBlockText=*powershell* *reg* *add*
*HKCU\\software\\microsoft\\windows\\currentversion\\run*) OR (ScriptBlockText=*bypass* *-
noprofile* *-windowstyle* *hidden* *new-object* *system.net.webclient* *.download*) OR
(ScriptBlockText=*iex* *New-Object* *Net.WebClient* *.Download*))
| table Computer, ScriptBlockText, UserID
```

17- Windows audit log cleared

Look for security logs filtered with EventCode 1102.

```
index=windows source="WinEventLog:Security" EventCode=1102
| table _time, host, signature, user
```

18- Detecting Network and Port Scanning

Look for distinct count of destination ports within a short span of time.

```
| from datamodel:"Network_Traffic"."All_Traffic"
| stats dc(dest_port) as dc_dest_port by src, dest
| where dc_dest_port > 10
```

OR

```
index=__your_sysmon_index__ sourcetype=firewall*
| stats dc(dest_port) as num_dest_port dc(dest_ip) as num_dest_ip by src_ip
| where num_dest_port >500 OR num_dest_ip >500
```

19- Unusual Access

Look for count of multiple failed login attempts where successful login is true.

```
| from datamodel:"Authentication"."Authentication"
| where like(app,"ssh")
| stats list(action) as Attempts, count(eval(match(action,"failure"))) as Failed,
count(eval(match(action,"success"))) as Success by user,src,dest,app
| where mvcount(Attempts)>=6 AND Success>0 AND Failed>=5
```

20- Malware Attack

Look for infection count of malware attack.

```
| from datamodel:"Malware"."Malware_Attacks"
| stats dc("signature") as "infection_count" by "dest"
| where 'infection_count'>1
```

21-Attempt To Add Certificate To Untrusted Store

Adversaries may add their own root certificate to the certificate store, to cause the web browser to trust that certificate and not display a security warning when it encounters the previously unseen certificate. This action may be the precursor to malicious activity.

```
| tstats count min(_time) as firstTime values(Processes.process) as process max(_time) as lastTime from
datamodel=Endpoint.Processes where Processes.process_name=*certutil* (Processes.process=*-addstore*) by
Processes.parent_process Processes.process_name Processes.user
```

22- **Batch File Write to System32**

While batch files are not inherently malicious, it is uncommon to see them created after OS installation, especially in the Windows directory. This analytic looks for the suspicious activity of a batch file being created within the C:\Windows\System32 directory tree. There will be only occasional false positives due to administrator actions.

```
| tstats count min(_time) as firstTime max(_time) as lastTime values(Filesystem.dest) as dest
values(Filesystem.file_name) as file_name values(Filesystem.user) as user from datamodel=Endpoint.Filesystem by
Filesystem.file_path   | rex field=file_name "(?<file_extension>\.[^\.]+)$" | search file_path=*system32* AND
file_extension=.bat
```

23- BCDEdit Failure Recovery Modification

This search looks for flags passed to bcdedit.exe modifications to the built-in Windows error recovery boot configurations. This is typically used by ransomware to prevent recovery.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name = bcdedit.exe Processes.process="*recoveryenabled*" (Processes.process="* no*") by
Processes.process_name Processes.process Processes.parent_process_name Processes.dest Processes.user
```

24- BITS Job Persistence

The following query identifies Microsoft Background Intelligent Transfer Service utility `bitsadmin.exe` scheduling a BITS job to persist on an endpoint. The query identifies the parameters used to create, resume or add a file to a BITS job. Typically seen combined in a oneliner or ran in sequence. If identified, review the BITS job created and capture any files written to disk. It is possible for BITS to be used to upload files and this may require further network data analysis to identify. You can use `bitsadmin /list /verbose` to list out the jobs during investigation.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name=bitsadmin.exe Processes.process IN (*create*, *addfile*, *setnotifyflags*,
*setnotifycmdline*, *setminretrydelay*, *setcustomheaders*, *resume* ) by Processes.dest Processes.user
Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.parent_process_id
```

25- BITSAdmin Download File

The following query identifies Microsoft Background Intelligent Transfer Service utility `bitsadmin.exe` using the `transfer` parameter to download a remote object. In addition, look for `download` or `upload` on the command-line, the switches are not required to perform a transfer. Capture any files downloaded. Review the reputation of the

IP or domain used. Typically once executed, a follow on command will be used to execute the dropped file. Note that the network connection or file modification events related will not spawn or create from `bitsadmin.exe`, but the artifacts will appear in a parallel process of `svchost.exe` with a command-line similar to `svchost.exe -k netsvcs -s BITS`. It's important to review all parallel and child processes to capture any behaviors and artifacts. In some suspicious and malicious instances, BITS jobs will be created. You can use `bitsadmin /list /verbose` to list out the jobs during investigation.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name=bitsadmin.exe Processes.process=*transfer* by Processes.dest Processes.user
Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.parent_process_id
```

26- CertUtil Download With URLCache and Split Arguments

Certutil.exe may download a file from a remote destination using `urlcache`. This behavior does require a URL to be passed on the command-line. In addition, `f` (force) and `split` (Split embedded ASN.1 elements, and save to files) will be used. It is not entirely common for `certutil.exe` to contact public IP space. However, it is uncommon for `certutil.exe` to write files to world writeable paths.\ During triage, capture any files on disk and review. Review the reputation of the remote IP or domain in question.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name=certutil.exe Processes.process=*urlcache* Processes.process=*split* by Processes.dest
Processes.user Processes.parent_process Processes.process_name Processes.process Processes.process_id
Processes.parent_process_id
```

27- CertUtil Download With VerifyCtl and Split Arguments

Certutil.exe may download a file from a remote destination using `VerifyCtl`. This behavior does require a URL to be passed on the command-line. In addition, `f` (force) and `split` (Split embedded ASN.1 elements, and save to files) will be used. It is not entirely common for `certutil.exe` to contact public IP space. \ During triage, capture any files on disk and review. Review the reputation of the remote IP or domain in question. Using `VerifyCtl`, the file will either be written to the current working directory or `%APPDATA%\..\LocalLow\Microsoft\CryptnetUrlCache\Content\<hash>`.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name=certutil.exe Processes.process=*verifyctl* Processes.process=*split* by Processes.dest
Processes.user Processes.parent_process Processes.process_name Processes.process Processes.process_id
Processes.parent_process_id
```

28- Certutil exe certificate extraction

This search looks for arguments to certutil.exe indicating the manipulation or extraction of Certificate. This certificate can then be used to sign new authentication tokens specially inside Federated environments such as Windows ADFS.

```
| tstats count min(_time) as firstTime values(Processes.process) as process max(_time) as lastTime from
datamodel=Endpoint.Processes where Processes.process_name=certutil.exe Processes.process = "* -exportPFX *" by
Processes.parent_process Processes.process_name Processes.process Processes.user
```

29- CertUtil With Decode Argument

CertUtil.exe may be used to `encode` and `decode` a file, including PE and script code. Encoding will convert a file to base64 with `----BEGIN CERTIFICATE-----` and `----END CERTIFICATE-----` tags. Malicious usage will include decoding a encoded file that was downloaded. Once decoded, it will be loaded by a parallel process. Note that there are two additional command switches that may be used - `encodehex` and `decodehex`. Similarly, the file will be encoded in HEX and later decoded for further execution. During triage, identify the source of the file being decoded. Review its contents or execution behavior for further analysis.

```
| tstats count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where
Processes.process_name=certutil.exe Processes.process=*decode* by Processes.dest Processes.user
Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.parent_process_id
```

30- Create local admin accounts using net exe

This search looks for the creation of local administrator accounts using net.exe.

```
| tstats count values(Processes.user) as user values(Processes.parent_process) as parent_process min(_time) as
firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where (Processes.process_name=net.exe OR
Processes.process_name=net1.exe) AND (Processes.process=*localgroup* OR Processes.process=*/add* OR
Processes.process=*user*) by Processes.process Processes.process_name Processes.dest
|`create_local_admin_accounts_using_net_exe_filter`
```

31- Create Remote Thread into LSASS

Actors may create a remote thread into the LSASS service as part of a workflow to dump credentials.

```
`sysmon` EventID=8 TargetImage=*lsass.exe | stats count min(_time) as firstTime max(_time) as lastTime by Computer,
EventCode, TargetImage, TargetProcessId | rename Computer as dest
```

32- Create Service In Suspicious File Path

This detection is to identify a creation of “user mode service” where the service file path is located in non-common service folder in windows.

```
`wineventlog_system` EventCode=7045  Service_File_Name = "*\.exe" NOT (Service_File_Name IN ("C:\\Windows\\*",
"C:\\Program File*", "C:\\Programdata\\*", "%systemroot%\\*")) Service_Type = "user mode service" | stats count
min(_time) as firstTime max(_time) as lastTime by EventCode Service_File_Name Service_Name Service_Start_Type
Service_Type
```

33- Common Windows Process Masquerading

“Masquerading occurs when the name or location of an object, legitimate or malicious, is manipulated or abused for the sake of evading defenses and observation. This may include manipulating file metadata, tricking users into misidentifying the file type, and giving legitimate task or service names.”

Malware authors often use this technique to hide malicious executables behind legitimate Windows executable names (e.g. `lsass.exe` , `svchost.exe` , etc).

```
index=__your_sysmon_index__ source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" AND (
(process_name=svchost.exe AND NOT (process_path="C:\\Windows\\System32\\svchost.exe" OR
process_path="C:\\Windows\\SysWow64\\svchost.exe"))
OR (process_name=smss.exe AND NOT process_path="C:\\Windows\\System32\\smss.exe")
OR (process_name=wininit.exe AND NOT process_path="C:\\Windows\\System32\\wininit.exe")
OR (process_name=taskhost.exe AND NOT process_path="C:\\Windows\\System32\\taskhost.exe")
OR (process_name=lasass.exe AND NOT process_path="C:\\Windows\\System32\\lsass.exe")
OR (process_name=winlogon.exe AND NOT process_path="C:\\Windows\\System32\\winlogon.exe")
OR (process_name=csrss.exe AND NOT process_path="C:\\Windows\\System32\\csrss.exe")
OR (process_name=services.exe AND NOT process_path="C:\\Windows\\System32\\services.exe")
OR (process_name=lsm.exe AND NOT process_path="C:\\Windows\\System32\\lsm.exe")
OR (process_name=explorer.exe AND NOT process_path="C:\\Windows\\explorer.exe")
)
```

34- Unusual Child Process spawned using DDE exploit

Adversaries may use Windows Dynamic Data Exchange (DDE) to execute arbitrary commands. DDE is a client-server protocol for one-time and/or continuous inter-process communication (IPC) between applications. Once a link is established, applications can autonomously exchange transactions consisting of strings, warm data links (notifications when a data item changes), hot data links (duplications of changes to a data item), and requests for command execution.

```
index = __your_sysmon__index__ (ParentImage="*excel.exe" OR ParentImage="*word.exe" OR ParentImage="*outlook.exe")
Image="*.exe"
```

35- Detecting Tampering of Windows Defender Command Prompt

In an attempt to avoid detection after compromising a machine, threat actors often try to disable Windows Defender. This is often done using “sc” [service control], a legitimate tool provided by Microsoft for managing services. This action interferes with event detection and may lead to a security event going undetected, thereby potentially leading to further compromise of the network.

```
index= __your_sysmon__index__ EventCode=1 Image = "C:\\Windows\\System32\\sc.exe"  | regex CommandLine="^sc\s*
(config|stop|query)\sWinDefend$"
```

36- Disable UAC

Threat actors often, after compromising a machine, try to disable User Access Control (UAC) to escalate privileges. This is often done by changing the registry key for system policies using “reg.exe”, a legitimate tool provided by Microsoft for modifying the registry via command prompt or scripts. This action interferes with UAC and may enable a threat actor to escalate privileges on the compromised system, thereby allowing further exploitation of the system.

```
sourcetype = __your_sysmon_index__ ParentImage = "C:\\Windows\\System32\\cmd.exe" | where
like(CommandLine,"reg.exe%HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System%REG_DWORD /d 0%")
```

37- Identifying Port Scanning Activity

After compromising an initial machine, adversaries commonly attempt to laterally move across the network. The first step to attempt the lateral movement often involves conducting host identification, port and service scans on the internal network via the compromised machine using tools such as Nmap, Cobalt Strike, etc.

```
sourcetype='firewall_logs' dest_ip = 'internal_subnet' | stats dc(dest_port) as pcount by src_ip | where pcount >5
```

38- Unusually Long Command Line Strings

Often, after a threat actor gains access to a system, they will attempt to run some kind of malware to further infect the victim machine. These malware often have long command line strings, which could be a possible indicator of attack. Here, we use sysmon and Splunk to first find the average command string length and search for command strings that stretch over multiple lines, thus identifying anomalies and possibly malicious commands.

```
index=__your_sysmon_index__ sourcetype="xmlwineventlog" EventCode=4688  |eval cmd_len=len(CommandLine) | eventstats
avg(cmd_len) as avg by host| stats max(cmd_len) as maxlen, values(avg) as avgperhost by host, CommandLine | where
maxlen > 10*avgperhost
```

39- Clearing Windows Logs with Wevtutil

In an attempt to clear traces after compromising a machine, threat actors often try to clear Windows Event logs. This is often done using “wevtutil”, a legitimate tool provided by Microsoft. This action interferes with event collection and notification, and may lead to a security event going undetected, thereby potentially leading to further compromise of the network.

```
index=__your_sysmon_index__ sourcetype= __your__windows__sysmon__sourcetype EventCode=1 Image=*wevtutil*
CommandLine=*cl* (CommandLine=*System* OR CommandLine=*Security* OR CommandLine=*Setup* OR
CommandLine=*Application*)
```

40- Unusual Child Process for Spoolsv.Exe or Connhost.Exe

After gaining initial access to a system, threat actors attempt to escalate privileges as they may be operating within a lower privileged process which does not allow them to access protected information or carry out tasks which require higher permissions. A common way of escalating privileges in a system is by externally invoking and exploiting spoolsv or connhost executables, both of which are legitimate Windows applications. This query searches for an invocation of either of these executables by a user, thus alerting us of any potentially malicious activity.

```
(index=__your_sysmon_index__ EventCode=1) (Image=C:\\Windows\\System32\\spoolsv.exe* OR
Image=C:\\Windows\\System32\\conhost.exe) ParentImage = "C:\\Windows\\System32\\cmd.exe"
```

41- Detecting Shadow Copy Deletion via Vssadmin.exe

After compromising a network of systems, threat actors often try to delete Shadow Copy in an attempt to prevent administrators from restoring the systems to versions present before the attack. This is often done via vssadmin, a legitimate Windows tool to interact with shadow copies. This non-detection of this technique, which is often employed by ransomware strains such as “Olympic Destroyer”, may lead to a failure in recovering systems after an attack.

```
index=__your_win_event_log_index__ EventType=4688 CommandLine:"delete" OriginalFileName:"VSSADMIN.EXE"
```

42- Webshell-Indicative Process Tree

A web shell is a web script placed on an openly accessible web server to allow an adversary to use the server as a gatway in a network. As the shell operates, commands will be issued from within the web application into the broader server operating system. This analytic looks for host enumeration executables initiated by any web service that would not normally be executed within that environment.

```
index=__your_sysmon_index__ (ParentImage="C:\\Windows\\System32\\services.exe"
Image="C:\\Windows\\System32\\cmd.exe" (CommandLine="*echo*" AND CommandLine="*\\pipe\\*"))
OR (Image="C:\\Windows\\System32\\rundll32.exe" CommandLine="*,a /p:*")
```

43- Get System Elevation

Cyber actors frequently escalate to the SYSTEM account after gaining entry to a Windows host, to enable them to carry out various attacks more effectively. Tools such as Meterpreter, Cobalt Strike, and Empire carry out automated steps to “Get System”, which is the same as switching over to the System user account. Most of these tools utilize multiple techniques to try and attain SYSTEM: in the first technique, they create a named pipe and connects an instance of cmd.exe to it, which allows them to impersonate the security context of cmd.exe, which is SYSTEM. In the second technique, a malicious DLL is injected into a process that is running as SYSTEM; the injected DLL steals the SYSTEM token and applies it where necessary to escalate privileges. This analytic looks for both of these techniques.

```
index=__your_sysmon_index__ (ParentImage="C:\\Windows\\System32\\services.exe"
Image="C:\\Windows\\System32\\cmd.exe" (CommandLine="*echo*" AND CommandLine="*\\pipe\\*"))
OR (Image="C:\\Windows\\System32\\rundll32.exe" CommandLine="*,a /p:*")
```

```
index=__your_sysmon_index__ (Image="C:\\Windows\\System32\\cmd.exe" OR CommandLine="*%COMSPEC%*")
(CommandLine="*echo*" AND CommandLine="*\pipe\*")
```

44- Boot or Logon Initialization Scripts

Adversaries may schedule software to run whenever a user logs into the system; this is done to establish persistence and sometimes for lateral movement. This trigger is established through the registry key HKEY_CURRENT_USER\Environment*UserInitMprLogonScript*. This signature looks edits to existing keys or creation of new keys in that path. Users purposefully adding benign scripts to this path will result in false positives; that case is rare, however. There are other ways of running a script at startup or login that are not covered in this signature. Note that this signature overlaps with the Windows Sysinternals Autoruns tool, which would also show changes to this registry path.

```
(index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\System32\\reg.exe"
CommandLine="*add*\\Environment*UserInitMprLogonScript") OR (index=__your_sysmon_index__ (EventCode=12 OR
EventCode=14 OR EventCode=13) TargetObject="*\\Environment*UserInitMprLogonScript")
```

45- Local Network Sniffing

Adversaries may use a variety of tools to gain visibility on the current status of things on the network: which processes are listening on which ports, which services are running on other hosts, etc. This analytic looks for the names of the most common network sniffing tools. While this may be noisy on networks where sysadmins are using any of these tools on a regular basis, in most networks their use is noteworthy.

```
(index=__your_sysmon_index__ EventCode=1) (Image="*tshark.exe" OR Image="*windump.exe" OR (Image="*logman.exe" AND
ParentImage!="?" AND ParentImage!="C:\\Program Files\\Windows Event
Reporting\\Core\\EventReporting.AgentService.exe") OR Image="*tcpdump.exe" OR Image="*wprui.exe" OR
Image="*wpr.exe")
```

46- DLL Injection with Mavinject

Injecting a malicious DLL into a process is a common adversary TTP. Although the ways of doing this are numerous, mavinject.exe is a commonly used tool for doing so because it roles up many of the necessary steps into one, and is available within Windows. Attackers may rename the executable, so we also use the common argument

"INJECTRUNNING" as a related signature here. Whitelisting certain applications may be necessary to reduce noise for this analytic.

```
(index=__your_sysmon_index__ EventCode=1) (Image="C:\\Windows\\SysWOW64\\mavinject.exe" OR
Image="C:\\Windows\\System32\\mavinject.exe" OR CommandLine="*\INJECTRUNNING*")
```

47- Processes Started From Irregular Parent

Adversaries may start legitimate processes and then use their memory space to run malicious code. This analytic looks for common Windows processes that have been abused this way in the past; when the processes are started for this purpose they may not have the standard parent that we would expect. This list is not exhaustive, and it is possible for cyber actors to avoid this discepency. These signatures only work if Sysmon reports the parent process, which may not always be the case if the parent dies before sysmon processes the event.

```
(index=__your_sysmon_index__ EventCode=1) AND ParentImage!="?" AND ParentImage!="C:\\Program
Files\\SplunkUniversalForwarder\\bin\\splunk-regmon.exe" AND ParentImage!="C:\\Program
Files\\SplunkUniversalForwarder\\bin\\splunk-powershell.exe" AND
((Image="C:\\Windows\System32\\smss.exe" AND (ParentImage!="C:\\Windows\\System32\\smss.exe" AND
ParentImage!="System")) OR
(Image="C:\\Windows\\System32\\csrss.exe" AND (ParentImage!="C:\\Windows\\System32\\smss.exe" AND
ParentImage!="C:\\Windows\\System32\\svchost.exe")) OR
(Image="C:\\Windows\\System32\\wininit.exe" AND ParentImage!="C:\\Windows\\System32\\smss.exe") OR
(Image="C:\\Windows\\System32\\winlogon.exe" AND ParentImage!="C:\\Windows\\System32\\smss.exe") OR
(Image="C:\\Windows\\System32\\lsass.exe" and ParentImage!="C:\\Windows\\System32\\wininit.exe") OR
(Image="C:\\Windows\\System32\\LogonUI.exe" AND (ParentImage!="C:\\Windows\\System32\\winlogon.exe" AND
ParentImage!="C:\\Windows\\System32\\wininit.exe")) OR
(Image="C:\\Windows\\System32\\services.exe" AND ParentImage!="C:\\Windows\\System32\\wininit.exe") OR
(Image="C:\\Windows\\System32\\spoolsv.exe" AND ParentImage!="C:\\Windows\\System32\\services.exe") OR
(Image="C:\\Windows\\System32\\taskhost.exe" AND (ParentImage!="C:\\Windows\\System32\\services.exe" AND
ParentImage!="C:\\Windows\\System32\\svchost.exe")) OR
(Image="C:\\Windows\\System32\\taskhostw.exe" AND (ParentImage!="C:\\Windows\\System32\\services.exe" AND
ParentImage!="C:\\Windows\\System32\\svchost.exe")) OR
(Image="C:\\Windows\System32\\userinit.exe" AND (ParentImage!="C:\\Windows\\System32\\dwm.exe" AND
ParentImage!="C:\\Windows\\System32\\winlogon.exe")))
```

48- Clear Powershell Console Command History

Adversaries may attempt to conceal their tracks by deleting the history of commands run within the Powershell console, or turning off history saving to begin with. This analytic looks for several commands that would do this. This does not capture the event if it is done within the console itself; only commandline-based commands are detected. Note that the command to remove the history file directly may very a bit if the history file is not saved in the default path on a particular system.

```
(index=__your_sysmon_index__ EventCode=1) (CommandLine="*rm (Get-PSReadlineOption).HistorySavePath*" OR
CommandLine="*del (Get-PSReadlineOption).HistorySavePath*" OR CommandLine="*Set-PSReadlineOption –HistorySaveStyle
SaveNothing*" OR CommandLine="*Remove-Item (Get-PSReadlineOption).HistorySavePath*" OR
CommandLine="del*Microsoft\\Windows\\Powershell\\PSReadline\\ConsoleHost_history.txt")
```

49- Local Permission Group Discovery

Cyber actors frequently enumerate local or domain permissions groups. The net utility is usually used for this purpose. This analytic looks for any instances of net.exe, which is not normally used for benign purposes, although system administrator actions may trigger false positives.

```
(index=__your_sysmon_index__ EventCode=1) Image="C:\\Windows\\System32\\net.exe" AND (CommandLine="* user*" OR
CommandLine="* group*" OR CommandLine="* localgroup*" OR CommandLine="*get-localgroup*" OR CommandLine="*get-
ADPrincipalGroupMembership*")
```

50- Network Share Connection Removal

Adversaries may use network shares to exfliltrate date; they will then remove the shares to cover their tracks. This analytic looks for the removal of network shares via commandline, which is otherwise a rare event.

```
(index=__your_sysmon_index__ EventCode=1) ((Image="C:\\Windows\\System32\\net.exe" AND CommandLine="*delete*") OR
CommandLine="*Remove-SmbShare*" OR CommandLine="*Remove-FileShare*")
```

51- MSBuild and msxsl

Trusted developer utilities such as MSBuild may be leveraged to run malicious code with elevated privileges. This analytic looks for any instances of msbuild.exe, which will execute any C# code placed within a given XML document; and msxsl.exe, which processes xsl transformation specifications for XML files and will execute a variaty of scripting languages contained within the XSL file. Both of these executables are rarely used outside of Visual Studio.

```
(index=__your_sysmon_index__ EventCode=1) (Image="C:\\Program Files (x86)\\Microsoft Visual
Studio\\*\\bin\\MSBuild.exe" OR Image="C:\\Windows\\Microsoft.NET\\Framework*\\msbuild.exe" OR
Image="C:\\users\\*\\appdata\\roaming\\microsoft\\msxsl.exe") ParentImage!="*\\Microsoft Visual Studio*")
```

52- Compiled HTML Access

Adversaries may hide malicious code in .chm compiled HTML files. When these files are read, Windows uses the HTML help executable named hh.exe, which is the signature for this analytic.

```
(index=__your_sysmon_index__ EventCode=1) (Image="C:\\Windows\\syswow64\\hh.exe" OR
Image="C:\\Windows\\system32\\hh.exe")
```

53- CMSTP

CMSTP.exe is the Microsoft Connection Manager Profile Installer, which can be leveraged to setup listeners that will receive and install malware from remote sources in trusted fashion. When CMSTP.exe is seen in combination with an external connection, it is a good indication of this TTP.

```
(index=__your_sysmon_index__ EventCode=3) Image="C:\\Windows\\System32\\CMSTP.exe" | where
((!cidrmatch("10.0.0.0/8", SourceIp) AND !cidrmatch("192.168.0.0/16", SourceIp) AND !cidrmatch("172.16.0.0/12",
SourceIp))
```

54- Registry Edit from Screensaver

Adversaries may use screensaver files to run malicious code. This analytic triggers on suspicious edits to the screensaver registry keys, which dictate which .scr file the screensaver runs.

```
index=your_sysmon_index (EventCode=12 OR EventCode=13 OR EventCode=14)
TargetObject="*\\Software\\Policies\\Microsoft\\Windows\\Control Panel\\Desktop\\SCRNSAVE.EXE"
```

55- Scheduled Task - File Access

In order to gain persistence, privilege escalation, or remote execution, an adversary may use the Windows Task Scheduler to schedule a command to be run at a specified time, date, and even host. Task Scheduler stores tasks as files in two locations - C:\Windows\Tasks (legacy) or C:\Windows\System32\Tasks. Accordingly, this analytic looks for the creation of task files in these two locations.

```
index=__your_sysmon_index__ EventCode=11 Image!="C:\\WINDOWS\\system32\\svchost.exe"
(TargetFilename="C:\\Windows\\System32\\Tasks\\*" OR TargetFilename="C:\\Windows\\Tasks\\*")
```

56- Component Object Model Hijacking

Adversaries may establish persistence or escalate privileges by executing malicious content triggered by hijacked references to Component Object Model (COM) objects. This is typically done by replacing COM object registry entries under the HKEY_CURRENT_USER\Software\Classes\CLSID or HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID keys. Accordingly, this analytic looks for any changes under these keys.

```
index=__your_sysmon_index__ (EventCode=12 OR EventCode=13 OR EventCode=14)
TargetObject="*\\Software\\Classes\\CLSID\\*"
```

57- Indicator Blocking - Driver Unloaded

Adversaries may attempt to evade system defenses by unloading minifilter drivers used by host-based sensors such as Sysmon through the use of the fltmc command-line utility. Accordingly, this analytic looks for command-line invocations of this utility when used to unload minifilter drivers.

```
index=client EventCode=1 CommandLine="*unload*" (Image="C:\\Windows\\SysWOW64\\fltMC.exe" OR
Image="C:\\Windows\\System32\\fltMC.exe")
```

58- Credentials in Files & Registry

Adversaries may search the Windows Registry on compromised systems for insecurely stored credentials for credential access. This can be accomplished using the query functionality of the reg.exe system utility, by looking for keys and values that contain strings such as “password”. In addition, adversaries may use toolkits such as PowerSploit in order to dump credentials from various applications such as IIS.Accordingly, this analytic looks for invocations of reg.exe in this capacity as well as that of several powersploit modules with similar functionality.

```
((index=__your_sysmon_index__ EventCode=1) OR (index=__your_win_syslog_index__ EventCode=4688)) (CommandLine="*reg*
query HKLM /f password /t REG_SZ /s*" OR CommandLine="reg* query HKCU /f password /t REG_SZ /s" OR
CommandLine="*Get-UnattendedInstallFile*" OR CommandLine="*Get-Webconfig*" OR CommandLine="*Get-ApplicationHost*"
OR CommandLine="*Get-SiteListPassword*" OR CommandLine="*Get-CachedGPPPassword*" OR CommandLine="*Get-
RegistryAutoLogon*")
```

59- AppInit DLLs

Adversaries may establish persistence and/or elevate privileges by executing malicious content triggered by AppInit DLLs loaded into processes. Dynamic-link libraries (DLLs) that are specified in the AppInit_DLLs value in the Registry keys `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows` or `HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows` are loaded by user32.dll into every process that loads user32.dll. These values can be abused to obtain elevated privileges by causing a malicious DLL to be loaded and run in the context of separate processes. Accordingly, this analytic looks for modifications to these registry keys that may be indicative of this type of abuse.

```
index=__your_sysmon_index__ (EventCode=12 OR EventCode=13 OR EventCode=14)
(TargetObject="*\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\Appinit_Dlls\\*" OR
TargetObject="*\\SOFTWARE\\Wow6432Node\\Microsoft\\Windows NT\\CurrentVersion\\Windows\\Appinit_Dlls\\*")
```

60- NTFS Alternate Data Stream Execution - System Utilities

NTFS Alternate Data Streams (ADSs) may be used by adversaries as a means of evading security tools by storing malicious data or binaries in file attribute metadata. ADSs are also powerful because they can be directly executed by various Windows tools; accordingly, this analytic looks at common ways of executing ADSs using system utilities such as powershell.

NTFS ADS - PowerShell

```
index=__sysmon_index__ EventCode=1 Image=C:\\Windows\\*\\powershell.exe|regex CommandLine="Invoke-CimMethod\s+-
ClassName\s+Win32_Process\s+-MethodName\s+Create.*\b(\w+(\.\w+)?):(\w+(\.\w+)?)|-ep bypass\s+-\s+<.*\b(\w+
(\.\w+)?):(\w+(\.\w+)?)|-command.*Get-Content.*-Stream.*Set-Content.*start-process .*(\w+(\.\w+)?)"NTFS ADS - wmic
```

NTFS ADS - wmic

```
index=__sysmon_index__ EventCode=1 Image=C:\\Windows\\*\\wmic.exe | regex CommandLine="process call create.*\"(\w+
(\.\w+)?):(\w+(\.\w+)?)"
```

NTFS ADS - rundll32

```
index=__sysmon_index__  EventCode=1 Image=C:\\Windows\\*\\rundll32.exe | regex CommandLine="\"?(\w+(\.\w+)?):(\w+
(\.\w+)?)?\"?,\w+\|(advpack\.dll\|ieadvpack\.dll),RegisterOCX\s+(\w+\.\w+):(\w+(\.\w+)?)\|
(shdocvw\.dll\|ieframe\.dll),OpenURL.*(\w+\.\w+):(\w+(\.\w+)?)"
```

NTFS ADS - wscript/cscript

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\*\\wscript.exe OR Image=C:\\Windows\\*\\cscript.exe) | regex
CommandLine="(?<!\/)\b\w+(\.\w+)?:\w+(\.\w+)?$"
```

61- NTFS Alternate Data Stream Execution - LOLBAS

NTFS Alternate Data Streams (ADSs) may be used by adversaries as a means of evading security tools by storing malicious data or binaries in file attribute metadata. ADSs are also powerful because their contents can be directly executed by various Windows tools; accordingly, this analytic looks at common ways of executing ADSs using Living off the Land Binaries and Scripts (LOLBAS).

NTFS ADS - control

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\System32\\control.exe OR
Image=C:\\Windows\SysWOW64\\control.exe) | regex CommandLine="(\w+(\.\w+)?):(\w+\.dll)"
```

NTFS ADS - appvlp

```
index=__sysmon_index__ EventCode=1 (Image="C:\\Program Files\\Microsoft Office\\root\\Client\\AppVLP.exe" OR
Image="C:\\Program Files (x86)\\Microsoft Office\\root\\Client\\AppVLP.exe") | regex CommandLine="(\w+(\.\w+)?):
(\w+(\.\w+)?)"
```

NTFS ADS - cmd

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\System32\\cmd.exe OR Image=C:\\Windows\\SysWOW64\\cmd.exe) |
regex CommandLine="-\s+<.*\b(\w+(\.\w+)?):(\w+(\.\w+)?)"
```

NTFS ADS - ftp

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\System32\\ftp.exe OR Image=C:\\Windows\\SysWOW64\\ftp.exe) |
regex CommandLine="-s:(\w+(\.\w+)?):(\w+(\.\w+)?)"
```

NTFS ADS - bash

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\System32\\bash.exe OR C:\\Windows\\SysWOW64\\bash.exe) |
regex CommandLine="-c.*(\w+(\.\w+)?):(\w+(\.\w+)?)"
```

NTFS ADS - mavinject

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\System32\\mavinject.exe OR
C:\\Windows\\SysWOW64\\mavinject.exe) | regex CommandLine="\d+\s+\/INJECTRUNNING.*\b(\w+(\.\w+)?):(\w+(\.\w+)?)"
```

NTFS ADS - bitsadmin

```
index=__sysmon_index__ EventCode=1 (Image=C:\\Windows\\System32\\bitsadmin.exe OR
C:\\Windows\\SysWOW64\\bitsadmin.exe) | regex CommandLine="\/create.*\/addfile.*\/SetNotifyCmdLine.*\b(\w+\.\w+):
(\w+(\.\w+)?)"
```

62- Execution with AT

In order to gain persistence, privilege escalation, or remote execution, an adversary may use the Windows built-in command AT (at.exe) to schedule a command to be run at a specified time, date, and even host. This method has been used by adversaries and administrators alike. Its use may lead to detection of compromised hosts and compromised users if it is used to move laterally. The built-in Windows tool schtasks.exe (CAR-2013-08-001) offers greater flexibility when creating, modifying, and enumerating tasks. For these reasons, schtasks.exe is more commonly used by administrators, tools/scripts, and power users.

```
index=__your_sysmon_index__ Image="C:\\Windows\\*\\at.exe"|stats values(CommandLine) as "Command Lines" by
ComputerName
```

63- Running executables with same hash and different names

Executables are generally not renamed, thus a given hash of an executable should only have ever one name. Identifying instances where multiple process names share the same hash may find cases where tools are copied by attackers to different folders or hosts to avoid detection.

Although this analytic was initially based on MD5 hashes, it is equally applicable to any hashing convention.

```
index=__your_sysmon_index__ EventCode=1|stats dc(Hashes) as Num_Hashes values(Hashes) as "Hashes" by Image|where
Num_Hashes > 1
```

64- Suspicious Arguments

Malicious actors may rename built-in commands or external tools, such as those provided by SysInternals, to better blend in with the environment. In those cases, the file path name is arbitrary and may blend in well with the background. If the arguments are closely inspected, it may be possible to infer what tools are running and understand what an adversary is doing. When any legitimate software shares the same command lines, it must be whitelisted according to the expected parameters.

```
index=__your_sysmon_index__ EventCode=1 (CommandLine="* -R * -pw*" OR CommandLine="* -pw * *@*" OR
CommandLine="*sekurlsa*" OR CommandLine="* -hp *" OR CommandLine="* a *")
```

65- User Login Activity Monitoring

Monitoring logon and logoff events for hosts on the network is very important for situational awareness. This information can be used as an indicator of unusual activity as well as to corroborate activity seen elsewhere.

Could be applied to a number of different types of monitoring depending on what information is desired. Some use cases include monitoring for all remote connections and building login timelines for users. Logon events are Windows Event Code 4624 for Windows Vista and above, 518 for pre-Vista. Logoff events are 4634 for Windows Vista and above, 538 for pre-Vista.

```
index=__your_win_event_log_index__ EventCode=4624|search NOT [search index=__your_win_event_log_index__
EventCode=4624|top 30 Account_Name|table Account_Name]
```

66- PowerShell Execution

PowerShell is a scripting environment included with Windows that is used by both attackers and administrators. Execution of PowerShell scripts in most Windows versions is opaque and not typically secured by antivirus which makes using PowerShell an easy way to circumvent security measures. This analytic detects execution of PowerShell scripts.

Powershell can be used to hide monitored command line execution such as:

- `net use`
- `sc start`

```
index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\*\\powershell.exe"
ParentImage!="C:\\Windows\\explorer.exe"|stats values(CommandLine) as "Command Lines" values(ParentImage) as
"Parent Images" by ComputerName
```

67- Services launching Cmd

Windows runs the Service Control Manager (SCM) within the process `services.exe`. Windows launches services as independent processes or DLL loads within a svchost.exe group. To be a legitimate service, a process (or DLL) must have the appropriate service entry point SvcMain. If an application does not have the entry point, then it will timeout (default is 30 seconds) and the process will be killed.

To survive the timeout, adversaries and red teams can create services that direct to `cmd.exe` with the flag `/c`, followed by the desired command. The `/c` flag causes the command shell to run a command and immediately exit. As a result, the desired program will remain running and it will report an error starting the service. This analytic will catch that command prompt instance that is used to launch the actual malicious executable. Additionally, the children and descendants of services.exe will run as a SYSTEM user by default. Thus, services are a convenient way for an adversary to gain Persistence and Privilege Escalation.

```
index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\*\\cmd.exe" ParentImage="C:\\Windows\\*\\services.exe"
```

68- Command Launched from WinLogon

An adversary can use accessibility features (Ease of Access), such as StickyKeys or Utilman, to launch a command shell from the logon screen and gain SYSTEM access. Since an adversary does not have physical access to the machine, this technique must be run within Remote Desktop. To prevent an adversary from getting to the login screen without first authenticating, Network-Level Authentication (NLA) must be enabled. If a debugger is set up for one of the accessibility features, then it will intercept the process launch of the feature and instead execute a new command line. This analytic looks for instances of `cmd.exe` or `powershell.exe` launched directly from the logon process, `winlogon.exe` . It should be used in tandem with CAR-2014-11-003, which detects the accessibility programs in the command line.

```
index=__your_sysmon_index__ EventCode=1 ParentImage="C:\\Windows\\*\\winlogon.exe" Image="C:\\Windows\\*\\cmd.exe"
```

69- Host Discovery Commands

When entering on a host for the first time, an adversary may try to discover information about the host. There are several built-in Windows commands that can be used to learn about the software configurations, active users, administrators, and networking configuration. These commands should be monitored to identify when an adversary is learning information about the system and environment. The information returned may impact choices an adversary can make when establishing persistence, escalating privileges, or moving laterally.

Because these commands are built in, they may be run frequently by power users or even by normal users. Thus, an analytic looking at this information should have well-defined white- or blacklists, and should consider looking at an anomaly detection approach, so that this information can be learned dynamically.

```
index=__your_sysmon_index__ EventCode=1 (Image="C:\\Windows\\*\\hostname.exe" OR
Image="C:\\Windows\\*\\ipconfig.exe" OR Image="C:\\Windows\\*\\net.exe" OR Image="C:\\Windows\\*\\quser.exe" OR
Image="C:\\Windows\\*\\qwinsta.exe" OR (Image="C:\\Windows\\*\\sc.exe" AND (CommandLine="* query *" OR
CommandLine="* qc *")) OR Image="C:\\Windows\\*\\systeminfo.exe" OR Image="C:\\Windows\\*\\tasklist.exe" OR
Image="C:\\Windows\\*\\whoami.exe")|stats values(Image) as "Images" values(CommandLine) as "Command Lines" by
ComputerName
```

70- Create Remote Process via WMIC

Adversaries may use Windows Management Instrumentation (WMI) to move laterally, by launching executables remotely.The analytic CAR-2014-12-001 describes how to detect these processes with network traffic monitoring and process monitoring on the target host. However, if the command line utility `wmic.exe` is used on the source host, then it can additionally be detected on an analytic. The command line on the source host is constructed into something like `wmic.exe /node:"\<hostname\>" process call create "\<command line\>"` . It is possible to also connect via IP address, in which case the string `"\<hostname\>"` would instead look like `IP Address` .

Although this analytic was created after CAR-2014-12-001, it is a much simpler (although more limited) approach. Processes can be created remotely via WMI in a few other ways, such as more direct API access or the built-in utility

```
index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\*\\wmic.exe" CommandLine="* process call create
*"|search CommandLine="* /node:*"
```

71- UAC Bypass

Bypassing user account control (UAC Bypass) is generally done by piggybacking on a system process that has auto-escalate privileges. This analytic looks to detect those cases as described by the open-source UACME tool.

```
index=_your_sysmon_index_ EventCode=1 IntegrityLevel=High|search
(ParentCommandLine="\"c:\\windows\\system32\\dism.exe\"*""*.xml" AND
Image!="c:\\users\\*\\appdata\\local\\temp\\*\\dismhost.exe") OR ParentImage=c:\\windows\\system32\\fodhelper.exe
OR (CommandLine="\"c:\\windows\\system32\\wusa.exe\"*/quiet*" AND User!=NOT_TRANSLATED AND
CurrentDirectory=c:\\windows\\system32\\ AND ParentImage!=c:\\windows\\explorer.exe) OR
CommandLine="*.exe\"*cleanmgr.exe /autoclean*" OR (ParentImage="c:\\windows\\*dccw.exe" AND
Image!="c:\\windows\\system32\\cttune.exe") OR Image="c:\\program files\\windows media player\\osk.exe" OR
ParentImage="c:\\windows\\system32\\slui.exe"|eval
PossibleTechniques=case(like(lower(ParentCommandLine),"%c:\\windows\\system32\\dism.exe%"), "UACME #23",
like(lower(Image),"c:\\program files\\windows media player\\osk.exe"), "UACME #32",
like(lower(ParentImage),"c:\\windows\\system32\\fodhelper.exe"),  "UACME #33",
like(lower(CommandLine),"%.exe\"%cleanmgr.exe /autoclean%"), "UACME #34",
like(lower(Image),"c:\\windows\\system32\\wusa.exe"), "UACME #36",
like(lower(ParentImage),"c:\\windows\\%dccw.exe"), "UACME #37",
like(lower(ParentImage),"c:\\windows\\system32\\slui.exe"), "UACME #45")
```

72- Generic Regsvr32

Regsvr32 can be used to execute arbitrary code in the context of a Windows signed binary, which can be used to bypass application whitelisting. This analytic looks for suspicious usage of the tool. It's not likely that you'll get millions of hits, but it does occur during normal activity so some form of baselining would be necessary for this to be an alerting analytic. Alternatively, it can be used for hunt by looking for new or anomalous DLLs manually.

```
index=__your_sysmon_data__ EventCode=1 regsvr32.exe | search ParentImage="*regsvr32.exe" AND
Image!="*regsvr32.exe*"
```

73- Squiblydoo

Squiblydoo is a specific usage of regsvr32.dll to load a COM scriptlet directly from the internet and execute it in a way that bypasses application whitelisting. It can be seen by looking for regsvr32.exe executions that load the scrobj.dll (which execute the COM scriptlet) or, if that is too noisy, those that also load content directly via HTTP or HTTPS.

Squiblydoo was first written up by Casey Smith at Red Canary, though that blog post is no longer accessible.

```
index=__your_sysmon_events__ EventCode=1 regsvr32.exe scrobj.dll | search Image="*regsvr32.exe"
```

74- Credential Dumping via Mimikatz

Credential dumpers like Mimikatz can be loaded into memory and from there read data from another processes. This analytic looks for instances where processes are requesting specific permissions to read parts of the LSASS process in order to detect when credential dumping is occurring. One weakness is that all current implementations are "overtuned" to look for common access patterns used by Mimikatz.

```
index=__your_sysmon_data__ EventCode=10
TargetImage="C:\\WINDOWS\\system32\\lsass.exe"
(GrantedAccess=0x1410 OR GrantedAccess=0x1010 OR GrantedAccess=0x1438 OR GrantedAccess=0x143a OR
GrantedAccess=0x1418)
CallTrace="C:\\windows\\SYSTEM32\\ntdll.dll+*|C:\\windows\\System32\\KERNELBASE.dll+20edd|UNKNOWN(*)"
| table _time hostname user SourceImage GrantedAccess
```

75- Access Permission Modification

Adversaries sometimes modify object access rights at the operating system level. There are varying motivations behind this action - they may not want some files/objects to be changed on systems for persistence reasons and therefore provide admin only rights; also, they may want files to be accessible with lower levels of permissions.

```
index=__your_windows_security_log_index__ EventCode=4670 Object_Type="File" Security_ID!="NT AUTHORITY\\SYSTEM"
```

76- Lsass Process Dump via Procdump

ProcDump is a sysinternal command-line utility whose primary purpose is monitoring an application for CPU spikes and generating crash dumps during a spike that an administrator or developer can use to determine the cause of the spike.

ProcDump may be used to dump the memory space of lsass.exe to disk for processing with a credential access tool such as Mimikatz. This is performed by launching procdump.exe as a privileged user with command line options indicating that lsass.exe should be dumped to a file with an arbitrary name.

```
index=__your_sysmon_index__ EventCode=1 Image="*\\procdump*.exe" CommandLine="*lsass*"
```

77- Credential Dumping via Windows Task Manager

The Windows Task Manager may be used to dump the memory space of `lsass.exe` to disk for processing with a credential access tool such as Mimikatz. This is performed by launching Task Manager as a privileged user, selecting `lsass.exe` , and clicking “Create dump file”. This saves a dump file to disk with a deterministic name that includes the name of the process being dumped.

```
index=__your_sysmon_index__ EventCode=11 TargetFilename="*lsass*.dmp" Image="C:\\Windows\\*\\taskmgr.exe"
```

78- Active Directory Dumping via NTDSUtil

The NTDSUtil tool may be used to dump a Microsoft Active Directory database to disk for processing with a credential access tool such as Mimikatz. This is performed by launching `ntdsutil.exe` as a privileged user with command line arguments indicating that media should be created for offline Active Directory installation and specifying a folder path. This process will create a copy of the Active Directory database, `ntds.dit` , to the specified folder path.

```
index=__your_sysmon_index__ EventCode=11 TargetFilename="*ntds.dit" Image="*ntdsutil.exe"
```

79- Shadow Copy Deletion

The Windows Volume Shadow Copy Service is a built-in OS feature that can be used to create backup copies of files and volumes.

Adversaries may delete these shadow copies, typically through the usage of system utilities such as vssadmin.exe or wmic.exe, in order prevent file and data recovery. This technique is commonly employed for this purpose by ransomware.

Vssadmin.exe delete shadows

```
index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\System32\\vssadmin.exe" CommandLine="*delete shadows*"
```

WMIC shadowcopy delete

```
index=__your_sysmon_index__ EventCode=1 Image="C:\\Windows\\*\\wmic.exe" CommandLine="*shadowcopy delete*"
```

80- MiniDump of LSASS

This analytic detects the minidump variant of credential dumping where a process opens lsass.exe in order to extract credentials using the Win32 API call MiniDumpWriteDump. Tools like SafetyKatz, SafetyDump, and Outflank-Dumpert default to this variant and may be detected by this analytic, though keep in mind that not all options for using those tools will result in this specific behavior.

The analytic is based on a Sigma analytic contributed by Samir Bousseaden and written up in a blog on MENASEC. It looks for a call trace that includes either dbghelp.dll or dbgcore.dll, which export the relevant functions/permissions to perform the dump. It also detects using the Windows Task Manager (taskmgr.exe) to dump lsass, which is described in CAR-2019-08-001. In this iteration of the Sigma analytic, the `GrantedAccess` filter isn’t included because it didn’t seem to filter out any false positives and introduces the potential for evasion.

```
index=__your_sysmon_index__ EventCode=10 TargetImage="C:\\windows\\system32\\lsass.exe" (CallTrace="*dbghelp.dll*"
OR CallTrace="*dbgcore.dll*")| table _time host SourceProcessId SourceImage
```

81- Rare LolBAS Command Lines

LoLBAS are binaries and scripts that are built in to Windows, frequently are signed by Microsoft, and may be used by an attacker. Some LoLBAS are used very rarely and it might be possible to alert every time they're used (this would depend on your environment), but many others are very common and can't be simply alerted on.

This analytic takes all instances of LoLBAS execution and then looks for instances of command lines that are not normal in the environment. This can detect attackers (which will tend to need the binaries for something different than normal usage) but will also tend to have false positives.

The analytic needs to be tuned. The `1.5` in the query is the number of standard deviations away to look. It can be tuned up to filter out more noise and tuned down to get more results. This means it is probably best as a hunting analytic when you have analysts looking at the screen and able to tune the analytic up and down, because the threshold may not be stable for very long.

```
index=__your_sysmon_index__ EventCode=1 (OriginalFileName = At.exe OR OriginalFileName = Atbroker.exe OR
OriginalFileName = Bash.exe OR OriginalFileName = Bitsadmin.exe OR OriginalFileName = Certutil.exe OR
OriginalFileName = Cmd.exe OR OriginalFileName = Cmdkey.exe OR OriginalFileName = Cmstp.exe OR OriginalFileName =
Control.exe OR OriginalFileName = Csc.exe OR OriginalFileName = Cscript.exe OR OriginalFileName = Dfsvc.exe OR
OriginalFileName = Diskshadow.exe OR OriginalFileName = Dnscmd.exe OR OriginalFileName = Esentutl.exe OR
OriginalFileName = Eventvwr.exe OR OriginalFileName = Expand.exe OR OriginalFileName = Extexport.exe OR
OriginalFileName = Extrac32.exe OR OriginalFileName = Findstr.exe OR OriginalFileName = Forfiles.exe OR
OriginalFileName = Ftp.exe OR OriginalFileName = Gpscript.exe OR OriginalFileName = Hh.exe OR OriginalFileName =
Ie4uinit.exe OR OriginalFileName = Ieexec.exe OR OriginalFileName = Infdefaultinstall.exe OR OriginalFileName =
Installutil.exe OR OriginalFileName = Jsc.exe OR OriginalFileName = Makecab.exe OR OriginalFileName = Mavinject.exe
OR OriginalFileName = Microsoft.Workflow.r.exe OR OriginalFileName = Mmc.exe OR OriginalFileName = Msbuild.exe OR
OriginalFileName = Msconfig.exe OR OriginalFileName = Msdt.exe OR OriginalFileName = Mshta.exe OR OriginalFileName
= Msiexec.exe OR OriginalFileName = Odbcconf.exe OR OriginalFileName = Pcalua.exe OR OriginalFileName = Pcwrun.exe
OR OriginalFileName = Presentationhost.exe OR OriginalFileName = Print.exe OR OriginalFileName = Reg.exe OR
OriginalFileName = Regasm.exe OR OriginalFileName = Regedit.exe OR OriginalFileName = Register-cimprovider.exe OR
OriginalFileName = Regsvcs.exe OR OriginalFileName = Regsvr32.exe OR OriginalFileName = Replace.exe OR
OriginalFileName = Rpcping.exe OR OriginalFileName = Rundll32.exe OR OriginalFileName = Runonce.exe OR
OriginalFileName = Runscripthelper.exe OR OriginalFileName = Sc.exe OR OriginalFileName = Schtasks.exe OR
OriginalFileName = Scriptrunner.exe OR OriginalFileName = SyncAppvPublishingServer.exe OR OriginalFileName =
Tttracer.exe OR OriginalFileName = Verclsid.exe OR OriginalFileName = Wab.exe OR OriginalFileName = Wmic.exe OR
OriginalFileName = Wscript.exe OR OriginalFileName = Wsreset.exe OR OriginalFileName = Xwizard.exe OR
OriginalFileName = Advpack.dll OR OriginalFileName = Comsvcs.dll OR OriginalFileName = Ieadvpack.dll OR
OriginalFileName = Ieaframe.dll OR OriginalFileName = Mshtml.dll OR OriginalFileName = Pcwutl.dll OR
OriginalFileName = Setupapi.dll OR OriginalFileName = Shdocvw.dll OR OriginalFileName = Shell32.dll OR
OriginalFileName = Syssetup.dll OR OriginalFileName = Url.dll OR OriginalFileName = Zipfldr.dll OR OriginalFileName
= Appvlp.exe OR OriginalFileName = Bginfo.exe OR OriginalFileName = Cdb.exe OR OriginalFileName = csi.exe OR
OriginalFileName = Devtoolslauncher.exe OR OriginalFileName = dnx.exe OR OriginalFileName = Dxcap.exe OR
OriginalFileName = Excel.exe OR OriginalFileName = Mftrace.exe OR OriginalFileName = Msdeploy.exe OR
OriginalFileName = msxsl.exe OR OriginalFileName = Powerpnt.exe OR OriginalFileName = rcsi.exe OR OriginalFileName
= Sqler.exe OR OriginalFileName = Sqlps.exe OR OriginalFileName = SQLToolsPS.exe OR OriginalFileName = Squirrel.exe
OR OriginalFileName = te.exe OR OriginalFileName = Tracker.exe OR OriginalFileName = Update.exe OR OriginalFileName
= vsjitdebugger.exe OR OriginalFileName = Winword.exe OR OriginalFileName = Wsl.exe OR OriginalFileName =
CL_Mutexverifiers.ps1 OR OriginalFileName = CL_Invocation.ps1 OR OriginalFileName = Manage-bde.wsf OR
OriginalFileName = Pubprn.vbs OR OriginalFileName = Slmgr.vbs OR OriginalFileName = Syncappvpublishingserver.vbs OR
OriginalFileName = winrm.vbs OR OriginalFileName = Pester.bat)|eval CommandLine=lower(CommandLine)|eventstats
count(process) as procCount by process|eventstats avg(procCount) as avg stdev(procCount) as stdev|eval lowerBound=
(avg-stdev*1.5)|eval isOutlier=if((procCount < lowerBound),1,0)|where isOutlier=1|table host, Image, ParentImage,
CommandLine, ParentCommandLine, procCount
```

References:

Splunk How-To

car.mitre.org

Analytics