FROM TECHNIQUE TO DETECTION

Rapid prototyping of ATT&CK-based analytics

Paul Ewing @_paulewing
Ross Wolf @rw_access

ENDGAME.

ATT&CK Detections?

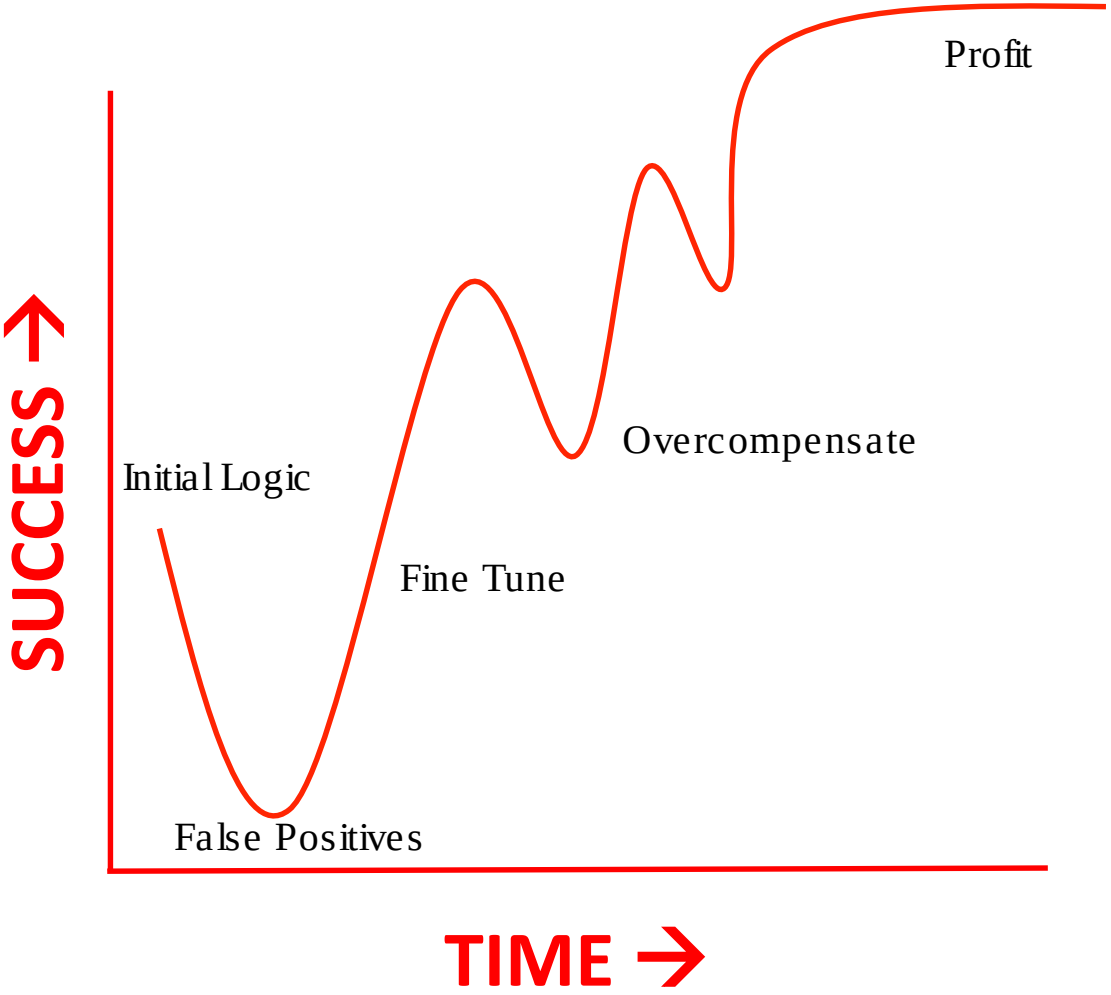
Will be powerful
Will enable hunters
Will transform your SOC

ATT&CK Detections?

Will not be as easy as described
Will not always lead to analytics
Will not be a silver bullet

TIMELINE OF A SOLID ANALYTIC

ATT&CK-based analytics required continual grooming as their success depends on it



1. SELECT YOUR TECHNIQUE

After performing gap-analysis and understanding your environment.

Windows Technique Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Drive-by Compromise	CMSTP	Accessibility Features	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	Application Deployment Software	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	Command-Line Interface	AppCert DLLs	Accessibility Features	BITS Jobs	Brute Force	Application Window Discovery	Distributed Component Object Model	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Control Panel Items	AppInit DLLs	AppCert DLLs	Binary Padding	Credential Dumping	Browser Bookmark Discovery	Exploitation of Remote Services	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Dynamic Data Exchange	Application Shimming	AppInit DLLs	Bypass User Account Control	Credentials in Files	File and Directory Discovery	Logon Scripts	Data Staged	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Execution through API	Authentication Package	Application Shimming	CMSTP	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Information Repositories	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Execution through Module Load	BITS Jobs	Bypass User Account Control	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Local System	Exfiltration Over Command and Control Channel	Data Encoding

ENDGAME.

1. SELECT YOUR TECHNIQUE

After performing gap-analysis and understanding your environment.

Bypass User Account Control

(Redirected from Bypass User Account Control)

Windows User Account Control (UAC) allows a program to elevate its privileges to perform a task under administrator-level permissions by prompting the user for confirmation. The impact to the user ranges from denying the operation under high enforcement to allowing the user to perform the action if they are in the local administrators group and click through the prompt or allowing them to enter an administrator password to complete the action.[1]

If the UAC protection level of a computer is set to anything but the highest level, certain Windows programs are allowed to elevate privileges or execute some elevated COM objects without prompting the user through the UAC notification box.[2][3] An example of this is use of rundll32.exe to load a specifically crafted DLL which loads an auto-elevated COM object and performs a file operation in a protected directory which would typically require elevated access. Malicious software may also be injected into a trusted process to gain elevated privileges without prompting a user.[4] Adversaries can use these techniques to elevate privileges to administrator if the target process is unprotected.

Bypass User Account Control

Technique

ID	T1088
Tactic	Defense Evasion, Privilege Escalation
Platform	Windows
Permissions Required	User, Administrator
Effective Permissions	Administrator
Data Sources	System calls, Process monitoring, Authentication logs, Process command-line parameters
Defense Bypassed	Windows User Account Control
Contributors	Stefan Kanthak, Casey Smith

Many methods have been discovered to bypass UAC. The Github readme page for UACMe contains an extensive list of methods[5] that have been discovered and implemented within UACMe, but may not be a comprehensive list of bypasses. Additional bypass methods are regularly discovered and some used in the wild, such as:

2. DETONATE

Find scripts or commands that perform the technique.

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Ross>
C:\Users\Ross>reg add hkcu\Environment /v windir /d "cmd /K reg delete
hkcu\Environment /v windir /f && REM "
The operation completed successfully.

C:\Users\Ross>schtasks /Run /TN \Microsoft\Windows\DiskCleanup\SilentCleanup /I
SUCCESS: Attempted to run the scheduled task
"\Microsoft\Windows\DiskCleanup\SilentCleanup".

C:\Users\Ross>
```

…after you detonate…be sure to leave yourself some breadcrumbs…

Administrator: c:\windows\system32\cmd.exe

```
The operation completed successfully.

C:\WINDOWS\system32>cmd /c echo FINDMEFINDME
FINDMEFINDME

C:\WINDOWS\system32>
```

https://tyranidslair.blogspot.com/2017/05/exploiting-environment-variables-in.html

3. PREP ANALYSIS

First let's create some PS functions to help us gather and parse Sysmon logs.

```
function Get-EventProps {
  [cmdletbinding()]
  Param (
    [parameter(ValueFromPipeline)]
    $event
  )
    Process {
        $eventXml = [xml]$event.ToXML()
        $eventKeys = $eventXml.Event.EventData.Data
        $Properties = @{}

        For ($i=0; $i -lt $eventKeys.Count; $i++) {
            $Properties[$eventKeys[$i].Name] = $eventKeys[$i].'#text'
        }

        [pscustomobject]$Properties
    }
}

function Get-LatestLogs {
    Get-WinEvent -filterhashtable @{logname="Microsoft-Windows-
         Sysmon/Operational"} -MaxEvents 1000 | Get-EventProps
}

function Get-LatestProcesses {
    Get-WinEvent -filterhashtable @{logname="Microsoft-Windows-
         Sysmon/Operational";id=1} -MaxEvents 1000 | Get-EventProps
}
```

3. PREP ANALYSIS

Let's store the relevant process data

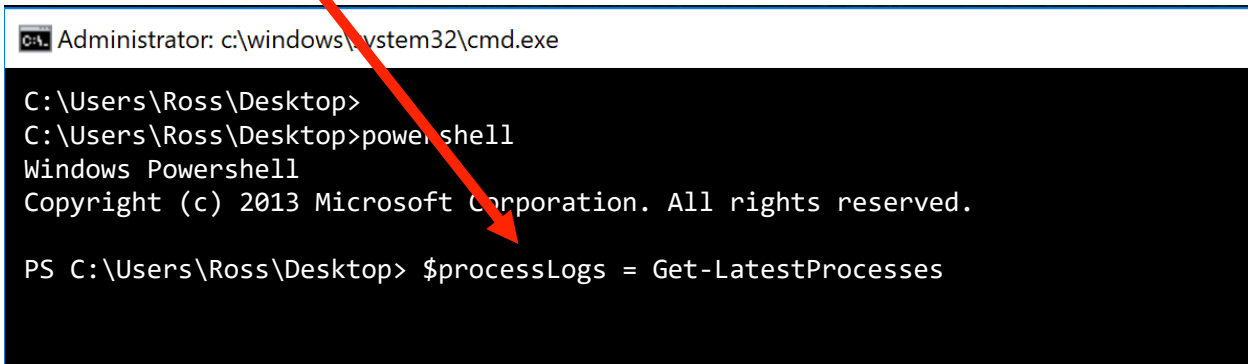
We need this data

We got this data

Bypass User Account Control

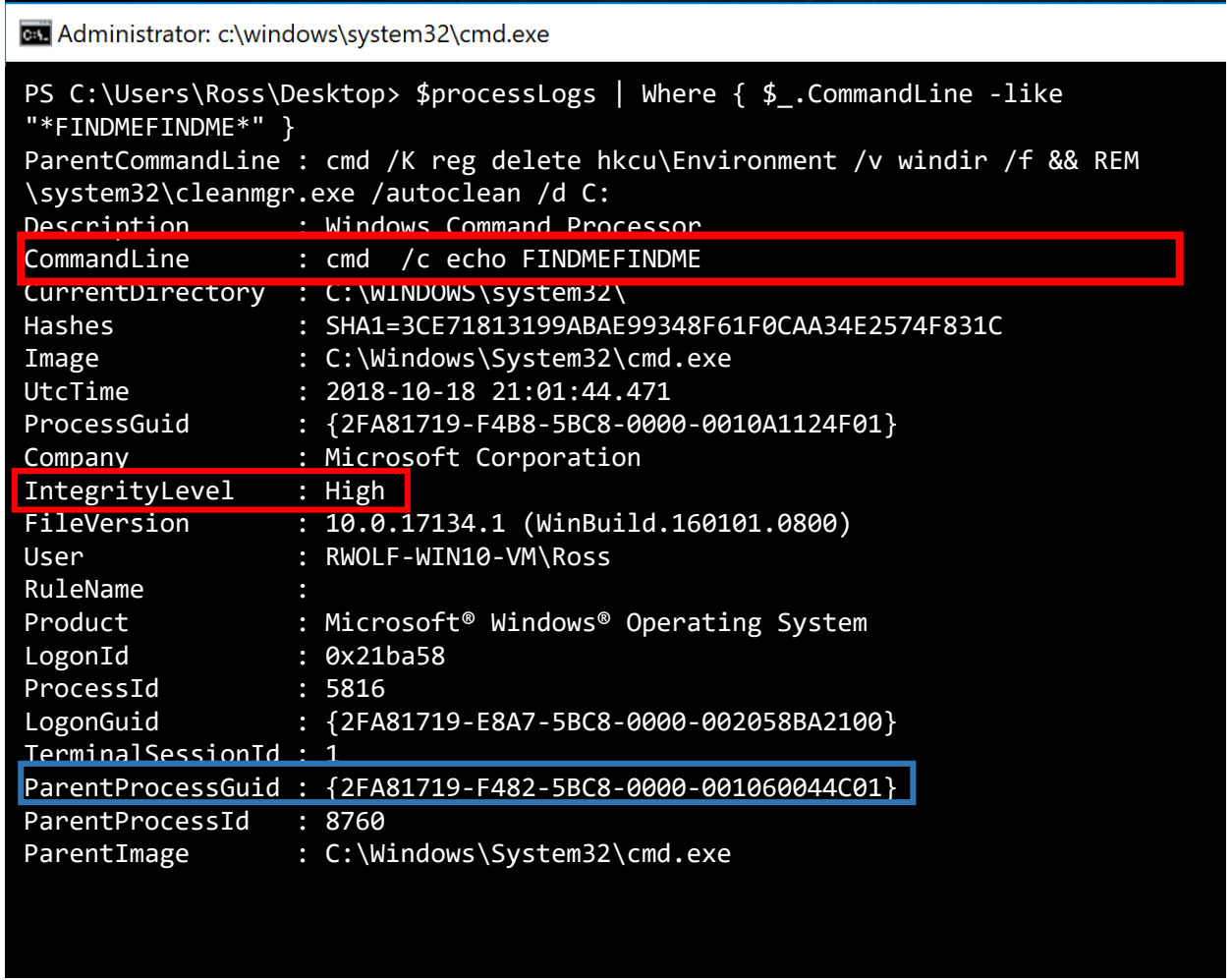
Technique

ID	T1088
Tactic	Defense Evasion, Privilege Escalation
Platform	Windows
Permissions Required	User, Administrator
Effective Permissions	Administrator
Data Sources	System calls, Process monitoring, Authentication logs, Process command-line parameters
Defense Bypassed	Windows User Account Control
Contributors	Stefan Kanthak, Casey Smith



4. EXPLORE DATA

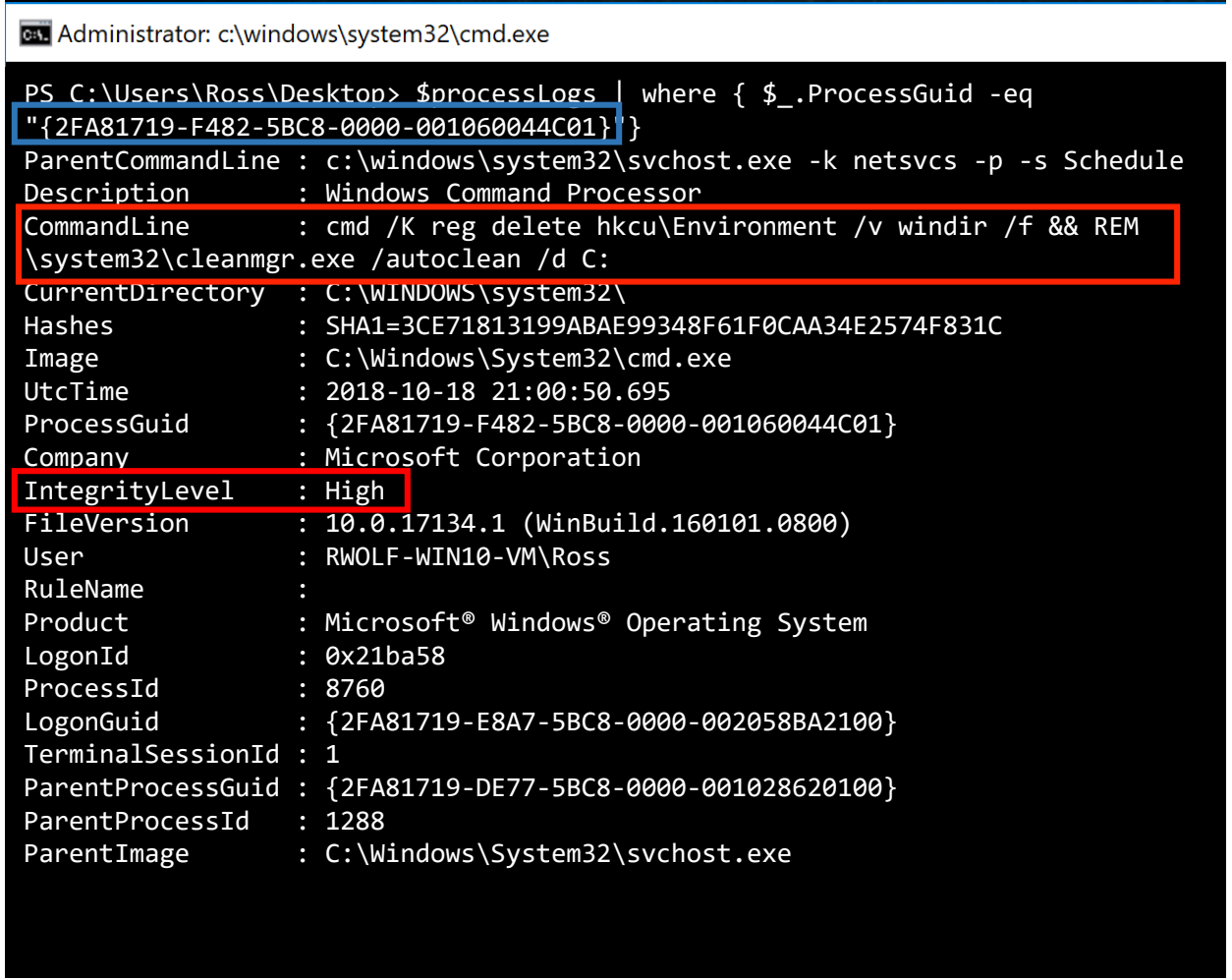
First, find your breadcrumbs that you left earlier.



4. EXPLORE DATA

Walk up the process tree to see when and how the bypass occurred.

We're looking for a **transition** from medium to high integrity.



4. EXPLORE DATA

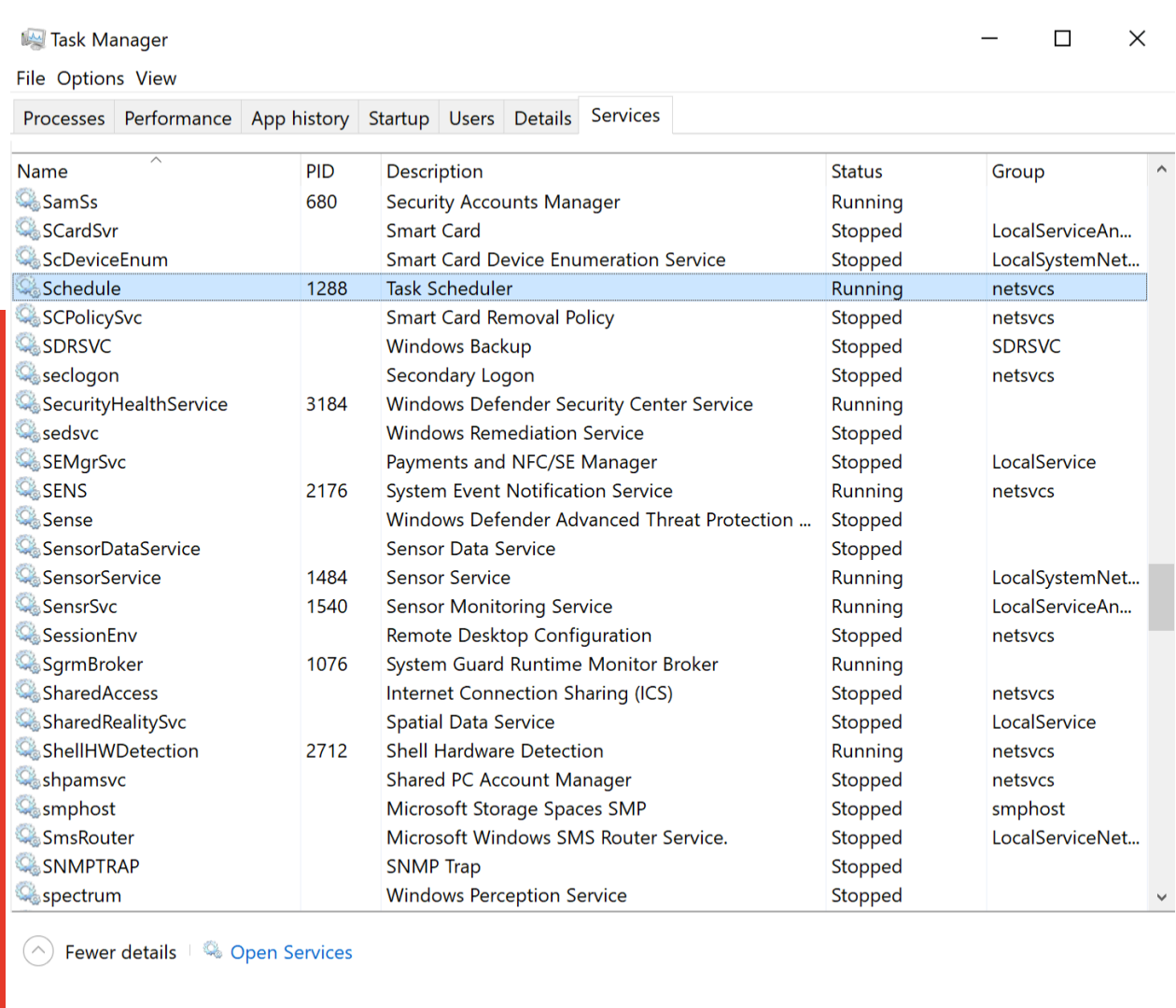
Keep walking up the tree until things look normal again. We found the Schedule service running as normal.

Administrator: c:\windows\system32\cmd.exe

```
PS C:\Users\Ross\Desktop> $processLogs | where { $_.ProcessGuid -eq
"{2FA81719-DE77-5BC8-0000-001028620100}"}
ParentCommandLine : C:\WINDOWS\system32\services.exe
Description       : Host Process for Windows Services
CommandLine       : c:\windows\system32\svchost.exe -k netsvcs -p -s Schedule
CurrentDirectory  : C:\WINDOWS\system32\
Hashes            : SHA1=660B76B6FB802417D513ADC967C5CAF77FC2BAC6
Image             : C:\Windows\System32\svchost.exe
UtcTime           : 2018-10-18 19:26:47.110
ProcessGuid       : {2FA81719-DE77-5BC8-0000-001028620100}
Company           : Microsoft Corporation
IntegrityLevel    : System
FileVersion       : 10.0.17134.1 (WinBuild.160101.0800)
User              : NT AUTHORITY\SYSTEM
RuleName          :
Product           : Microsoft® Windows® Operating System
LogonId           : 0x3e7
ProcessId         : 1288
LogonGuid         : {2FA81719-DE75-5BC8-0000-0020E7030000}
TerminalSessionId : 0
ParentProcessGuid : {2FA81719-DE75-5BC8-0000-0010C6AC0000}
ParentProcessId   : 632
ParentImage       : C:\Windows\System32\services.exe
```

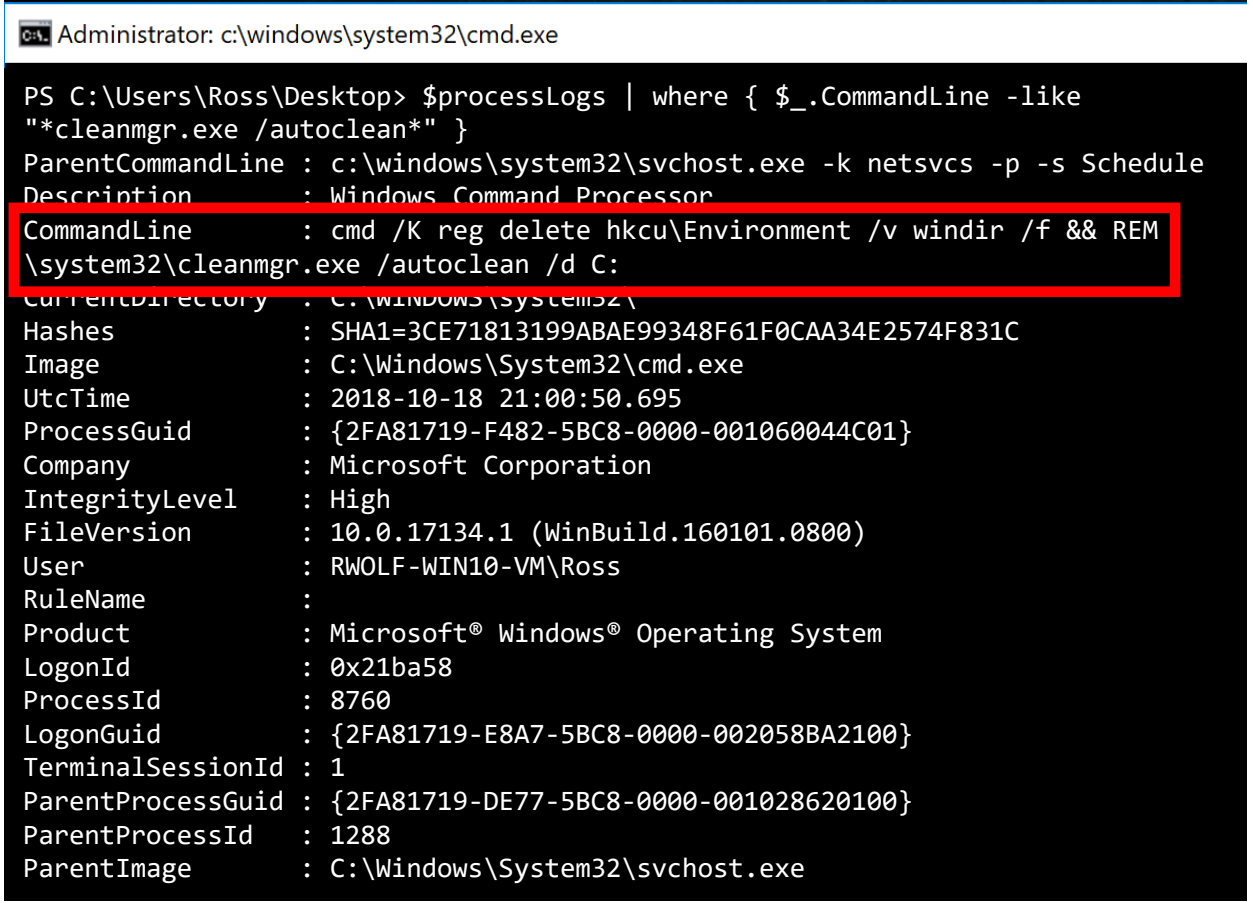
4. EXPLORE DATA

We went too far. The task scheduler ran the SilentCleanup task, but we're interested in the task, not the scheduler.



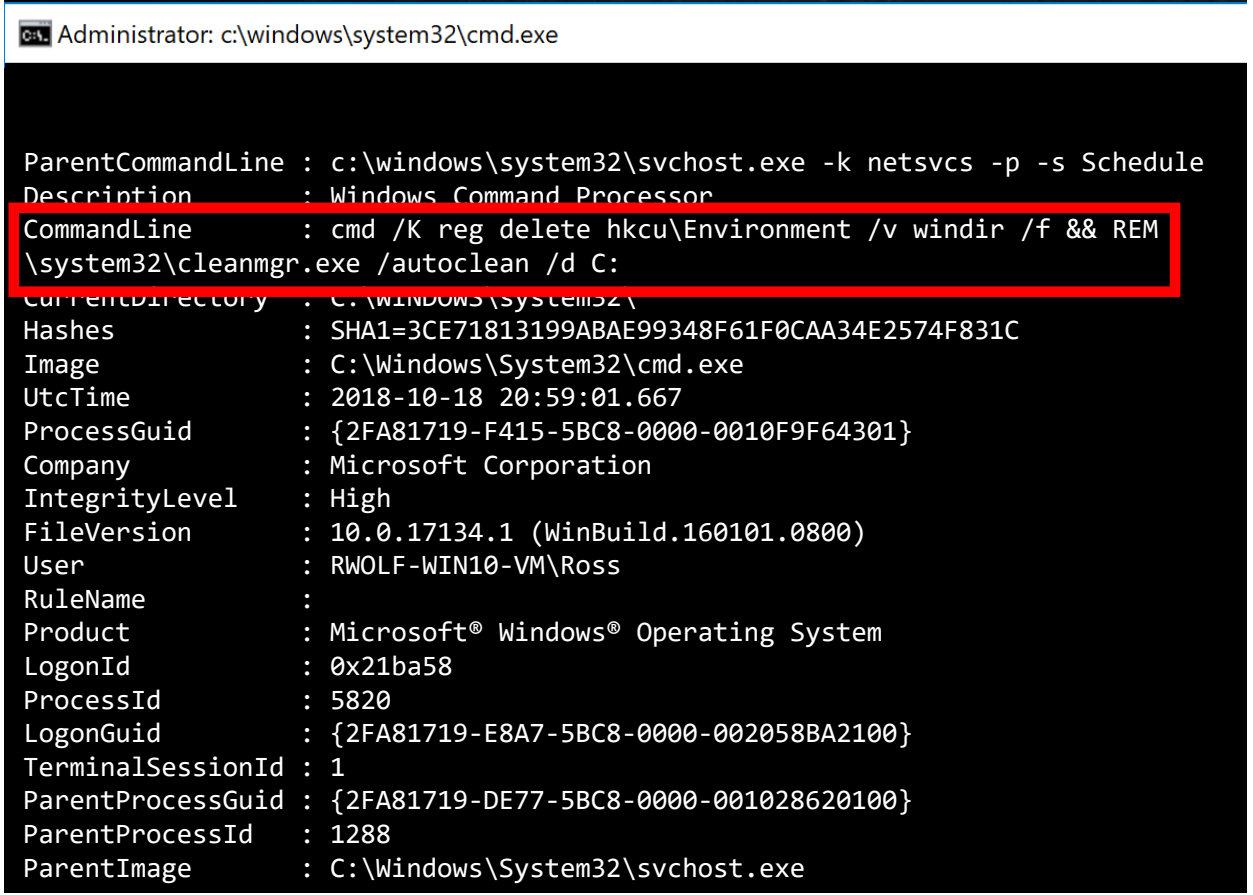
4. EXPLORE DATA

What happens when the SilentCleanup task is not abused?



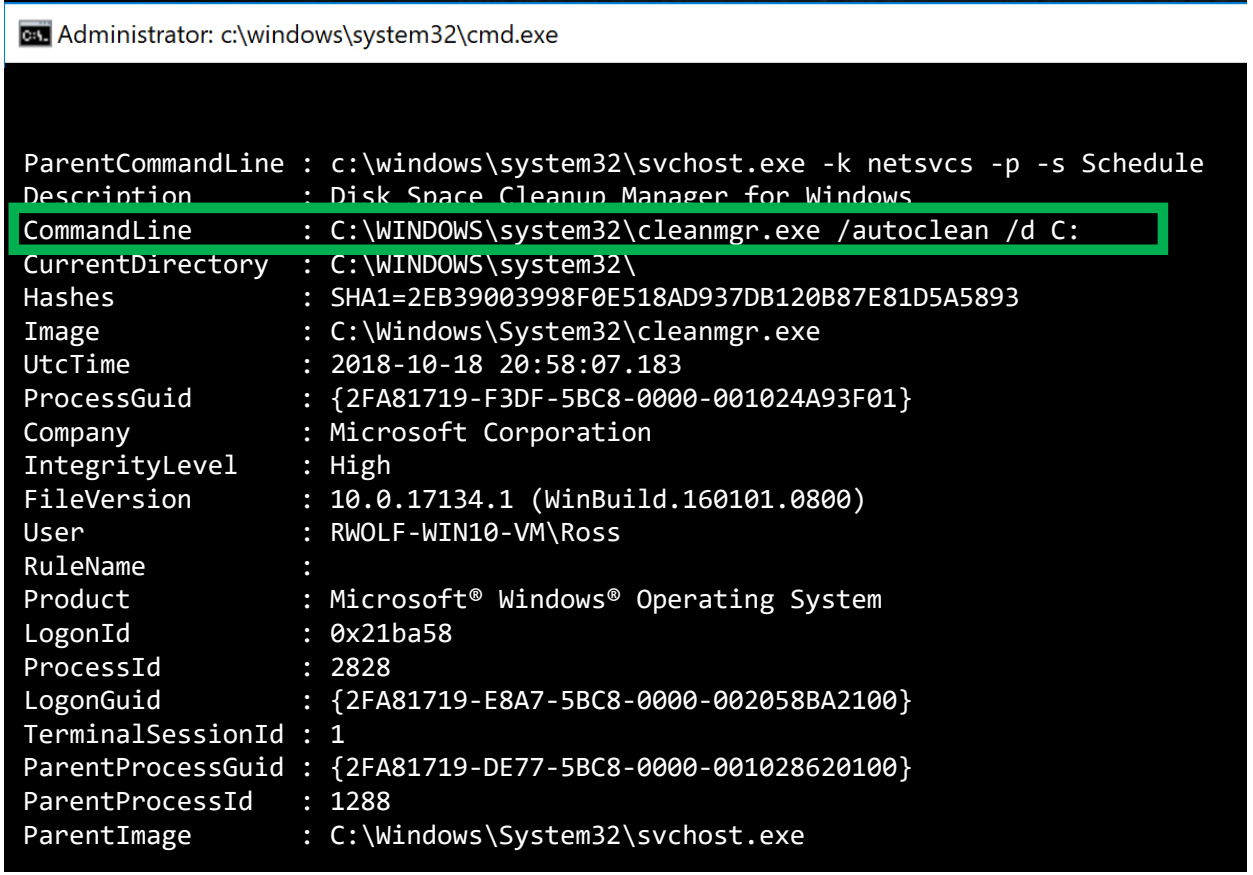
4. EXPLORE DATA

What happens when the SilentCleanup task is not abused?



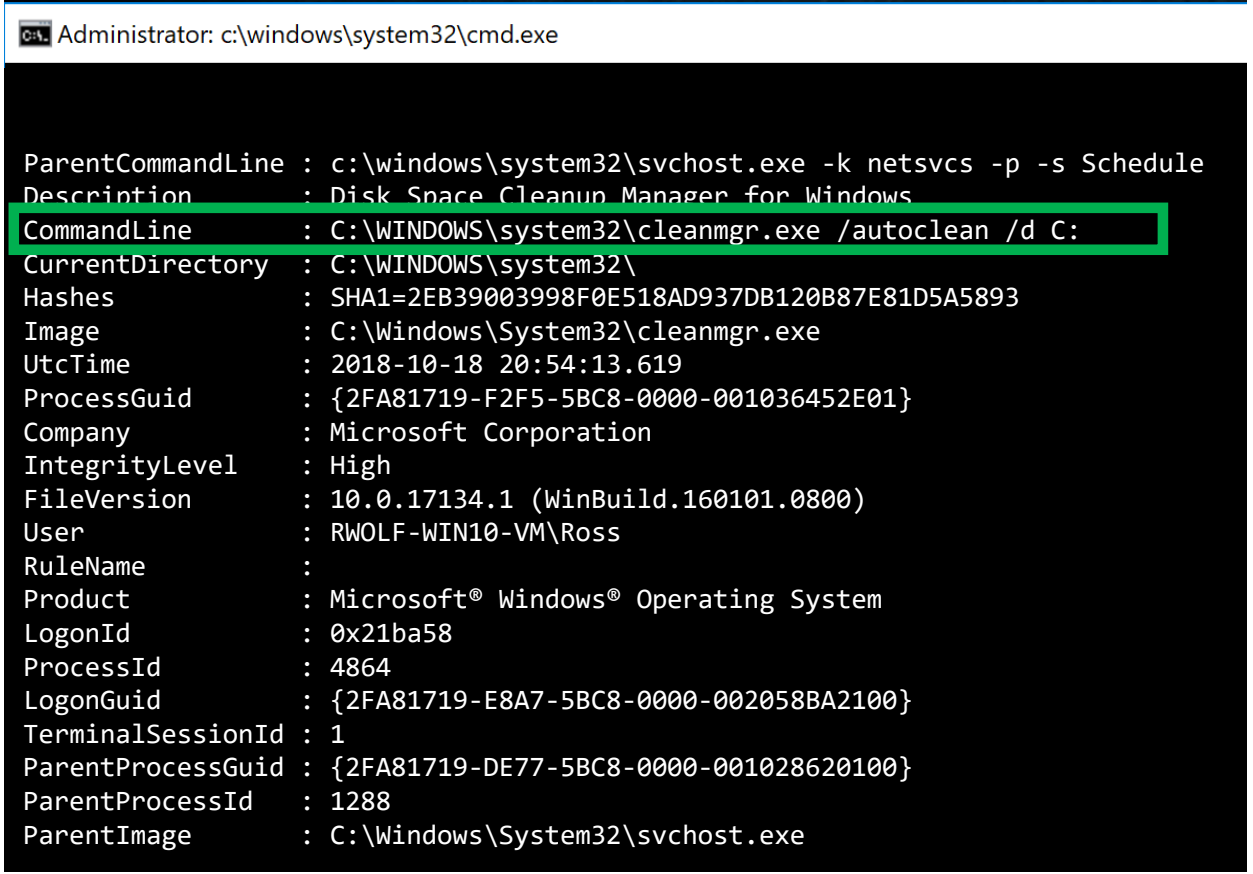
4. EXPLORE DATA

What happens when the SilentCleanup task is not abused?



4. EXPLORE DATA

What happens when the SilentCleanup task is not abused?



5. PROTOTYPE ANALYTIC

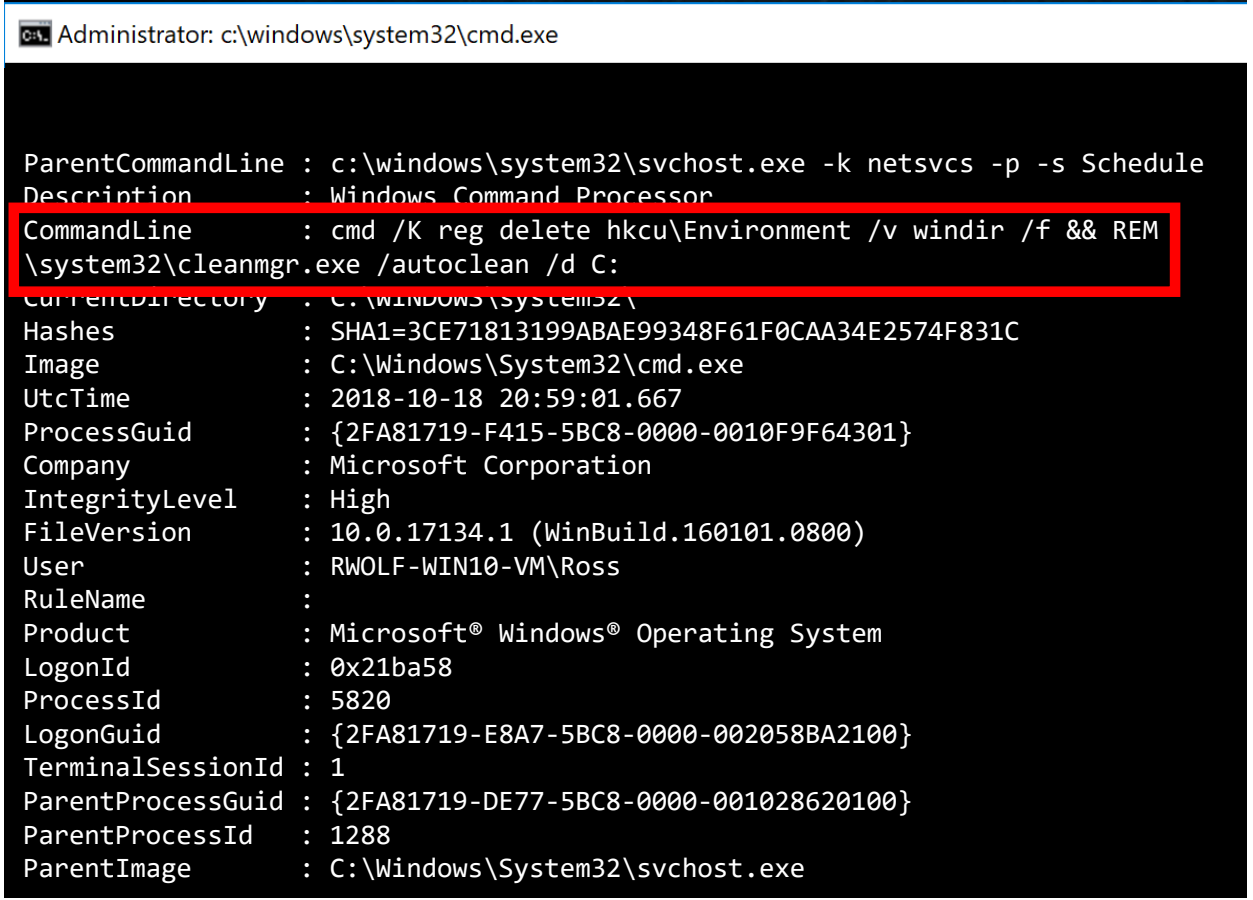
Draft an analytic in PowerShell that detects the malicious behavior without triggering on the benign behavior.

Administrator: c:\windows\system32\cmd.exe

```
PS C:\Users\Ross\Desktop> $processLogs | where {$_.IntegrityLevel -eq "High"}
-and $_.CommandLine -like "* *\system32\cleanmgr.exe /autoclean*" }
ParentCommandLine : c:\windows\system32\svchost.exe -k netsvcs -p -s Schedule
Description       : Windows Command Processor
CommandLine       : cmd /K reg delete hkcu\Environment /v windir /f && REM
\system32\cleanmgr.exe /autoclean /d C:
CurrentDirectory  : C:\WINDOWS\system32\
Hashes            : SHA1=3CE71813199ABAE99348F61F0CAA34E2574F831C
Image             : C:\Windows\System32\cmd.exe
UtcTime           : 2018-10-18 21:00:50.695
ProcessGuid       : {2FA81719-F482-5BC8-0000-001060044C01}
Company           : Microsoft Corporation
IntegrityLevel    : High
FileVersion       : 10.0.17134.1 (WinBuild.160101.0800)
User              : RWOLF-WIN10-VM\Ross
RuleName          :
Product           : Microsoft® Windows® Operating System
LogonId           : 0x21ba58
ProcessId         : 8760
LogonGuid         : {2FA81719-E8A7-5BC8-0000-002058BA2100}
TerminalSessionId : 1
ParentProcessGuid : {2FA81719-DE77-5BC8-0000-001028620100}
ParentProcessId   : 1288
ParentImage       : C:\Windows\System32\svchost.exe
```

5. PROTOTYPE ANALYTIC

Draft an analytic in PowerShell.



6. GO LIVE

Rule looks good!
Let's write an EQL query.


```
process where
  command_line ==  "* *\\system32\\cleanmgr.exe /autoclean*"
```

https://www.endgame.com/blog/technical-blog/introducing-event-query-language

6. GO LIVE

Results…uh oh…
True or false positive?


```
{
  "authentication_id": 224881,
  "command_line": "taskhostw.exe C:\\WINDOWS\\system32\\cleanmgr.exe
                   /autoclean /d C:",
  "event_subtype_full": "creation_event",
  "event_type_full": "process_event",
  "md5": "ce95e236fc9fe2d6f16c926c75b18baf",
  "original_file_name": "taskhostw.exe",
  "parent_process_name": "svchost.exe",
  "parent_process_path": "C:\\Windows\\System32\\svchost.exe",
  "pid": 14920,
  "ppid": 1700,
  "process_name": "taskhostw.exe",
  "process_path": "C:\\Windows\\System32\\taskhostw.exe",
  "sha1": "2a594345fbcaad453c72bd0937cbf67fb43a74df",
  "signature_signer": "Microsoft Windows",
  "signature_status": "trusted",
  "timestamp_utc": "2018-10-02 16:05:32Z",
  ...
}
```


Investigation
Details

24
Total Hits

12/83
Endpoints with Hits

6. GO LIVE

Oh, this is just another benign instance of the Windows task scheduler on a different OS.


```
{
  "authentication_id": 224881,
  "command_line": "taskhostw.exe C:\\WINDOWS\\system32\\cleanmgr.exe
                   /autoclean /d C:",
  "event_subtype_full": "creation_event",
  "event_type_full": "process_event",
  "md5": "ce95e236fc9fe2d6f16c926c75b18baf",
  "original_file_name": "taskhostw.exe",
  "parent_process_name": "svchost.exe",
  "parent_process_path": "C:\\Windows\\System32\\svchost.exe",
  "pid": 14920,
  "ppid": 1700,
  "process_name": "taskhostw.exe",
  "process_path": "C:\\Windows\\System32\\taskhostw.exe",
  "sha1": "2a594345fbcaad453c72bd0937cbf67fb43a74df",
  "signature_signer": "Microsoft Windows",
  "signature_status": "trusted",
  "timestamp_utc": "2018-10-02 16:05:32Z",
  ...
}
```


7. FINE TUNING

Filter results, unique results, or specifically account for false positives in your environment


```
process where
  command_line ==  "* *\\system32\\cleanmgr.exe /autoclean*”
  and process_path != "C:\\Windows\\System32\\taskhostw.exe"
```

Investigation
Details

0
Total Hits

0/83
Endpoints with Hits

CONCLUSION

Seems easy enough?
But FPs are everywhere.

Always calibrate to the software and configurations in your environment

1. Stealthy PowerShell, this module is common huh?

```
image_load where process_name != "powershell.exe" and
    image_name == "System.Management.Automation.ni.dll"
```

2. Emails, Links, Child Browser Process?

```
process where
  parent_process_name == "outlook.exe" and
  process_name in ("wscript.exe", "cmd.exe",
                   "powershell.exe", ...)
```

3. MSHTA Network Connections, never happens?

```
sequence by unique_pid
  [process where process_name == "mshta.exe"]
  [network where event_subtype_full == "*connection_attempt_event"]
```

4. Unusual Lateral Movement via SMB, shouldn't FP?

```
network where destination_port == 445 and pid != 4
| unique process_path
```

Thank You.

ENDGAME.