Subverting ~~INSERT PRODUCT NAME~~ Sysmon

Application of a Formalized Security Product Evasion Methodology

Matt Graeber and Lee Christensen

Black Hat USA 2018

Who are we?

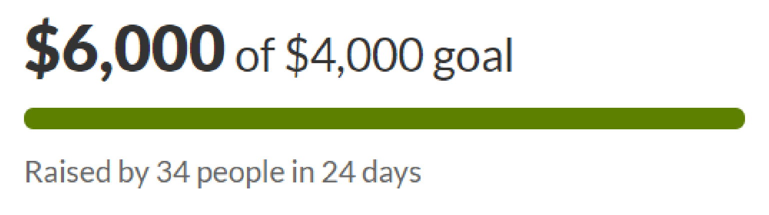
Matt Graeber, Security Researcher @ SpecterOps

- Researcher, Threat Hunter, 🔥 Tradecraft 🔥

Lee Christensen, Security Researcher/Operator @ SpecterOps

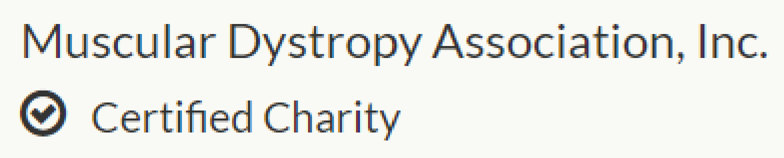
- Researcher, Red Teamer, Threat Hunter
- Likes shiny security things (red and blue)

Why are we wearing these stupid things???



$6,000 of $4,000 goal

Raised by 34 people in 24 days



Muscular Dystropy Association, Inc.

✓ Certified Charity

Outline

1. Goals of an Evasive Adversary
2. Detection and Detection Subversion Methodologies
3. Rationale for Targeting Sysmon
4. Data Collector Subversion Strategies Applied to Sysmon
5. Conclusion

Goals of an Evasive Adversary

Avoid detection at an organizational level

1. Blend in with “normal”
2. Exploit naive defender behaviors/methodology
3. Avoid human eyes

Subverting security solutions is simply an engineering challenge of adversaries.

Adversary Detection Methodology

1. Attack Technique Identification
2. Data Source Identification
3. **Data Collection**
4. Event Transport
5. Event Enrichment and Analysis
6. Malignant/Benign Classification
7. Alerting/Response

At a micro level, security products perform one or more of these

Detection Subversion Methodology

Evading or **tampering with**

any steps of the detection methodology

Rationale for Targeting Sysmon

Defenders use it heavily.

Some vendors take a dependency on it.

We are not picking on Sysmon.

Data Collector Subversion Strategies

Sysmon is a host-based data collection tool (step 3 of the detection methodology)

Our interests: Tampering, Evasion, Attack Surface Analysis

Analysis Strategies

1. Tool Familiarization and Scoping
2. Data Source Resilience Auditing
3. Footprint/Attack Surface Analysis
4. Data Collection Implementation Analysis
5. Configuration Analysis

1. Tool Familiarization and Scoping

Understand purpose, guarantees, and threat models

Install it, configure it, update it, use it

Tool Familiarization and Scoping

Purpose: User-mode activity sensor

Standalone executable + Driver

- No centralized deployment/configuration management
- No analysis capabilities, some enrichment

Guarantees:

- Tamper-resistant against non-admins
- Data sources comprehensively collected (unless filtered by rules)

2. Data Source Resilience Auditing

What are the events and event fields?

What event fields are attacker-controlled?

What fields do defenders likely use?

Generic Rule Evasion Analysis

Identify what can be logged and attributes of the event can be influenced by an attacker (prioritizing non-admin primitives).

Sysmon Supported Rule Types

- **ProcessCreate**
- FileCreateTime
- NetworkConnect
- ProcessTerminate
- DriverLoad
- ImageLoad
- CreateRemoteThread
- RawAccessRead
- ProcessAccess
- FileCreate
- RegistryEvent
- FileCreateStreamHash
- PipeEvent
- WmiEvent

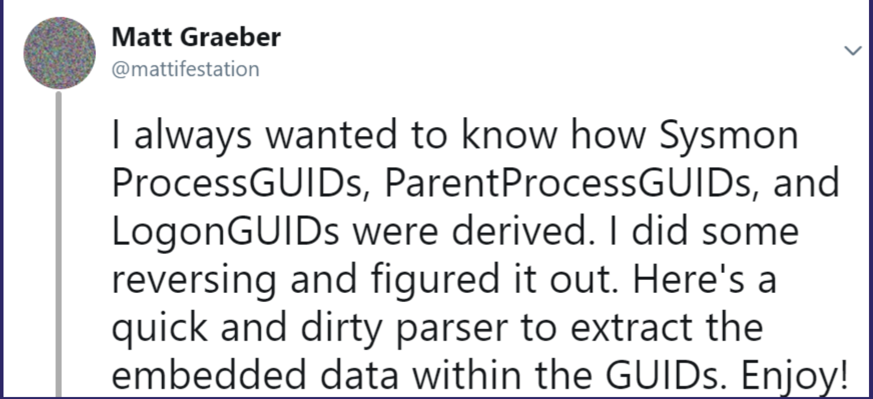
ProcessCreate - Attacker-influenceable Attributes

Image	User	ProcessGuid
CommandLine	ParentImage	ProcessId
CurrentDirectory	ParentCommandLine	LogonGuid
Description	UtcTime	LogonId
FileVersion		TerminalSessionId
Product		IntegrityLevel
Company		Hashes
ParentProcessId		ParentProcessGuid

↑

Also the highest likelihood in which a rule will be written!

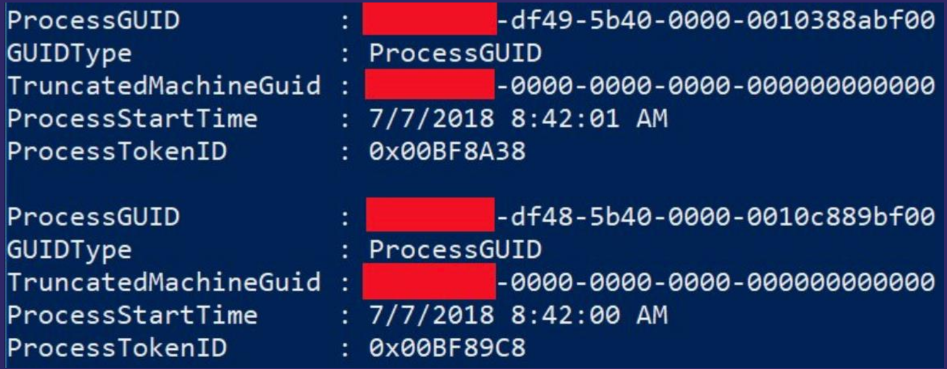
ProcessCreate - Attacker-influenceable Attributes



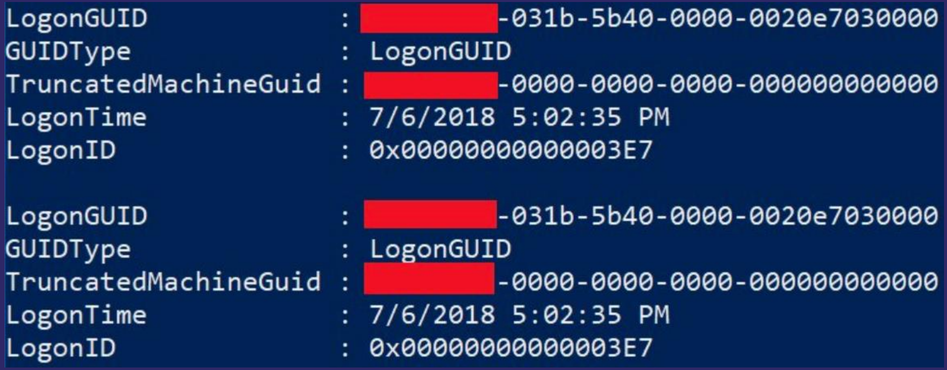
Matt Graeber
@mattifestation

I always wanted to know how Sysmon ProcessGUIDs, ParentProcessGUIDs, and LogonGUIDs were derived. I did some reversing and figured it out. Here's a quick and dirty parser to extract the embedded data within the GUIDs. Enjoy!

https://gist.github.com/mattifestation/0102042160c9a60b2b847378c0ef70b4



```
ProcessGUID           : [redacted]-df49-5b40-0000-0010388abf00
GUIDType              : ProcessGUID
TruncatedMachineGuid  : [redacted]-0000-0000-0000-000000000000
ProcessStartTime      : 7/7/2018 8:42:01 AM
ProcessTokenID        : 0x00BF8A38

ProcessGUID           : [redacted]-df48-5b40-0000-0010c889bf00
GUIDType              : ProcessGUID
TruncatedMachineGuid  : [redacted]-0000-0000-0000-000000000000
ProcessStartTime      : 7/7/2018 8:42:00 AM
ProcessTokenID        : 0x00BF89C8
```

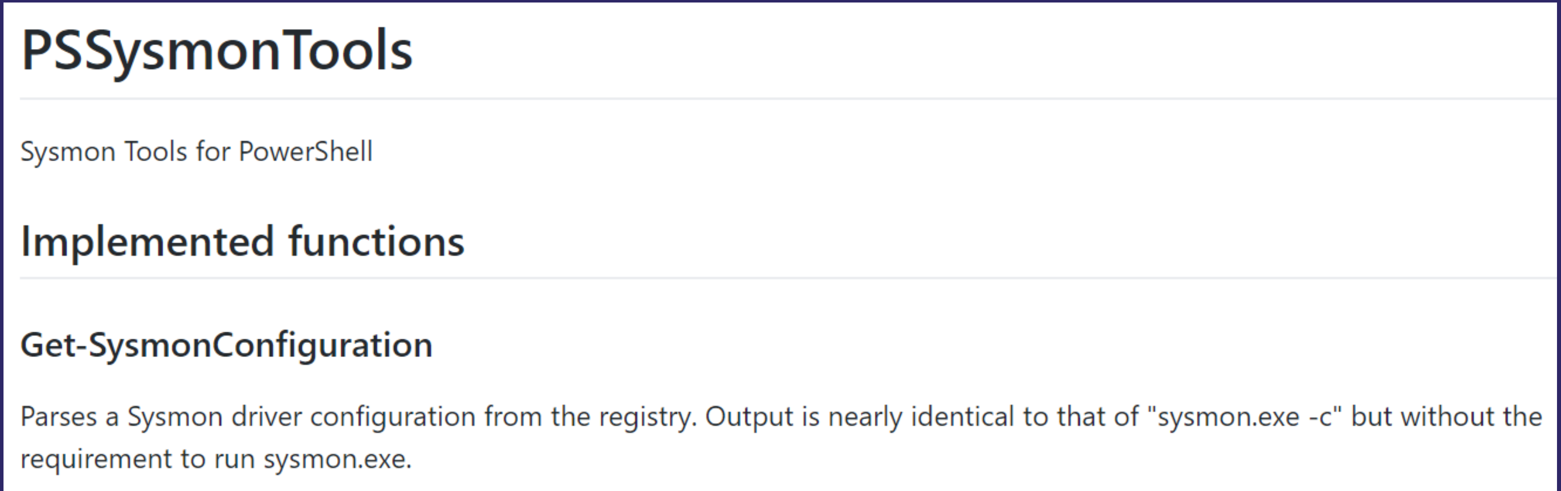


```
LogonGUID             : [redacted]-031b-5b40-0000-0020e7030000
GUIDType              : LogonGUID
TruncatedMachineGuid  : [redacted]-0000-0000-0000-000000000000
LogonTime             : 7/6/2018 5:02:35 PM
LogonID               : 0x00000000000003E7

LogonGUID             : [redacted]-031b-5b40-0000-0020e7030000
GUIDType              : LogonGUID
TruncatedMachineGuid  : [redacted]-0000-0000-0000-000000000000
LogonTime             : 7/6/2018 5:02:35 PM
LogonID               : 0x00000000000003E7
```

Configuration Auditing - Rationale

“Adversaries will be students of your configuration to learn how to bypass/blend in.” Casey Smith and Matt Graeber, BlueHat Israel 2017

Configuration Auditing

- sysmon.exe -c
- 

  # PSSysmonTools

  Sysmon Tools for PowerShell

  ## Implemented functions

  ### Get-SysmonConfiguration

  Parses a Sysmon driver configuration from the registry. Output is nearly identical to that of "sysmon.exe -c" but without the requirement to run sysmon.exe.

  - Parses binary ruleset from:
    - HKLM\SYSTEM\CurrentControlSet\Services\SysmonDrv\Parameters - Rules

3. Data Collection Implementation Analysis

What are the data sources?

How do defenders use the event fields?

Is collection comprehensive?

Bypassing Sysmon WmiEvents

Goal:

Identify a technique such that WMI persistence would never be logged.

Strategy:

Determine how WMI persistence logging is achieved.

Bypassing Sysmon WmiEvents

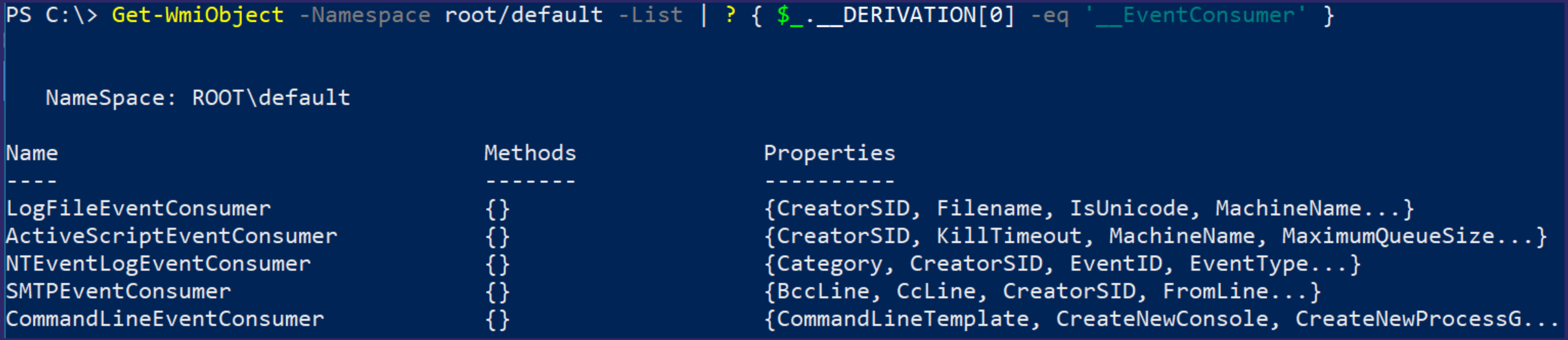
-
  ```
  SELECT * FROM __InstanceOperationEvent
  WITHIN 5 WHERE TargetInstance ISA
  '__EventConsumer' OR TargetInstance ISA
  '__EventFilter' OR TargetInstance ISA
  '__FilterToConsumerBinding'
  ```

- Only relevant to the root/subscription namespace

Bypassing Sysmon WmiEvents

Bypass #1

Persist in the root/default namespace.

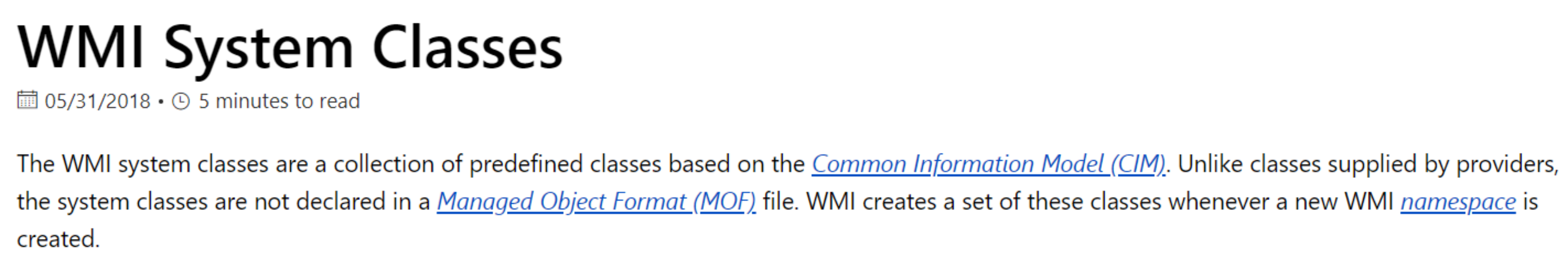


```
PS C:\> Get-WmiObject -Namespace root/default -List | ? { $_.__DERIVATION[0] -eq '__EventConsumer' }


   NameSpace: ROOT\default

Name                                Methods              Properties
----                                -------              ----------
LogFileEventConsumer                {}                   {CreatorSID, Filename, IsUnicode, MachineName...}
ActiveScriptEventConsumer           {}                   {CreatorSID, KillTimeout, MachineName, MaximumQueueSize...}
NTEventLogEventConsumer             {}                   {Category, CreatorSID, EventID, EventType...}
SMTPEventConsumer                   {}                   {BccLine, CcLine, CreatorSID, FromLine...}
CommandLineEventConsumer            {}                   {CommandLineTemplate, CreateNewConsole, CreateNewProcessG...
```

Cons: easy to fix

Bypassing Sysmon WmiEvents

Can we do better?



WMI System Classes

05/31/2018 • 5 minutes to read

The WMI system classes are a collection of predefined classes based on the *Common Information Model (CIM)*. Unlike classes supplied by providers, the system classes are not declared in a *Managed Object Format (MOF)* file. WMI creates a set of these classes whenever a new WMI *namespace* is created.

__EventFilter, __EventConsumer, and __FilterToConsumerBinding are built in to every namespace!

Goal: Figure out how to implement __EventConsumer classes in arbitrary namespaces.

Bypassing Sysmon WmiEvents

Goal: Figure out how to implement __EventConsumer classes in arbitrary namespaces.

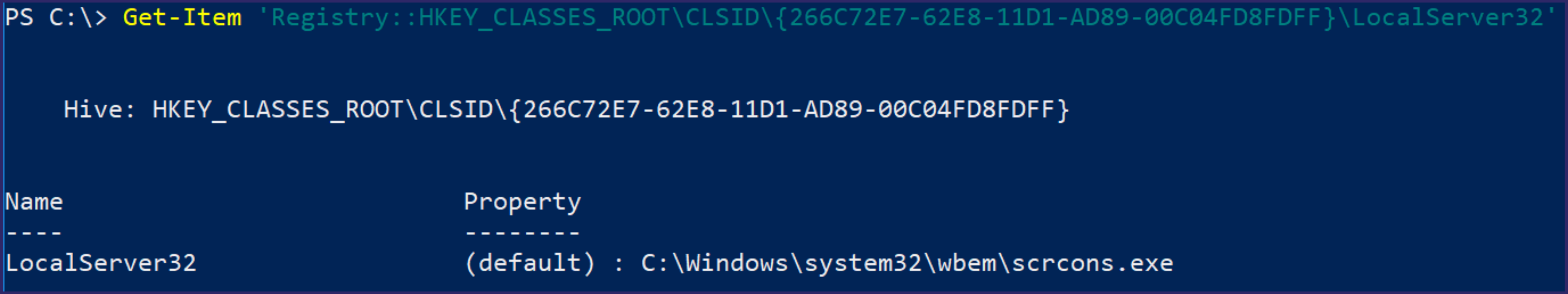
Strategy: Observe how they are implemented in root/subscription.

Bypassing Sysmon WmiEvents

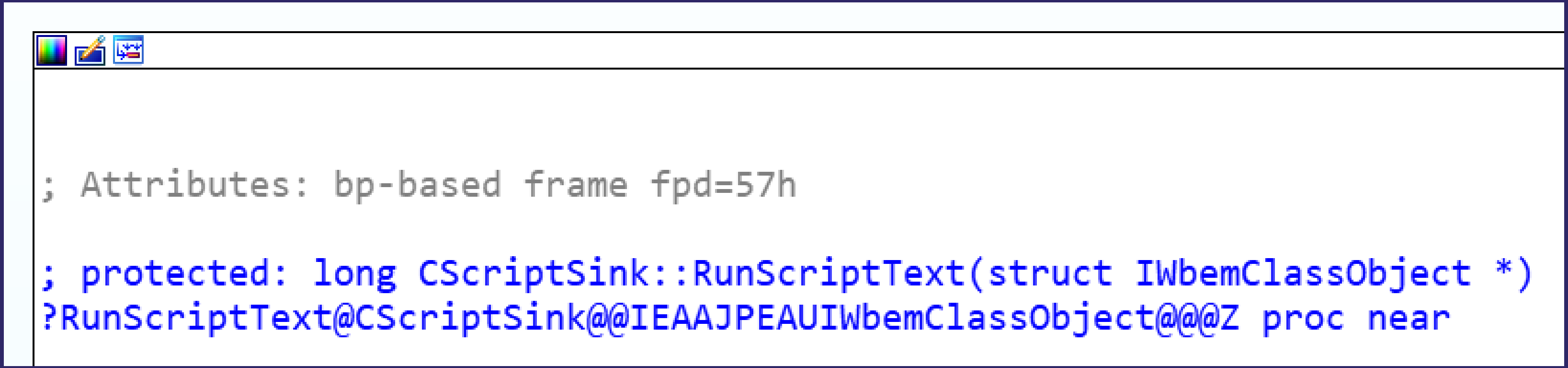
scrcons.mof:

```
class ActiveScriptEventConsumer : __EventConsumer {
  [key] string Name;
  [not_null, write] string ScriptingEngine;
  [write] string ScriptText;
  [write] string ScriptFilename;
  [write] uint32 KillTimeout = 0; };

Instance of __Win32Provider as $SCRCONS_P {
  Name = "ActiveScriptEventConsumer";
  Clsid = "{266c72e7-62e8-11d1-ad89-00c04fd8fdff}";
  PerUserInitialization = TRUE;
  HostingModel = "SelfHost"; };

Instance of __EventConsumerProviderRegistration {
  Provider = $SCRCONS_P;
  ConsumerClassNames = {"ActiveScriptEventConsumer"}; };
```

Bypassing Sysmon WmiEvents



```
PS C:\> Get-Item 'Registry::HKEY_CLASSES_ROOT\CLSID\{266C72E7-62E8-11D1-AD89-00C04FD8FDFF}\LocalServer32'


    Hive: HKEY_CLASSES_ROOT\CLSID\{266C72E7-62E8-11D1-AD89-00C04FD8FDFF}


Name                           Property
----                           --------
LocalServer32                  (default) : C:\Windows\system32\wbem\scrcons.exe
```



```
; Attributes: bp-based frame fpd=57h

; protected: long CScriptSink::RunScriptText(struct IWbemClassObject *)
?RunScriptText@CScriptSink@@IEAAJPEAUIWbemClassObject@@@Z proc near
```

Bypassing Sysmon WmiEvents

Weaponization Strategy:

Enable ActiveScriptEventConsumer and CommandLineEventConsumer class creation in any arbitrary namespace remotely.

https://gist.github.com/mattifestation/f38a79c7983208aa230030f61dfeb767

- New-ActiveScriptEventConsumerClass
- New-CommandLineEventConsumerClass

Class names can also be whatever you want. e.g. root/foo:DoNotDetectMeClass

4. Footprint/Attack Surface Analysis

What things get added to the host?

How does the tool behave?

What resources does the tool depend on?

Sysmon Installation

Update requires uninstall + install

Behavior varies for 32-bit and 64-bit binaries

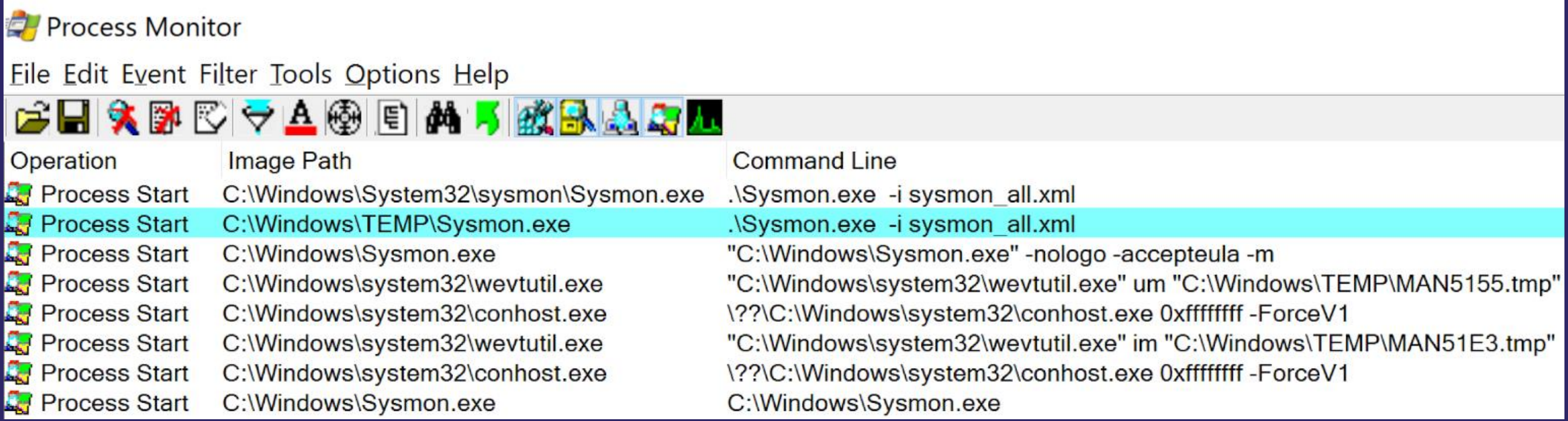
Added Components

- Files
  - C:\Windows\Sysmon.exe
  - C:\Windows\SysmonDrv.sys
- Services - Sysmon and SysmonDrv
- Registry Keys
  - HKLM\SYSTEM\CurrentControlSet\Services\Sysmon
  - HKLM\SYSTEM\CurrentControlSet\Services\SysmonDrv
  - HKLM\SYSTEM\CurrentControlSet\Services\SysmonDrv\Parameters
    - Only readable by admins because rules stored here
- ETW Provider
- Event Log

Installation - 32-bit Sysmon.exe on 64-bit system

64-bit installer extracted to %temp%

- DLL Hijacking
- Symlink redirection to exploit TOCTOU as well? (see James Forshaw’s work)

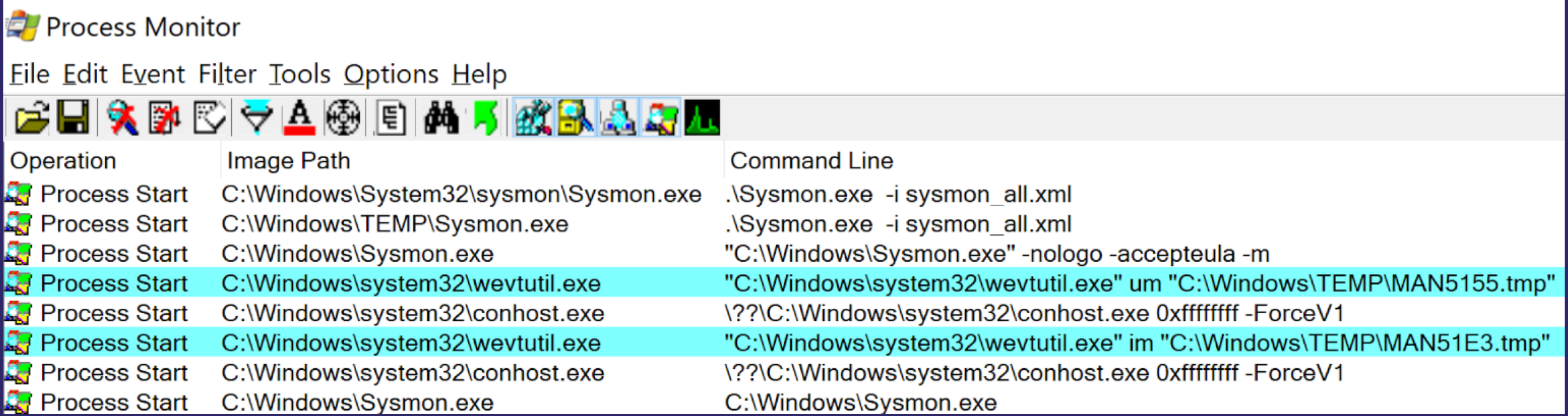


Operation	Image Path	Command Line
Process Start	C:\Windows\System32\sysmon\Sysmon.exe	.\Sysmon.exe -i sysmon_all.xml
Process Start	C:\Windows\TEMP\Sysmon.exe	.\Sysmon.exe -i sysmon_all.xml
Process Start	C:\Windows\Sysmon.exe	"C:\Windows\Sysmon.exe" -nologo -accepteula -m
Process Start	C:\Windows\system32\wevtutil.exe	"C:\Windows\system32\wevtutil.exe" um "C:\Windows\TEMP\MAN5155.tmp"
Process Start	C:\Windows\system32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
Process Start	C:\Windows\system32\wevtutil.exe	"C:\Windows\system32\wevtutil.exe" im "C:\Windows\TEMP\MAN51E3.tmp"
Process Start	C:\Windows\system32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
Process Start	C:\Windows\Sysmon.exe	C:\Windows\Sysmon.exe

Event Log Installation

Event log manifest copied to unique file at %TEMP%\MAN####.tmp

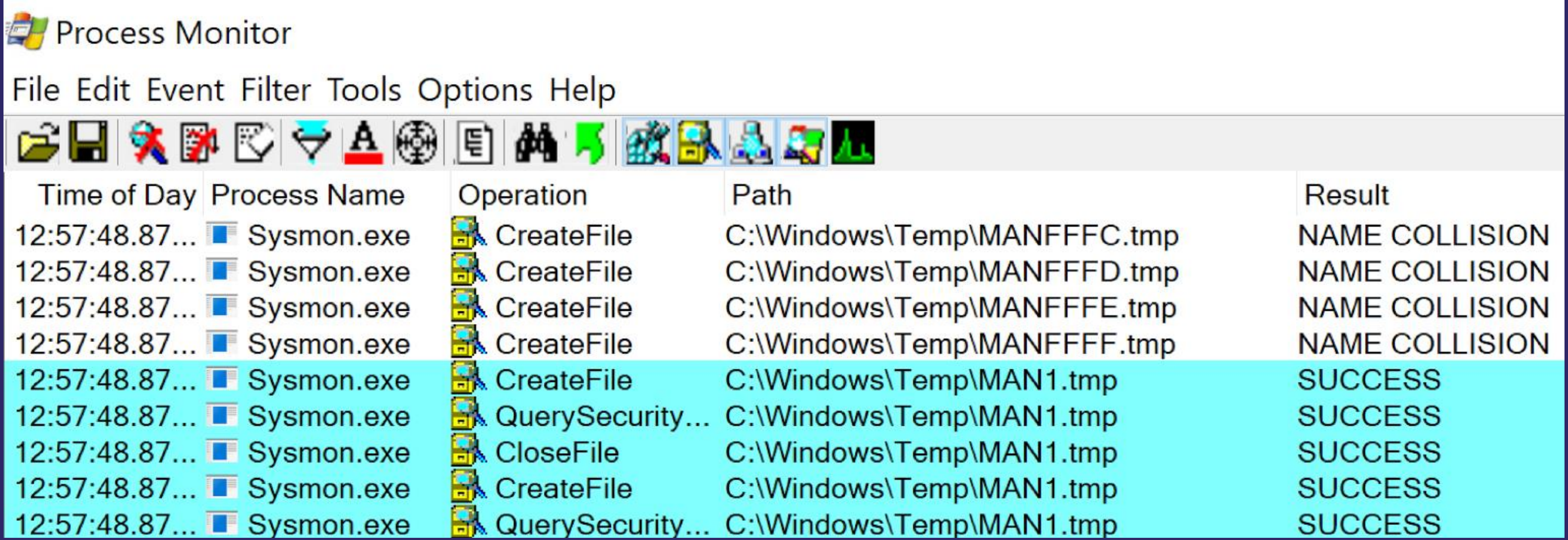
- #### = Alpha numeric characters



Operation	Image Path	Command Line
Process Start	C:\Windows\System32\sysmon\Sysmon.exe	.\Sysmon.exe -i sysmon_all.xml
Process Start	C:\Windows\TEMP\Sysmon.exe	.\Sysmon.exe -i sysmon_all.xml
Process Start	C:\Windows\Sysmon.exe	"C:\Windows\Sysmon.exe" -nologo -accepteula -m
Process Start	C:\Windows\system32\wevtutil.exe	"C:\Windows\system32\wevtutil.exe" um "C:\Windows\TEMP\MAN5155.tmp"
Process Start	C:\Windows\system32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
Process Start	C:\Windows\system32\wevtutil.exe	"C:\Windows\system32\wevtutil.exe" im "C:\Windows\TEMP\MAN51E3.tmp"
Process Start	C:\Windows\system32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
Process Start	C:\Windows\Sysmon.exe	C:\Windows\Sysmon.exe

Event Log Manifest Hijack

- Exhausting all MAN####.tmp options results in MAN1.tmp
- TOCOU - Strategy: use symlinks to hijack the manifest between when it was written and when wevtutil.exe uses it



Time of Day	Process Name	Operation	Path	Result
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MANFFFC.tmp	NAME COLLISION
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MANFFFD.tmp	NAME COLLISION
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MANFFFE.tmp	NAME COLLISION
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MANFFFF.tmp	NAME COLLISION
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MAN1.tmp	SUCCESS
12:57:48.87...	Sysmon.exe	QuerySecurity...	C:\Windows\Temp\MAN1.tmp	SUCCESS
12:57:48.87...	Sysmon.exe	CloseFile	C:\Windows\Temp\MAN1.tmp	SUCCESS
12:57:48.87...	Sysmon.exe	CreateFile	C:\Windows\Temp\MAN1.tmp	SUCCESS
12:57:48.87...	Sysmon.exe	QuerySecurity...	C:\Windows\Temp\MAN1.tmp	SUCCESS

Driver Analysis

- Minifilter driver with an altitude of 385201
- Need SeDebugPrivilege to interact with driver

IOCTLs

- 0x83400000 - Signal driver that a handle to it was obtained?
- 0x83400004 - Retrieves raw event information from event queue
  - Could potentially exhaust the queue
- 0x83400008 - Signals that it should consume the new registry rules config
  - Changing registry manually results in no event
- 0x8340000C - Retrieves raw ProcessCreate data for a PID

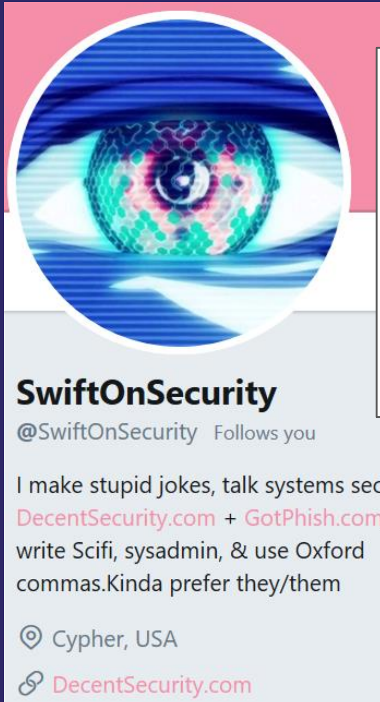
5. Configuration Analysis

How is it commonly configured?

Generic Rule Evasion Analysis

1. Include rules log potential evil.
2. Exclude rules filter out “noise”
3. A single exclude rule overrides all include rules.
4. We, as the attacker, want to be the “noise.”
5. If not feasible:
    a. Identify/develop generic bypasses
    b. Avoid certain actions (difficult in practice)

Configuration-specific Evasion Case Study



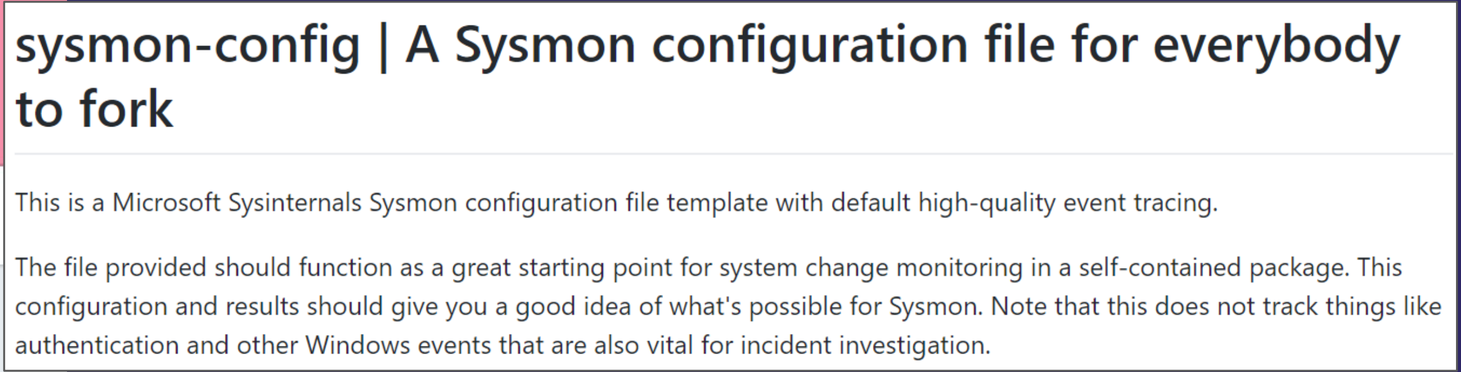
SwiftOnSecurity

@SwiftOnSecurity Follows you

I make stupid jokes, talk systems security, DecentSecurity.com + GotPhish.com, write Scifi, sysadmin, & use Oxford commas.Kinda prefer they/them

Cypher, USA

DecentSecurity.com



sysmon-config | A Sysmon configuration file for everybody to fork

This is a Microsoft Sysinternals Sysmon configuration file template with default high-quality event tracing.

The file provided should function as a great starting point for system change monitoring in a self-contained package. This configuration and results should give you a good idea of what's possible for Sysmon. Note that this does not track things like authentication and other Windows events that are also vital for incident investigation.

https://github.com/SwiftOnSecurity/sysmon-config

Configuration-specific Evasion Case Study

Evasion scenario:

- An admin left their Sysmon config XML on disk.
- An elevated attacker recovered the config from registry.
- Config pushed via GPO that can be read by any domain user

Plan of Attack:

1. Identify attacker-influenceable exclude rules for each rule type
2. Form a composition of evasions
3. Where rules cannot be outright evaded, identify:
    a. Alternative, generic bypass/evasion techniques
    b. Annotate actions that should be avoided.

ProcessCreate

Exclude Rule Evasion Candidates:

```
<CommandLine condition="contains">AcroRd32.exe" /CR </CommandLine>
<CommandLine condition="contains">AcroRd32.exe" --channel=</CommandLine>
```

Action:

- Include “AcroRd32.exe” strings in command-line invocations

Rationale:

- So long as the command line string contains this string anywhere, our malicious program will evade all ProcessCreate actions.

FileCreateTime

Exclude Rule Evasion Candidates:

```
<Image condition="image">OneDrive.exe</Image>
<Image condition="contains">setup</Image>
<Image condition="end with">redist.exe</Image>
```

Action:

- Drop to directory containing “setup” or name EXE “OneDrive.exe” or “redist.exe”

Rationale:

- All of these are attacker-controllable. The “contains” rules are likely ideal from an evasion perspective as they are more composable.

NetworkConnect

Exclude Rule Evasion Candidates:

```
<Image condition="image">OneDrive.exe</Image>
```

Action:

- Name malicious EXE “OneDrive.exe”

Rationale:

- This exclude rule is attacker-controllable. The downside is that the “image” attribute is not the most ideal for composability. One upside is that this exclude rule also resides in the FileCreateTime ruleset.

ProcessTerminate

Avoidance Rule:

```
<Image condition="begin with">C:\Users</Image>
```

Action:

- Avoid dropping your code to “C:\Users”.

Rationale:

- Since no exclude rules are present, we must resort to tradecraft avoidance.

DriverLoad

Exclude Rule Evasion Candidates:

```
<Signature condition="contains">microsoft</Signature>
<Signature condition="contains">windows</Signature>
<Signature condition="begin with">Intel </Signature>
```

Action:

- Our tradecraft is likely to avoid loading drivers anyway
- Signature rules potentially vulnerable to cert cloning attack. Test-signing required to load.

Rationale:

- An attacker controls the Subject field of the certificate that they use to sign their code.

CreateRemoteThread

Exclude Rule Evasion Candidates:

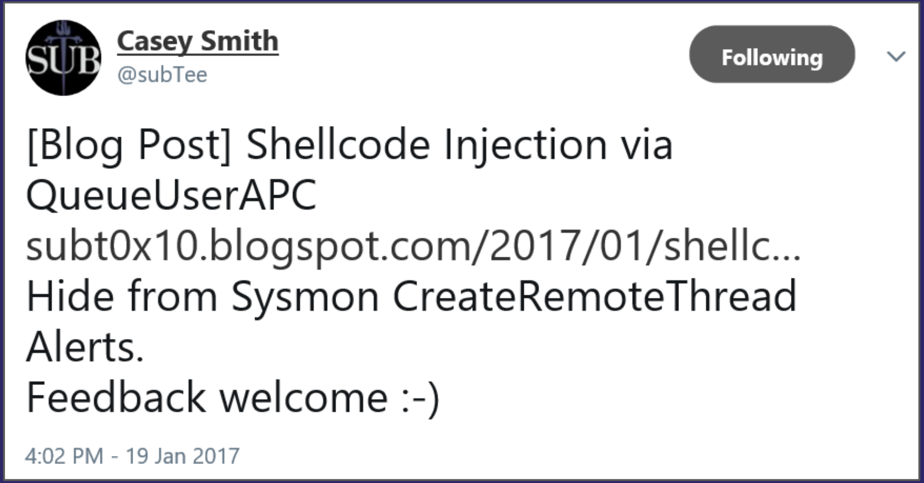
- None are ideal

Action:

- Employ generic evasion: QueueUserApc

Rationale:

- Exclude rules would require that you be running elevated. If process injection is required, use QueueUserApc



[Blog Post] Shellcode Injection via QueueUserAPC
subt0x10.blogspot.com/2017/01/shellc...
Hide from Sysmon CreateRemoteThread Alerts.
Feedback welcome :-)

4:02 PM - 19 Jan 2017

Bonus: Evasion research opportunities abusing the StartModule rule...

FileCreate

Action:

- Avoid dropping code with common executable file types.
- Rule is evaded if you change the extension after dropping the file.

Rationale:

- No ideal exclude rules are present, so we must resort to tradecraft avoidance.

RegistryEvent

Exclude Rule Evasion Candidates:

```
<Image condition="end with">Office\root\integration\integrator.exe</Image>
```

Action:

- Have the full path end in this string.

Possible Conflict:

- Previous evasion candidates were selected where the filename was “OneDrive.exe”. We will need to manage this conflict - e.g. an IPC mechanism

WmiEvent

Action:

- A generic evasion for WmiEvents exists. Avoid using the root/subscription namespace. Use root/default or ideally, a namespace created by the attacker.
- Ideally opt for ActiveScriptEventConsumer.

Rationale:

- root/subscription persistence is detected. CommandLineEventConsumer spawns a more detectable child process of wmiprvse.exe.

Overall Rule Composition

Evasion Candidates:

```
<CommandLine condition="contains">AcroRd32.exe" /CR </CommandLine>
<Image condition="contains">setup</Image>
<Image condition="image">OneDrive.exe</Image>
<Image condition="end with">Office\root\integration\integrator.exe</Image>
```

Resulting Evasive Invocation (e.g. PowerShell in disguise):

```
C:\setup\Office\root\integration\integrator.exe -
version 2 "& {iex(gc
C:\setup\Office\root\integration\integrator.txt);
$null = 'AcroRd32.exe" /CR '}"
```

Sysmon Conclusions

Tampering - Effective against a non-administrator

Attack Surface - Has some weaknesses(local privilege escalation), but fixable

Evasion

- Rule enumeration is critical for attacker
- Collection rules leave us wanting (exclude rules)

General

- Effective, but not a security solution - supplementary data source

Conclusions

- Attackers will continue to subvert products
- Vendors
  - Are you doing subversion research into your product?
    - Tampering?
    - Detection evasion?
    - Attack surface analysis?
- Defenders
  - Are you asking vendors the extent of their threat model?
  - Are you aware of the resilience of your security products?

Thank you!

Please donate to the Muscular Dystrophy Association!

https://www.mda.org/make-a-donation