Matthew Green

I am a DFIR and detection guy from Sydney Australia.

(sitemap)~$ type to search

Navigation

- » Home
- » About Me
- » Talks / Projects
- » XML Feed

Blue Team Hacks - Binary Rename

12 May 2019 » posts

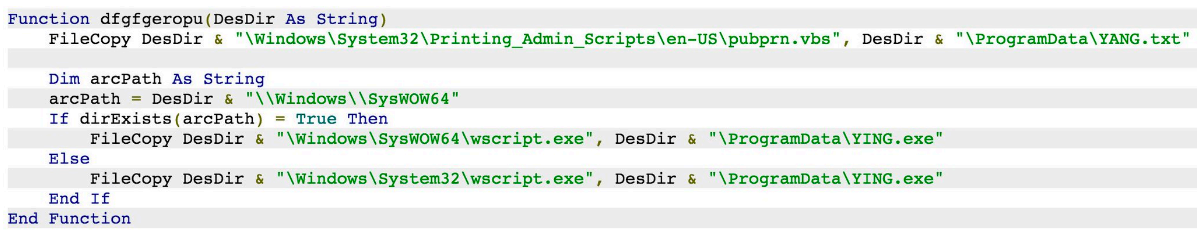
In this post I thought I would share an interesting proof of concept I developed to detect Binary Rename of commonly abused binaries. Im going to describe the detection, its limitations and share the code.

Background

Binary rename is a defence evasion technique used to bypass brittle process name and path based detections. Following the mantra of misdirection and hiding in plain sight, binary rename is a sub technique of T1036 - Masquerading in the Mitre ATT&CK framework. Binary rename can be observed in use across all stages of the attack lifecycle and is a technique used by a large selection of actors from commodity malware crews through to Nation States. One of the most well recognised use of the binary rename technique was NotPetya, a renamed psexec binary enabling the automated and devastating lateral infection.

In my current $dayjob I developed a query to hunt for this activity by reviewing an executed process' binary attributes and comparing with unexpected process names and paths. These have been some of my goto hunt queries pulling in hits for javascript based junkware, through to lateral movement, exfiltration tools and nation state level defence evasion.

For attacks sitting earlier in the attack lifecycle, often this involves an extension to the living off the land techniques copying a monitored binary to a less conspicuous path. For interactive attacks or attacks later in lifecycle, often an attacker will leverage a hack tool or administration binary not native to the environment but similarly "legitimate" looking to an unfamiliar eye. Understanding the types of binaries used, the PE attributes enables some interesting detection anchors and subsequent hunts.



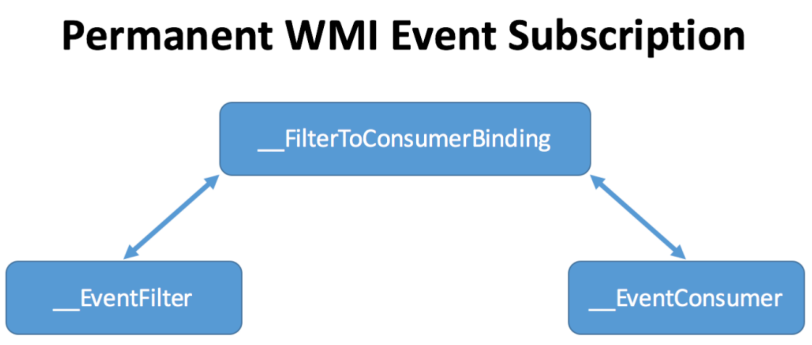
```
Function dfgfgeropu(DesDir As String)
    FileCopy DesDir & "\Windows\System32\Printing_Admin_Scripts\en-US\pubprn.vbs", DesDir & "\ProgramData\YANG.txt"

    Dim arcPath As String
    arcPath = DesDir & "\\Windows\\SysWOW64"
    If dirExists(arcPath) = True Then
        FileCopy DesDir & "\Windows\SysWOW64\wscript.exe", DesDir & "\ProgramData\YING.exe"
    Else
        FileCopy DesDir & "\Windows\System32\wscript.exe", DesDir & "\ProgramData\YING.exe"
    End If
End Function
```

VBA Macro example: https://twitter.com/ItsReallyNick/status/945682763486777345

With that in mind, not all security teams have a capable EDR solution (End Point Detection and Response) that enables binary attribute visibility at scale. Thinking about this problem led me to thinking about an open source solution available on a wide selection of machines.

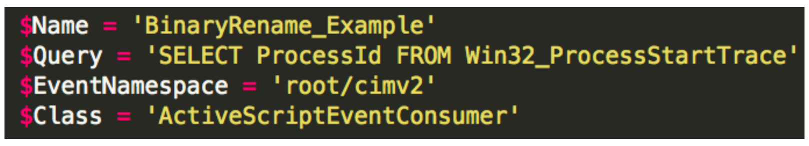
Solution

In absence of a mature logging or EDR, an interesting visability tool is WMI Eventing. A WMI event subscription is a method of subscribing to certain system events with a trigger (filter) and action (event consumer). WMI eventing can be used to action on almost any operating system event. For example - logon, process, registry or file activity.



WMI Eventing

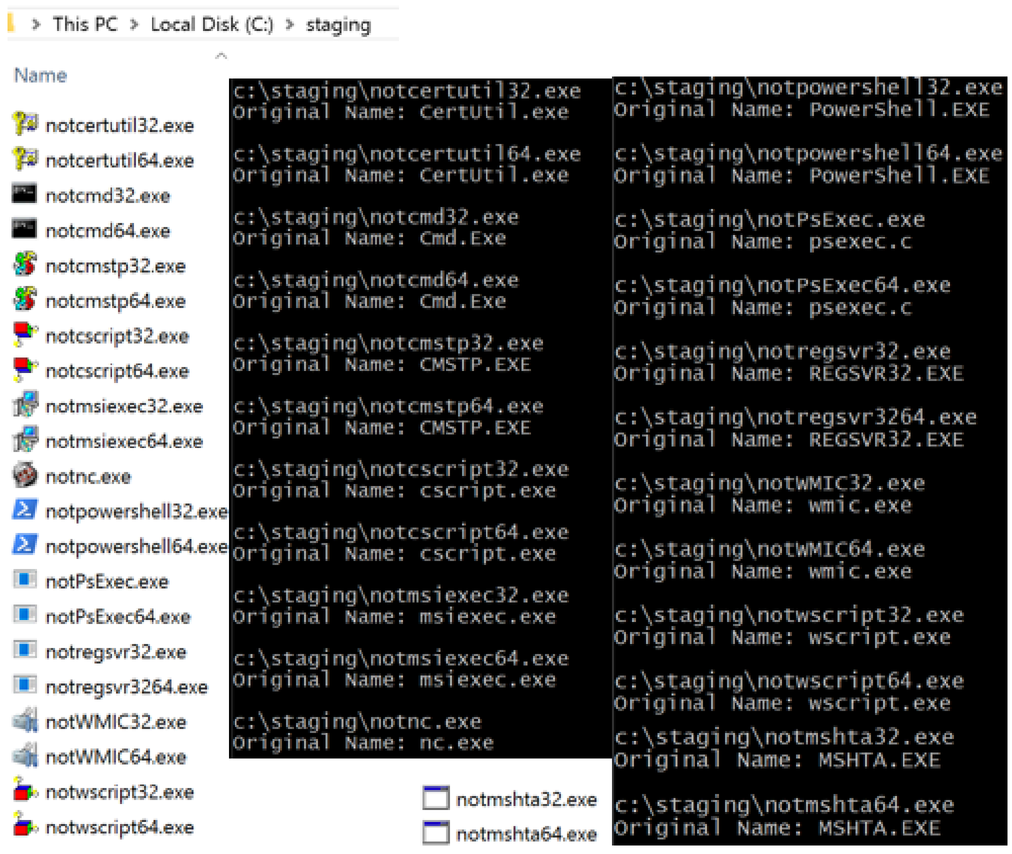
WMI is the Blue Team's equivalent to "living off the land" providing telemetry. This telemetry is similar to a limited version of modern EDR userland event tracing without the need to install a service or execute a binary directly. WMI Eventing is not new, Fireeye discussed the use of WMI as an endpoint intrusion detection system back in 2016. I have previously built WMI Eventing based solutions for a variety of niche IR use cases and visibility gaps. Although a complete description of WMI and WMI Event Consumers is outside the scope of this post, please see the further reading section for some detailed links and background in this space.



```
$Name = 'BinaryRename_Example'
$Query = 'SELECT ProcessId FROM Win32_ProcessStartTrace'
$EventNamespace = 'root/cimv2'
$Class = 'ActiveScriptEventConsumer'
```

WMI Filter Query

An ActiveScript Event Consumers allows the Blue Team to add logic and enrichment to WMI event triggers through the powerful Windows Scripting Host. In this POC I leverage a real time "extrinsic" wmi trigger for process execution monitoring, collecting ProcessID from all executed processes. A query of Win32_Process enables further process metadata to enable lookup of PE Attributes for detection. The PE Attribute in this use case is Original Name, with the Detection to lookup and alert against a list of high priority Original Names



This PC > Local Disk (C:) > staging

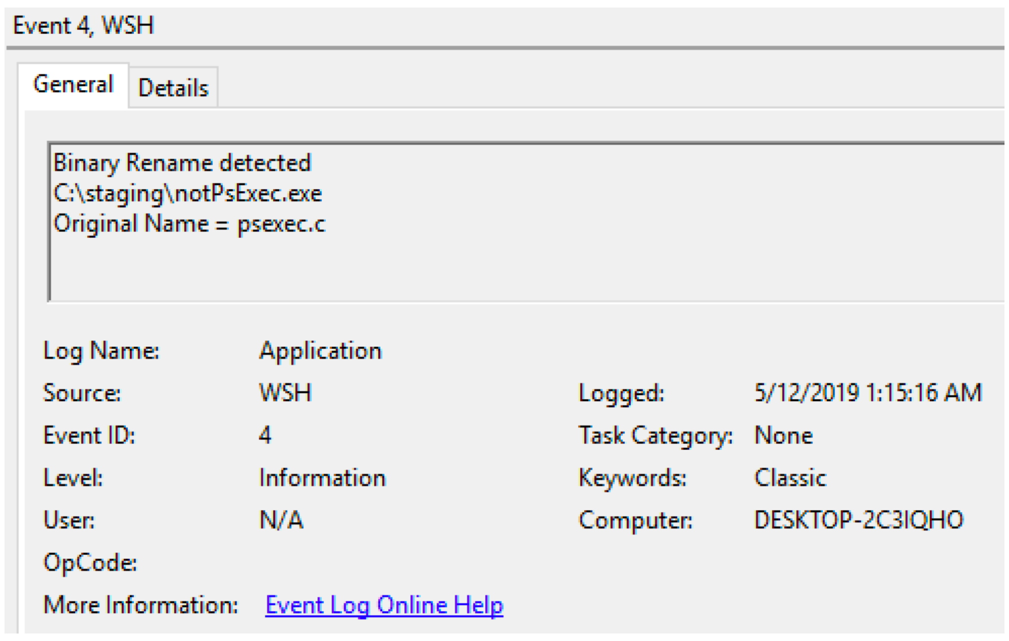
Name
notcertutil32.exe
notcertutil64.exe
notcmd32.exe
notcmd64.exe
notcmstp32.exe
notcmstp64.exe
notcscript32.exe
notcscript64.exe
notmsiexec32.exe
notmsiexec64.exe
notnc.exe
notpowershell32.exe
notpowershell64.exe
notPsExec.exe
notPsExec64.exe
notregsvr32.exe
notregsvr3264.exe
notWMIC32.exe
notWMIC64.exe
notwscript32.exe
notwscript64.exe
notmshta32.exe
notmshta64.exe

```
c:\staging\notcertutil32.exe
Original Name: CertUtil.exe

c:\staging\notcertutil64.exe
Original Name: CertUtil.exe

c:\staging\notcmd32.exe
Original Name: Cmd.Exe

c:\staging\notcmd64.exe
Original Name: Cmd.Exe

c:\staging\notcmstp32.exe
Original Name: CMSTP.EXE

c:\staging\notcmstp64.exe
Original Name: CMSTP.EXE

c:\staging\notcscript32.exe
Original Name: cscript.exe

c:\staging\notcscript64.exe
Original Name: cscript.exe

c:\staging\notmsiexec32.exe
Original Name: msiexec.exe

c:\staging\notmsiexec64.exe
Original Name: msiexec.exe

c:\staging\notnc.exe
Original Name: nc.exe
```

```
c:\staging\notpowershell32.exe
Original Name: PowerShell.EXE

c:\staging\notpowershell64.exe
Original Name: PowerShell.EXE

c:\staging\notPsExec.exe
Original Name: psexec.c

c:\staging\notPsExec64.exe
Original Name: psexec.c

c:\staging\notregsvr32.exe
Original Name: REGSVR32.EXE

c:\staging\notregsvr3264.exe
Original Name: REGSVR32.EXE

c:\staging\notWMIC32.exe
Original Name: wmic.exe

c:\staging\notWMIC64.exe
Original Name: wmic.exe

c:\staging\notwscript32.exe
Original Name: wscript.exe

c:\staging\notwscript64.exe
Original Name: wscript.exe
c:\staging\notmshta32.exe
Original Name: MSHTA.EXE

c:\staging\notmshta64.exe
Original Name: MSHTA.EXE
```

Binaries targetted: Original Name

On condition match the POC has the capability to write to the Application event log. Event ID 4, with relevant alert details. The decision to leave out a hash calculation was decided for performance reasons, process path and Original Name providing a lead for live response.

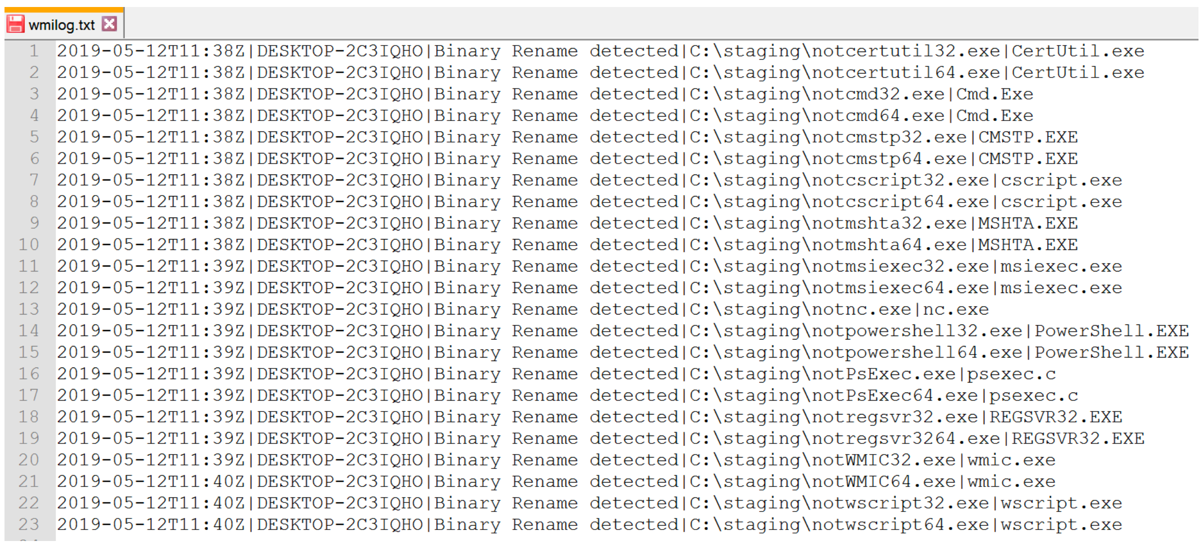


Binary Rename detected
C:\staging\notPsExec.exe
Original Name = psexec.c

Log Name:	Application		
Source:	WSH	Logged:	5/12/2019 1:15:16 AM
Event ID:	4	Task Category:	None
Level:	Information	Keywords:	Classic
User:	N/A	Computer:	DESKTOP-2C3IQHO
OpCode:			
More Information:	Event Log Online Help		

Generated EventLog

Similarly, output to a logfile for data ingestion. It is worthy to note: the POC can easily be modifed to suit requirements, removing the write to file or event log functions and function calls.



```
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcertutil32.exe|CertUtil.exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcertutil64.exe|CertUtil.exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcmd32.exe|Cmd.Exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcmd64.exe|Cmd.Exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcmstp32.exe|CMSTP.EXE
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcmstp64.exe|CMSTP.EXE
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcscript32.exe|cscript.exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notcscript64.exe|cscript.exe
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notmshta32.exe|MSHTA.EXE
2019-05-12T11:38Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notmshta64.exe|MSHTA.EXE
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notmsiexec32.exe|msiexec.exe
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notmsiexec64.exe|msiexec.exe
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notnc.exe|nc.exe
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notpowershell32.exe|PowerShell.EXE
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notpowershell64.exe|PowerShell.EXE
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notPsExec.exe|psexec.c
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notPsExec64.exe|psexec.c
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notregsvr32.exe|REGSVR32.EXE
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notregsvr3264.exe|REGSVR32.EXE
2019-05-12T11:39Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notWMIC32.exe|wmic.exe
2019-05-12T11:40Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notWMIC64.exe|wmic.exe
2019-05-12T11:40Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notwscript32.exe|wscript.exe
2019-05-12T11:40Z|DESKTOP-2C3IQHO|Binary Rename detected|C:\staging\notwscript64.exe|wscript.exe
```

Generated Log File

Limitations

One of the limitations of leveraging WMI Eventing as an event source is events typically do not hold all the appropriate data for a mature detection use case. To enrich the detection, we require to query the Win32_Process class. There is a slight delay in obtaining process metadata so very short-lived processes (fraction of a second) may cause missed results. In my testing, these very short-lived commands like renamed cmd: `cdm /c echo <string>` failed to generate wmi data however a slight pause during a local ping recorded the event. For the use case of a download cradle, access shell or other interactive commands typically of interest I do not foresee an issue but please keep this limitation in mind.

A second limitation is performance. Although not particularly resource intensive in my testing (no visible resource utilisation), in production there may be unexpected constraints. I have filtered my Process Events with this in mind but testing is recommended. I have also specifically kept the binary attribute matching use case very simple with a eye on performance. This may mean less fidelity in alerting, however the POC is fairly simple to modify and add capabilities.

It is also worthy to note: in some environments there may be legitimate binary rename activity for some of the targeted Original Names. Some of the binaries listed may require some tweaking of the matching logic to match host environment detection tolerance.

Finally management, WMI Event consumers are notoriously hard to manage. I have included a Powershell installation script with uninstall instructions to support Powershell 2.0 and above.

Final Thoughts

This has been a fun short project working on an open source detection capability. I was pleasantly surprised when I discovered vbscript has the ability query PE attributes. I hope others may find it useful, feel free to reach out if you have any feedback, questions, or improvements.

The POC template can be found here - WMIEvent-BinaryRename.ps1

Further reading

1. Ballenthin,William. Graeber, Matt. Teodorescu Claudiu. Windows Management Instrumentation (WMI) Offense, Defense, and Forensics, 2015
2. Green, Matthew. Blue Team Hacks - WMI Eventing, 2017
3. The MITRE Corporation. Technique: Masquerading - MITRE ATT&CK™
4. MSDN. Win32_ProcessStartTrace class
5. MSDN. System.OriginalFileName
6. Parisi, Timothy. Pena, Evan. WMI vs. WMI: Monitoring for Malicious Activity, 2016

Share by: X Post

Related Posts

- WMI Event Consumers: what are you missing? (Categories: posts)
- Cobalt Strike Payload Discovery And Data Manipulation In VQL (Categories: posts)
- Windows IPSEC for endpoint quarantine (Categories: posts)
- Local Live Response with Velociraptor ++ (Categories: posts)
- Live response automation with Velociraptor (Categories: posts)
- O365: Hidden InboxRules (Categories: posts)

« Live Response Script Builder　　　　Binary Rename 2 »