SANS ISC: InfoSec Handlers Diary Blog - SANS Internet Storm Center SANS Site Network Current Site SANS Internet Storm Center Other SANS Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government OnSite Training InfoSec Handlers Diary Blog

isc.sans.edu/diary/rss/27036

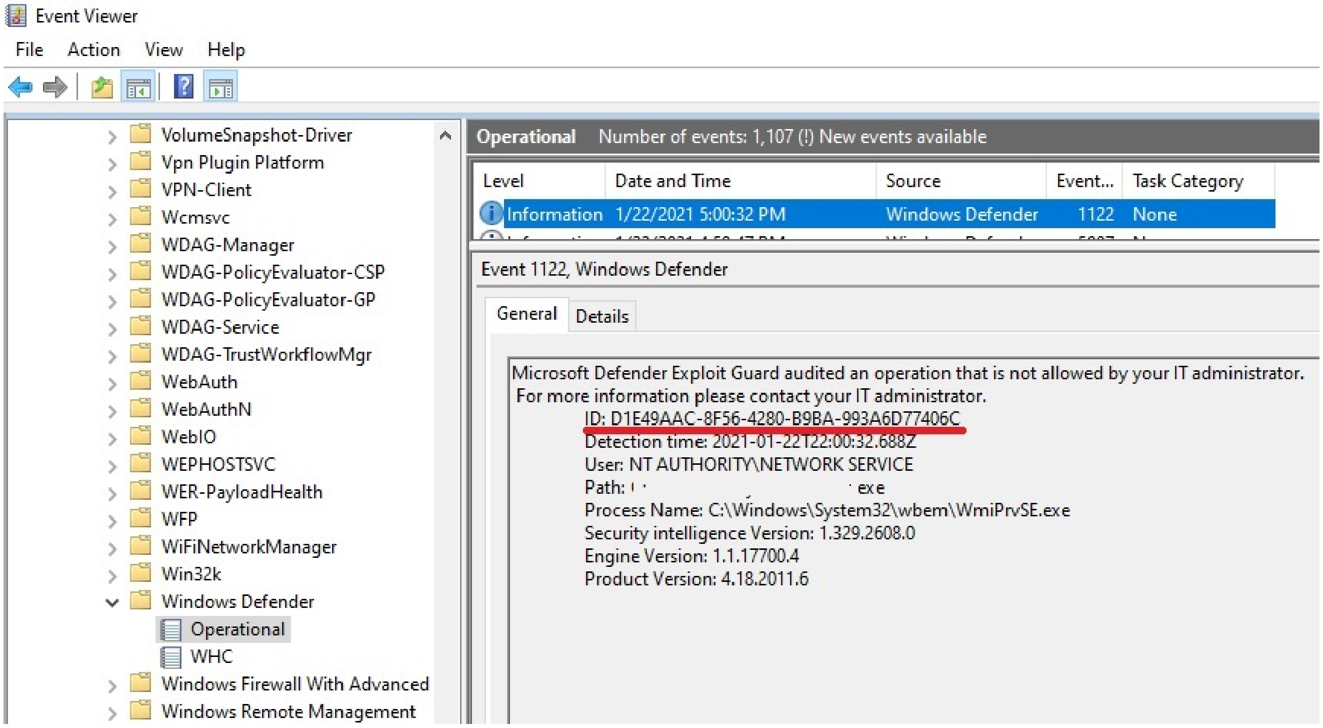
Emotet vs. Windows Attack Surface Reduction

Published: 2021-01-28
Last Updated: 2021-01-28 00:02:31 UTC
by Daniel Wesemann (Version: 1)
0 comment(s)
Emotet malware in the form of malicious Word documents continued to make the rounds over the past weeks, and the samples initially often had pretty poor anti-virus coverage (Virustotal) .The encoding used by the maldoc is very similar to what Didier Stevens analyzed in his recent diary, and the same method can be used to extract the mal-code from the current Emotet docs.

With the de-obfuscation reasonably straight forward, I proceeded to look into how the malware crooks accomplish execution from within the Word doc, and in particular, why Microsoft's "Attack Surface Reduction Rules" do not seem to help much.

But first, let's take a quick detour into what Attack Surface Reduction (ASR) promises to do on modern Windows devices. ASR is a somewhat clunky set of additional protections in Microsoft Defender Antivirus that can be turned on to log or intercept (block) some common attack scenarios. Microsoft's web site offers meager documentation, including a marginally helpful list of GUIDs that can be used to activate the feature.

One rule, "Block all Office Applications from creating child processes" (GUID D4F940AB-401B-4EFC-AADC-AD5F3C50688A) is supposed to prevent a Word document from launching any other task. Therefore, when this rule is configured, we would expect that the current Emotet and its execution chain of Word-Doc -> cmd.exe -> Powershell should not be successful. But it is.
Closer inspection of the Defender Event Log gives a hint "why":



Operational Number of events: 1,107 (!) New events available

Level	Date and Time	Source	Event...	Task Category
Information	1/22/2021 5:00:32 PM	Windows Defender	1122	None

Event 1122, Windows Defender

```
Microsoft Defender Exploit Guard audited an operation that is not allowed by your IT administrator.
 For more information please contact your IT administrator.
        ID: D1E49AAC-8F56-4280-B9BA-993A6D77406C
        Detection time: 2021-01-22T22:00:32.688Z
        User: NT AUTHORITY\NETWORK SERVICE
        Path: [redacted] exe
        Process Name: C:\Windows\System32\wbem\WmiPrvSE.exe
        Security intelligence Version: 1.329.2608.0
        Engine Version: 1.1.17700.4
        Product Version: 4.18.2011.6
```

The only ASR rule that we see firing when the Emotet Doc is being opened is the one with ID d1e49aac-8f56-4280-b9ba-993a6d77406c, corresponding to "Block process creations originating from PSExec and WMI commands". Yes, the Emotet VBA Macro is using a WMI (windows management instrumentation) call to launch the subsequent attack code. For such WMI invocation via the Win32 Process class, the parent process of "cmd" ends up being WmiPrvSe.exe, which in turn is launched from "svchost". Therefore, "cmd" is not a child process of Word, and the ASR block rule to prevent child processes of Word consequently doesn't trigger. Bah!

In corporate environments, remote management of user devices often uses tools like SCCM or Endpoint Manager, which in turn rely on WMI to function. Therefore, setting the ASR Rule for WMI/PSExec to "block" will likely break device management, and cause a huge mess. Chances are, the Emotet crooks were fully aware of this, and that's exactly why they chose this particular execution method for their attack code.

If you have Microsoft ATP, you can also use a hunting rule like this to search for WMI process creation

```
DeviceEvents
| where ActionType == "ProcessCreatedUsingWmiQuery"
| project Timestamp, DeviceName, ActionType, FileName, SHA1, FolderPath,
InitiatingProcessCommandLine, ProcessCommandLine
| sort by Timestamp desc
```

You might have to add a couple of exclusions to cover your management instrumentation or software distribution tools, but with a bit of tuning, you should see any current Emotet-like WMI attempts in your environment. The ProcessCommandLine in these cases will be long

(>600chars) and contain Base64 encoded Powershell, and the InitiatingProcess is Winword.

In the meantime, the probably best bet to protect your Windows users against Emotet and similar malware remains to quarantine password protected zips or Office documents with macros on your email gateway, or to disable macros within Office outright if you can get away with it.

Maybe, in a decade or three, Microsoft will get to the point where malware introduced via Office documents really no longer is a concern and prevalent problem. Until then, I guess we have to kinda hope that today's international raid by law enforcement against the Emotet gang really got the right guys, and got them good.

Keywords: Emotet WMI word

0 comment(s)

DEV522 Defending Web Application Security Essentials
LEARN MORE
Learn to defend your apps before they're hacked

Top of page

×

Diary Archives