RedLotus Compiled Knowledge.

Four screenshare handbooks. One complete reading collection.

279 original pages, with their illustrations, tables, technical examples and author credits preserved.

Compiled for the Bergsson Forensics library. Source material by the RedLotus authors; original attribution remains on the source pages.

A note before you begin.

This collection brings together the four PDFs supplied to Bergsson Forensics. It preserves their original order and page designs rather than rewriting their advice or removing duplicated material.

Use the contents page or PDF bookmarks to jump to a volume. The web reader includes page search, a readable text view, original-page links and optional synthetic narration.

Original screenshots and diagrams are retained in this PDF. Search and narration use the available text layer; words embedded in pictures, unusual typography and tables may not read accurately. Refer to the original page whenever a detail matters.

References to rules, ranks, organizational policies or tool instructions are statements from the source authors. They are not Bergsson Forensics site policies. The material is provided as historical reference and should be evaluated in its original context.

Credits: RedLotus and the authors named in the source documents, including ItzIceHere, Omqr, StrqfeToggled and UnMonsieur. Compilation and reading interface: Bergsson Forensics.

Edition assembled 13 October 2026. The original source filenames are listed on the following page.

The four volumes.

01 / Introduction to screensharing

16 original pages / collection pages 5-20

RedLotus Introductory Guide to SS_ENG (1).pdf

02 / Public screenshare guide

98 original pages / collection pages 22-119

[RedLotus] - Public SS Guide (FULL Release) (5).pdf

03 / Screenshare Handbook - Tome II

118 original pages / collection pages 121-238

[FULL] [RL] SS Handbook Tome II (5).pdf

04 / Tome II - Book II

47 original pages / collection pages 240-286

[RL] SS Handbook Tome II, Book II (4).pdf

01

Introduction to screensharing

16 original pages / reproduced in full

RedLotus Introductory Guide to SS_ENG (1).pdf

Original source content begins on the next page. Author credits and illustrations are preserved.



This guide is designed to provide a clear and accessible overview of essential SS knowledge. Created by the RedLotus team, it is an ideal starting point for those who wish to delve into more advanced methods at a later date.

Why is this guide important?

Ensuring a solid and consistent understanding of the topics covered is essential for promoting effective learning. This document provides a structured foundation that prepares readers to tackle more complex content, whether through webinars or in the forums of our community.

How did this guide come about?

The guide originates from a series of videos I, ItzIceHere, published on YouTube, initially intended for the Italian community. The videos, recorded in Italian, offer detailed explanations of the methods presented, but creating versions in other languages would have required a significant effort.

For this reason, I have chosen to create this written guide, which can be used either as an alternative or as a complement to the videos. Thanks to the automatically generated subtitles on YouTube, the videos remain accessible even to those who do not speak Italian, but this guide allows for a more precise following of the visually presented concepts, providing a clear and structured reference.

The project was also made possible thanks to the contribution of Omqr and StrqfeToggled, who transcribed the video content. I then reorganized and refined the material to create the document you are reading

The added value of video support?

Purely theoretical learning can be demanding and not very engaging. The videos represent an indispensable support, thanks to practical demonstrations that facilitate both understanding and memorization.

With this guide, I wanted to expand the educational experience offered by the videos, providing a tool that allows anyone, regardless of language, to follow the lessons with greater attention. The combination of visual explanations and a well-organized textual reference offers a complete approach, ideal for effective learning

RedLotus Community
Video Playlist

Terminology and Level of Detail Warning

It is important to underline that the terminology and explanations provided within this document have been deliberately simplified, avoiding excessive technicalities and an excessively in-depth level of detail. This choice was dictated by the desire to make the material accessible to a wide audience, avoiding overloading the reader with excessively complex or specific information. The primary objective is to provide a clear and essential overview of the concepts covered, allowing an effective understanding of the methods illustrated.

Introduction to ScreenSharing [SS]

Many companies that develop non-competitive video games avoid adopting client-side AntiCheat systems, mainly due to high costs and technical complexity. However, in competitive contexts such as Minecraft or FiveM servers, ScreenShares are essential to ensure a fair and cheater-free gaming environment. Overtime, the methods used to perform these checks have been refined to counteract increasingly advanced bypasses, with the main goal of gathering reliable evidence of cheat use by players.

In this guide, we will focus on basic methods to detect hacks, avoiding advanced techniques.

How to Check Forge Mods

During ScreenShares, it is important to be aware that some players may try to bypass using cheats hidden inside seemingly normal mods. For this reason, an analysis of the mods used by the player is essential.

Step 1: Check Modified Date of Mod Folder

Before examining individual mods, it is essential to check the last modified date of the folder containing the player's mods. If this date coincides with a moment immediately before the check, it is advisable to proceed with the player's ban. This action is necessary because it indicates possible tampering with the mods, which occurred shortly before the check.

Step 2: Analyze Mod Weight

A quick way to spot anomalies is to check the weight (size) of the mod files. Legitimate mods usually have a weight that falls within a range considered normal. If a mod's weight seems suspicious, because it is too large or too small compared to the standard, a more thorough investigation is needed.

Mods Weight

Step 3: Deep Analysis with a Java Decompiler

If the weight of a mod raises concerns, the next step is to use a Java decompiler. This tool allows you to analyze the source code of the mod and determine if it contains elements that should not be there, such as unauthorized functions. In this way, you can understand with certainty whether the mod is legitimate or not..

How to Access the .minecraft Folder

To begin the check, you need to locate the .minecraft folder, where all the mods are located installed by the player. To do this, follow these simple steps:

Go to the game Options, select the Resource Packages item, click on the "Open Packages Folder" button.

In the folder that opens, go back to the previous folder (the one called .minecraft). From here look for the "mods" folder.

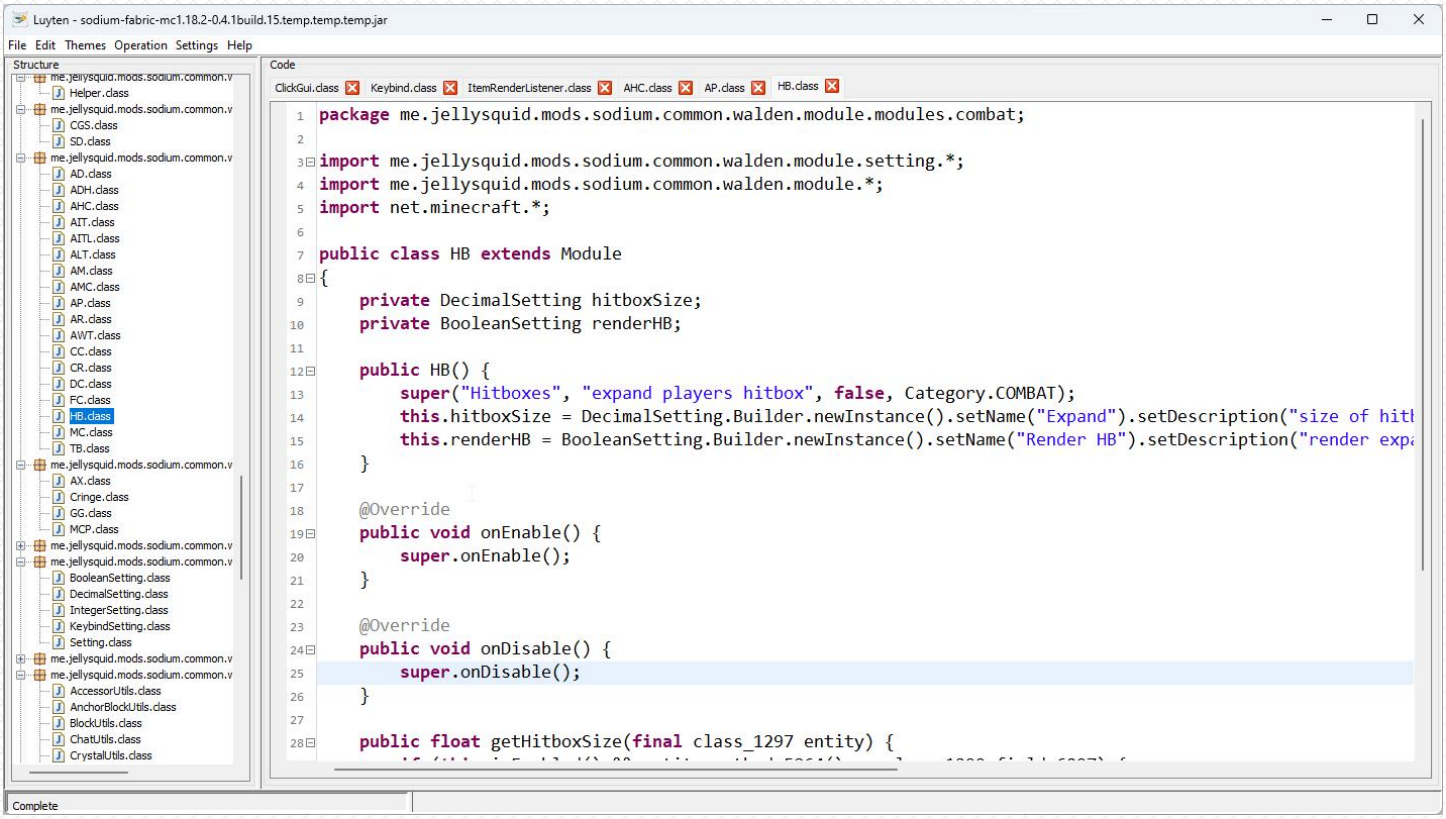
Luyten

Luyten is a Java decompiler that allows you to view the source code of a .jar file, such as those of mods, to verify whether they contain unauthorized code or cheats.

Steps to use Luyten:

Start by opening the Luyten program on the computer, then locate the mod file you want to analyze (the .jar file) and drag it into the Luyten window. Once the mod is loaded, Luyten will display the decompiled source code. Carefully examine the different classes to identify any suspicious elements, such as unnecessary code or unexpected modifications that could indicate the presence of illegitimate functionality.

Luyten download



RedLotus Mod Analyzer

It is a tool created by Red Lotus, called ModAnalyzer, that simplifies the analysis of Minecraft mods. This tool is useful for detecting any suspicious modifications or automatically hidden features.

Video: For an overview of how the tool works, a demo video is available.
ModAnalyzer download

Unloaded Mod

A method that some players use to bypass is to "unload" a mod. This means stopping Minecraft from using it, allowing you to delete, move, or rename it even if it was in use during the game session. To find these mods, you need to follow a specific procedure:

Start the "System Informer" tool on the computer, locate the process called "javaw.exe" in the process list, use the program's filter function. Set a "Contains (Case-Insensitive)" filter, and paste the path to the folder containing the player's mods. Compare the results obtained from the filter with the list of mods actually present in the player's mod folder. If you notice any differences, i.e. mods appearing in the process but not in the folder, the player may have moved, deleted, or renamed the mod after unloading it.

System Informer

System Informer is an advanced tool, developed by Winsiderss, designed to monitor system resources, debug software and detect malware. In a ScreenShare context during hack checks, it is mainly used to detect traces of cheats by analyzing the memory of processes and services.

Key Features of System Informer:
Allows you to observe the real-time usage of system resources.
Provides tools to analyze the operation of programs.
Helps identify suspicious activity related to malicious software.
Allows you to examine the memory of processes and services to detect anomalies.

Processes and Services Monitored:
During a ScreenShare, several processes and services are monitored, including: explorer, csrss, MsMpEng, SearchIndexer, Javaw, PcaSvc, Diagtrack and many others.

System Informer Configuration:
Download System Informer from the official website, start the program with administrator privileges, access the Options menu, select General, check the Kernel-Mode Driver option, finally restart the program. This option also allows you to view protected processes such as csrss and MsMpEng.

Process Filtering:

Find the process you want to analyze (eg: explorer.exe) and right-click on it, select Properties, then the Memory tab, click on Options and check Hide free pages and Hide reserved pages, set a minimum value of 5, select Extended Unicode and Mapped, then click on Ok. Once you have done this, a Results window will open. Click on Filter and choose the contains (case-insensitive) or Regex (case-insensitive) option depending on the type of search you want to perform:
Contains (case-insensitive): Filters for a specific keyword within the process (eg: "autoclicker").
Regex (case-insensitive): Allows you to perform more complex searches based on "search patterns" (eg: specific patterns).

Journal

The Journal is a log of system file activity that Windows keeps in the hidden and protected folder "$Extend$UsnJrnl", in a specific data stream called $J. This log tracks changes, creations, deletions and other operations on system files. Journal analysis can reveal suspicious activity performed by the player.
There are three main methods used in SS to "view" the Journal, each with its own characteristics and advantages:

CMD (Prompt dei Comandi)

Open Command Prompt as administrator and use the following command to export the Journal to a text file:

fsutil usn readjournal C: csv > Journal.txt

This command saves the entire Journal on the C: drive to a file called "Journal.txt".
To get more specific results, you can modify the command. For example, to display only logs containing ".exe", use:

fsutil usn readjournal C: csv | findstr /i /C:".exe" > Exe.txt

This command exports to "Exe.txt" only the logs that include the keyword ".exe".
You can modify the filter to search for other file extensions, keywords or to perform increasingly specific filters depending on your needs.

Echo Journal

Echo-Journal is a tool developed to view the Journal quickly and easily. Once opened, simply select the boxes for the operations you are interested in (Delete, Create, Renamed from, Renamed to, Closed and Data Changed).

Echo Journal Download

Journal Trace

Journal Trace is a tool developed by Ponei to view the Journal more efficiently.

To use it you can follow the steps below:
Open Journal Trace with administrator permissions, click on "Drive" and then on "Select" to choose the disk you want to analyze, after selecting the disk, click on "Drive" again and choose "Scan" to start scanning the Journal, once the upload is complete, go to "Layout" and select "Data Grid", click on the "Date" header to sort the results by the most recent date. Change the "Name" or "Reason" header and enter the term you want to search for (for example: "Delete", "Rename", "Data Extend").
JournalTrace download

BAM

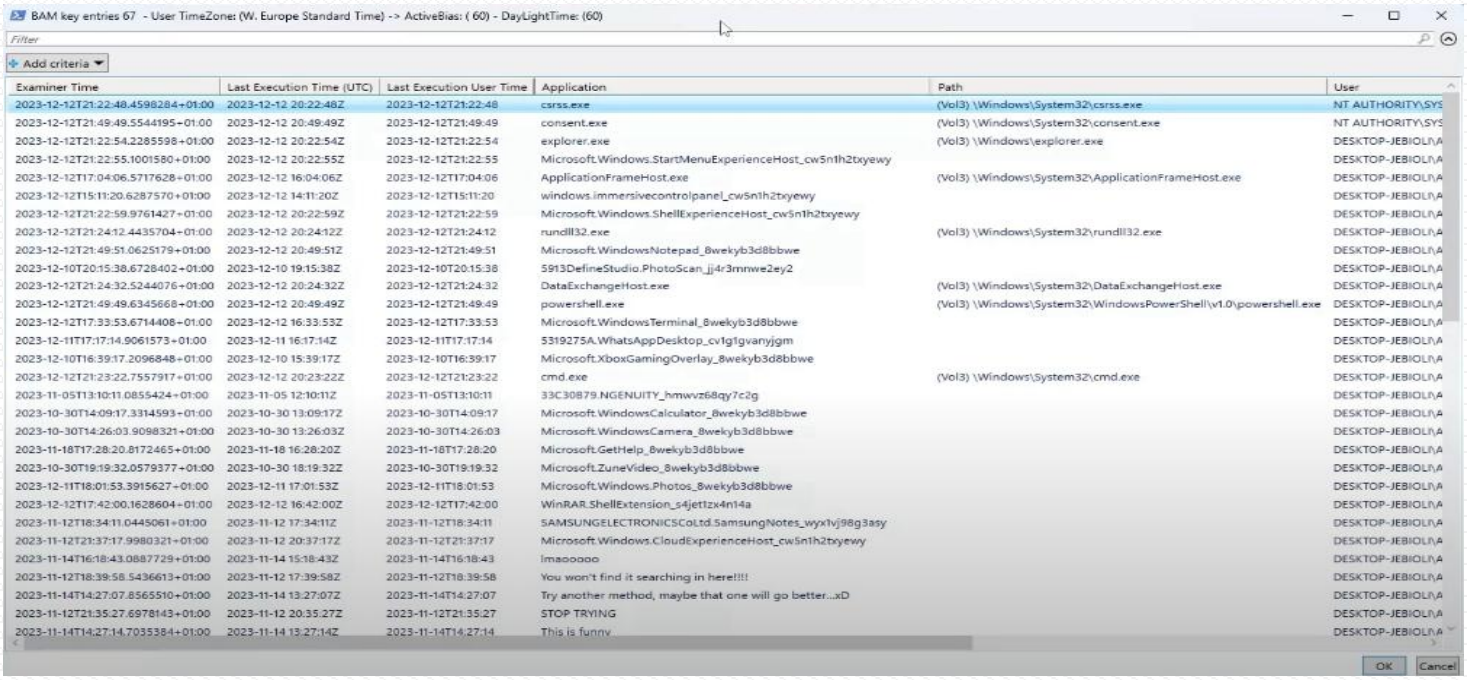
The BAM, or Background Activity Monitor, is a Windows driver introduced starting with Windows 10. Its function is to manage the background activities of applications. When an application is executed, the BAM creates a specific value in the registry (Regedit) that includes important information about the execution of the program. This information can be valuable for identifying suspicious activities or traces of cheats.

To view the BAM, a command developed by RedLotus is usually used, this is because it offers the possibility of checking the status of the digital signature of the file.

How to view the BAM

Start the Command Prompt with administrator privileges, paste the following command in the Command Prompt. A window will then open. By double-clicking on "Last Execution User Time", you will be able to view the date and time of the last "activity" that occurred on the program.

As soon as a program is opened, the date in the BAM will indicate the opening time, subsequently every time an activity is done on the program window, this date will be updated, finally, if the program is closed the date will update one last time.



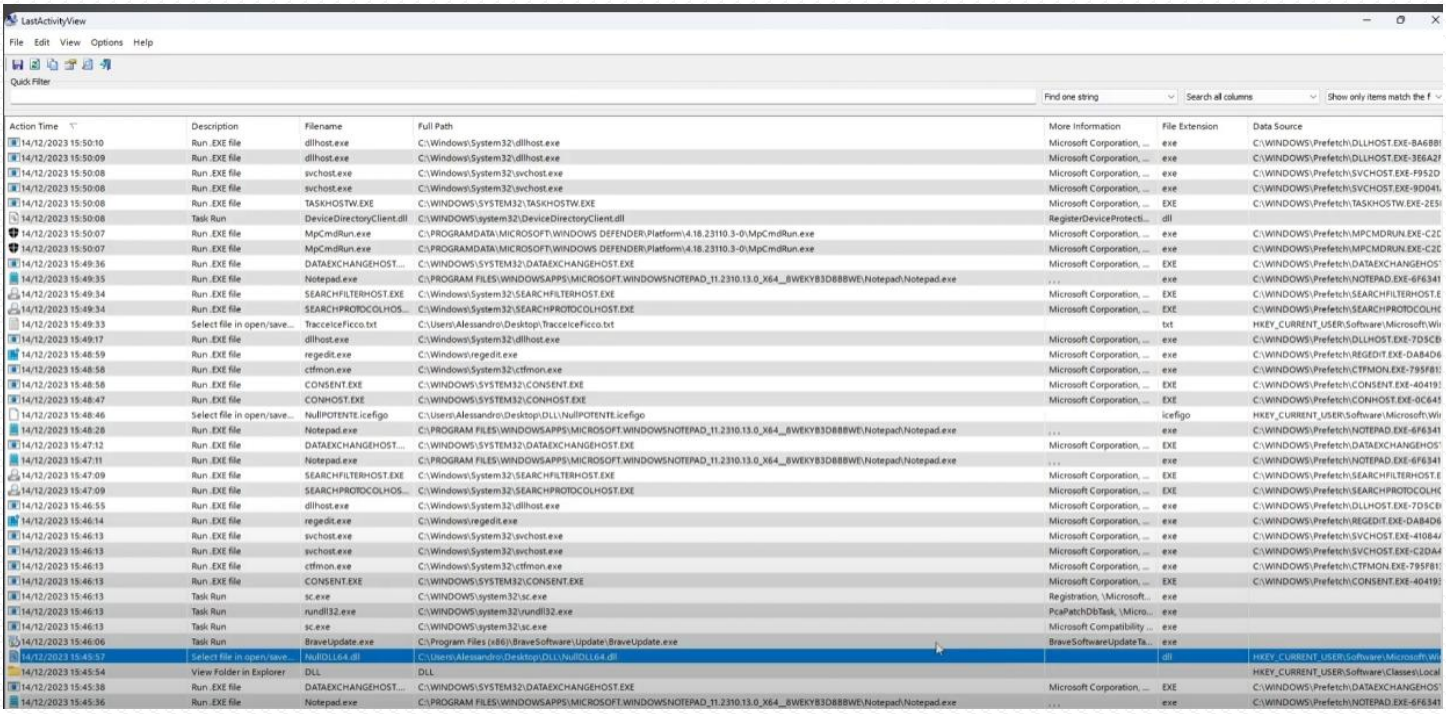
Injected DLL

There are several methods to find out if a .dll file has been injected. In this guide I will explain 3 of them. The first two methods are explained only for “general knowledge” and not for their actual use during SS. This is not because they do not work, but simply because the third is clearly superior to the first two. The

first two also work only and exclusively to identify injected DLLs as basic methods, for example by processhackers or poor Injectors.

LastActivityView

LastActivityView is a tool from NirSoft that collects data from various sources, such as prefetch, system registry, and recent file history (shell:recent).
This tool can be useful for detecting the launch of an injected .dll file even with modified (spoofed) extensions.
Download and launch LastActivityView, then analyze the results to detect suspicious file launches.



Regedit

Open the registry (Regedit) and go to this path:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU
Find the subkeys that contain the extension ".dll", right-click on the found subkey (eg: ".dll"), save the file with a name (eg: "DLL") and the extension ".txt", click "OK".
Open the exported .txt file and analyze its contents to locate the .dll files mentioned.
If you suspect that the .dll file has a different extension (eg: .cfg), repeat the procedure, locating and exporting the subkey related to the new extension.

System Informer + RL Signature Check

Open System Informer and find the "csrss" process. Select the process that uses the most "private bytes". Right-click the selected process, then choose "Properties" and then "Memory", in the "Options" tab, check "Hide free pages" and "Hide reserved pages", in the "Strings" tab, set "Minimum length" to 4, check "Extended Unicode", "Mapped" and "Image".

Use the following regex in the "Filter" field:
^[A-Z]:\\.+\.dll$
This regex searches for full paths to files that end with ".dll". It can be modified if necessary to search for spoofed extensions. Select the results that contain .dll files, right-click and select "Copy Result", paste the copied paths into a text file called "paths.txt" on your desktop, open Command Prompt as administrator.
Type: *cd C:\Users\%username%\Desktop*
Type: *powershell Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass && powershell Invoke-Expression (Invoke-RestMethod https://raw.githubusercontent.com/bacanoicua/Screenshare/main/RedLotusSignatures.ps1)*

This script will show the paths of the found .dll files and the status of the digital signature

In the screen that opens, double-click on "Signature Status" and analyze the .dll files.

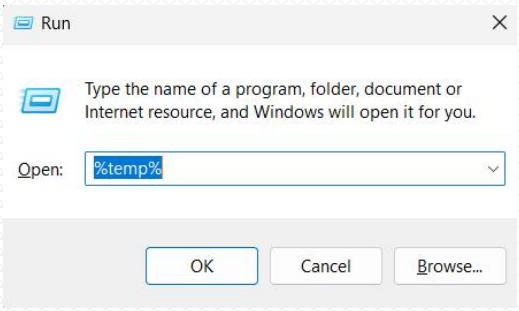
JAR Cheats (Not mods)

Files with the .jar extension are often associated with Java programs. Detecting the launch of such files can be critical in detecting the execution of unauthorized software, such as cheats or other suspicious programs.

JnativeHook

JnativeHook is a Java library that allows you to intercept keyboard and mouse events. Some cheats in .jar use this library.

To get this test:
Press Windows + R, type %temp% and press Enter.



In the Temp folder, look for files or folders with this name. If you find a file with the name "JnativeHook", it means that a .jar program (potentially a cheat) that uses this library has been executed. The modification date of these files corresponds to the time of execution.



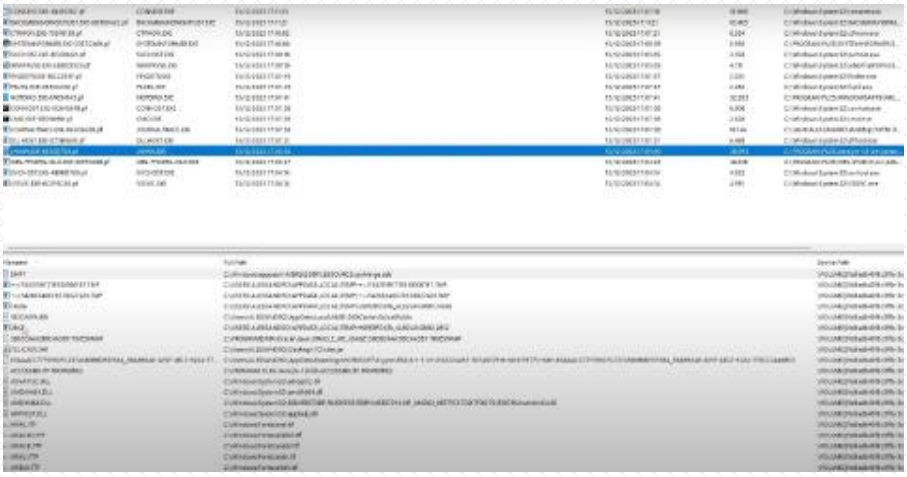
Note: This method is not always reliable. Some .jar cheats do not use JnativeHook and therefore do not leave traces in this file. Also, the player may have deleted the JnativeHook files.
In that case, use the Journal to check if the files have been deleted.
Also, as many can imagine, you could completely skip the procedure of searching inside %temp% for JnativeHook and just check its creation date through the Journal from the start.

WinPrefetchView

WinPrefetchView is a NirSoft tool that analyzes prefetch (.pf) files. Prefetch records information about frequently launched programs, including .jar files.

Open the tool and check the "Indexes" of the .pf files related to "java.exe" or "javaw.exe". If you find a .jar file (e.g. "7Clicker.jar") in the "Indexes" of a Java .pf, it means that the file was started by Java. Prefetch also shows the path where the file was located when it was started.

For a reason that we will spare you for now, given the digression and the details that it would involve, this method could lead to false flags in very rare cases. Specifically, the index related to the .jar present in the .pf could be left over from a previous start of the process. For this reason, it is advisable to integrate it with further checks, for example through the following method.



System Informer (DCOmLaunch/PlugPlay , PcaSvc)

Run the tool as administrator, search for the "PlugPlay" process, apply a "Contains (Case-Insensitive)" filter by searching for "jar", then analyze the results.
The same method can also be repeated for the PcaSvc process..

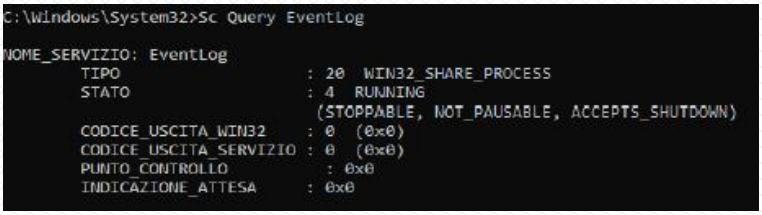


EventViewr

Event Viewer is a Windows tool that records information in dedicated logs, documenting errors, crashes, and other system activity. This information can be useful during audits to find anomalies or evidence of attempted bypasses.

Servizio EventLog

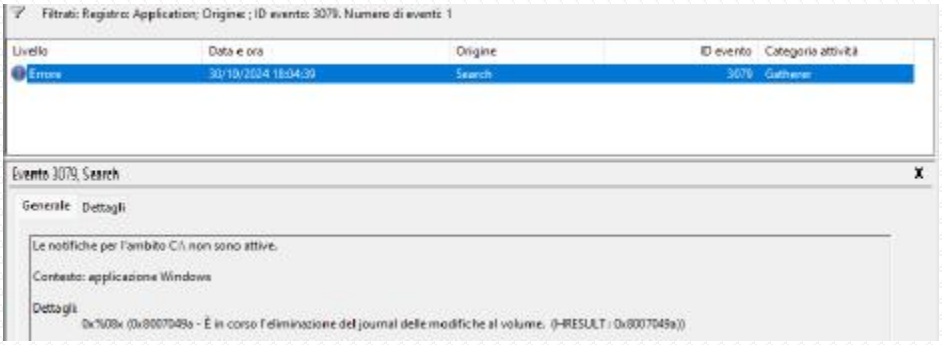
Event Viewer is connected to the "EventLog" service. If this service is disabled, Event Viewer will not open.
To check the status of the "EventLog" service, open the Command Prompt as an administrator and type: sc query eventlog.
If the status is "RUNNING", the service is running.
If the status is "STOPPED", the service is disabled and Event Viewer will not work.



Event Viewer is mainly used for the following purposes in SS at basic levels:

Finding Journal Deletion:

Open Event Viewer, go to "Windows Logs" section and select "Application", click "Filter Current Log", enter Event ID 3079 and analyze the result..



System Registry

Event ID 104 indicates deletion of registry log events



Time Change

Open Event Viewer, go to the "Windows Logs" section and select "Security", click on "Filter Current Log", enter the event ID 4616 and analyze the result.

Events associated with C:\Windows\System32\svchost.exe: Generally generated by the system (but also by the user, if the time is changed from the settings).

Events associated with C:\Windows\System32\cmd.exe: Indicate a time change made via the Command Prompt and, therefore, by the user..



Security Log

Open Event Viewer, go to "Windows Logs" section and select "Security", click on "Filter Current Log", enter event ID 1102.

This event ID indicates an Audit Log Deletion.
Note: If you try to delete this event as well, the system will automatically recreate it..

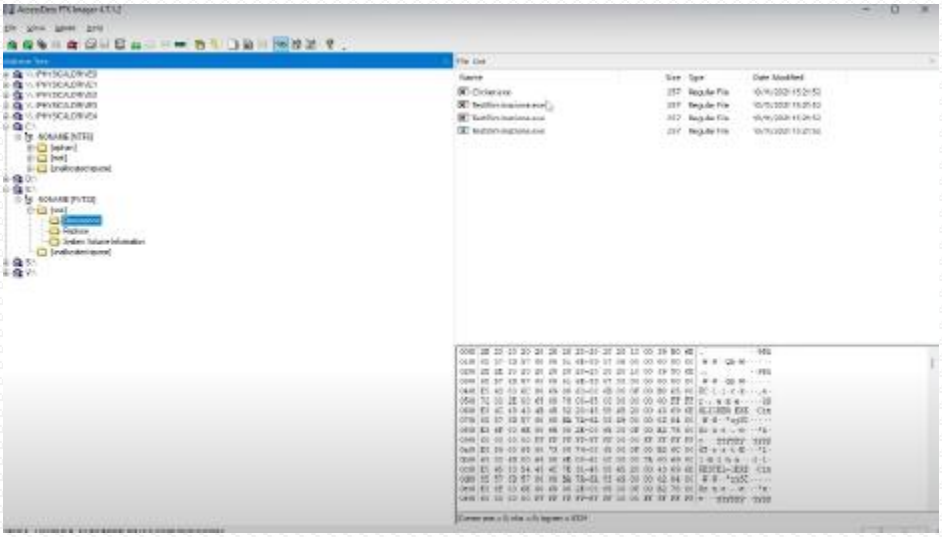


Deletions and Replaces in Fat32

The FAT32 (File Allocation Table 32-bit) file system is a type of driver used primarily on USB flash drives and small partitions. Unlike the NTFS file system, which is the current standard on computers, FAT32 does not support journaling, making it more difficult to track file deletions.
One way to detect file deletion or replacement on FAT32 is to use a tool like AccessData's FTK Imager
This tool allows you to view changes made to many different types of drivers, including FAT32.

How to use FTK Imager to search for deleted files:

Open FTK Imager with administrator permissions, select the "Add All Attached Devices" option, locate and select the folder you want to scan. Then check the files on the volume, looking for deleted files.
FTK Imager highlights the files that have been deleted with a red X in it's icon.



Deleted File Recovery:

FTK Imager also allows you to recover files that were deleted normally (not overwritten).
Right-click on the deleted file you want, Select "Export Files...", choose the folder where you want to save the recovered file and click OK..

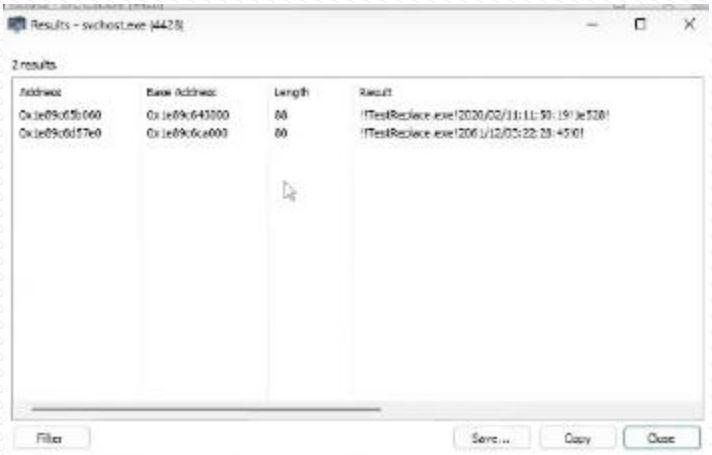


How to Use FTK Imager to Find a Simple Replace:

The procedure to find a replace on FAT32 is similar to that used to detect a deletion. When a file is replaced, the previous file is in fact deleted from the file system. Therefore, you can use FTK Imager to detect the removal of the original file.

To prove that the previous file and the current one are different, you can use several methods.
As for the exe, a recommended and quick one is that of DPS.

First of all, start the exe currently present in the folder.
Then start System Informer with administrator privileges, locate the -s DPS process and filter for "!!"
Then filter for "FileName.exe" by entering the name of the .exe file that you suspect has been replaced.
Finally, analyze the results. If you find two results with different timestamps (date and time) related to the same .exe file, it means that two different exe with the same name have been opened in the instance.



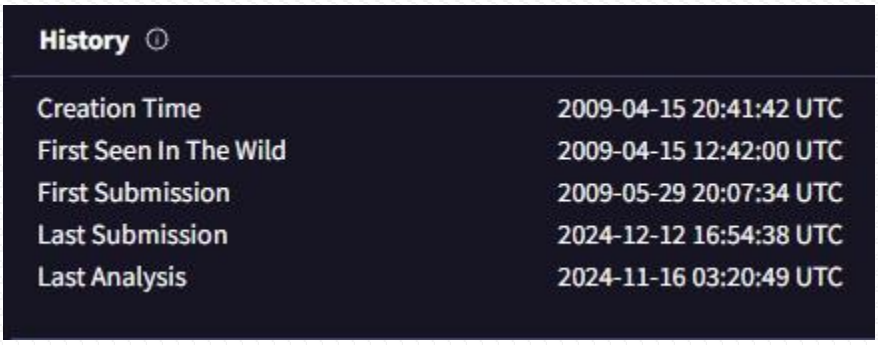
Attention:
The presence of two files with the same name and with different (or identical) TimeStamp is not in itself a valid proof to ban a player.

In fact, it is necessary to perform a final check before being able to take action against the player, that is, it is necessary to prove that the executable currently present in the folder in FAT32 is different from the one previously present.

To do this, we must first of all make sure that the timestamps inside the DPS do not belong to other files with the same name present in the player's PC. To do this, through a tool like everything we can check the various files present in the player's PC with the same name, open them and see if their opening generate new strings inside the DPS.

Once we have ascertained that the two strings initially present in the DPS did not belong to other programs with the same name present in the player's computer, we can ban the player.

Note: A further step that could be taken, but only if the timestamps of the legit and unlegit files are different, is to prove that one timestamp belongs to the file currently present in the folder, and the other timestamp belongs to a file that at the time of the check is no longer present on the computer and has therefore been deleted. This can be done very simply through VirusTotal website in the details tab by grabbing the timestamp of the file currentlly present in the pc.



02

Public screenshare guide

98 original pages / reproduced in full

[RedLotus] - Public SS Guide (FULL Release) (5).pdf

Original source content begins on the next page. Author credits and illustrations are preserved.



|———————Introduction———————|

≛ *Red Lotus Unified* ≛

〔————————〕*The Foundation*〔————————〕

Red Lotus consists of
Currently the best SS Team
In the entire SS **Community**;

It aims to improve the **quality**
of ScreenShares on servers.

Its main objective is **safety**
of Staff and Players together.

by **UnMonsieur**

Following material is a collection of the most of Red Lotus methods and digital handbooks to aid ScreenSharers of all levels. Most of what was ever discussed in Webinars and Academy meetings is updated here. **(Tome I, Tome II [Lotus+], Tome III)**

The guide is made by:

UnMonsieur

Table of Contents

|———————Introduction———————| 2
『Red Lotus』 5
RL Charter of SS Establishment 5
Requisites of ScreenSharers, Article I 6
Protocols of ScreenSharing, Article II 7
Extended regulations, Article III 8
『Chapter I』 9
|++++|—First Phase—|++++| 10
Prefetch 10
Temp 12
shell:recent 13
Search Everything & System Informer 15
Deleted Items 15
MiniSS Powershell Script 17
WinPrefetchView 18
Task Scheduler 18
Suspended services & threads 19
Search Everything 20
USBDriveLog 21
X-Mouse button control & Startup items 21
Mod Clients & .Jars 24
|++++|—Second Phase—|++++| 25

Event Logs....25
System Informer....27
Anti-Scheduler....28
Maceta....28
OSForensics....29
[__Processing the Case....29
[__File System Browser....30
[__Volume Shadow Copies....31
[__Program Artifacts Viewer....31
[__Deleted Files Search....32
[__User Activity Timeline....32
Sherlock Finds....33
|++++|—Third Phase—|++++|....34
Red Lotus Collector....34
Velociraptor NTFS....35
Rancio's Tools....35
DoomsDay Client Finder....36
Unicode Detector....36
Ocean SS Tool....36
GlobalLister....37
Powershell WinPrefetchView....37
『Chapter II』....38
<>!<> Forensics Stage I <>!<>....39
Magnet EDD....39
INDXRipper....39
$MFT....40
Kernel Live Dump....40
<>!<> Forensics Stage II <>!<>....41
Red Lotus....41
Linux ScreenSharing....42
Terminal....43
Directories....43
Opening Minecraft/Jar files....44
Modified recycle bin....46
Installed applications....46
Current user....46
Regedit....47
Fast access....47
Restarting shell before SS....47
Modifying files....48
Unplugging USB....49
『Chapter III』....50
Ban Evasion....51
Mod Analyzer....52
DPS-Sanner....52
Anti-Debugger....52
How will you detect proxy cheats....53
Event Logs bypass....54
Java bytecode execution traces....54
Cheats that don't load anything with AnyDesk....54
Renamed extension with no traces....54
Kernel live dump....55
Startup XML....55
『Chapter IV』....56
History of Bypasses 📑....57
First wave....58
Read-only rundll & regsvr....58
Alternate Data Streams....58
Unicode Characters....58
(Section redacted)....59
(Section redacted)....60
(Section redacted)....60
RunPE (VBA)....60
Suspending Threads & Tokens....61
『Chapter V』....62

📑**Anti-Forensic Labs**.....**62**
Anti-Forensics.....62
Extensionless Files.....62
(Section redacted).....62
(Section redacted).....62
(Section redacted).....62
(Section redacted).....63
(Section redacted).....63
(Section redacted).....63
(Section redacted).....63
(Section redacted).....64
『Chapter VI』.....**65**
Blue Team DFIR🧿.....**65**
Digital Forensics.....65
Activities Cache.....66
AMCache.....66
Windows Tasks.....70
EVTX.....71
ADS Detections.....76
(Section redacted).....77
『Chapter VII』.....**78**
Collection of Evidence.....78
Evidences of Execution.....79
-BAM/DAM.....80
-Renamed Extensions.....81
-Injected & Renamed DLL.....82
-EventLog Manipulations.....83
-Temporary Traces.....84
-Uptime of Apps.....85
『Chapter VIII』.....**87**
Hypothetical Scenarios.....87
Example A.....87
Example B.....87
Example C.....89
Example D.....90
Example E.....90
Example F.....92
Example G.....93
Additional Material.....93
『Red Lotus』.....**94**
Chapter 1 - Granting Rights.....**95**
Chapter 2 - Branch Markets.....**96**
Chapter 3 - RedLotus Merchanting.....**96**
Chapter 4 - RedLotus Packets.....**97**

『Red Lotus』

RL Charter of SS Establishment

This charter establishes norms and regulations for the selection and proper implementation of ScreenShare teams within servers, communities, individual teams; suggested for no imperative reason for implementation by Red Lotus Unity.
It is deemed by this server that the security and safety of players is more important than the accuracy and speed of ScreenShares, whether advantages are detected or not.
It is deemed by this server that the importance of data security is imperative over anything else that may favor the server.
It is condemned by this server to disestablish institutions regulating fair-play and privacy measures within communities.
It is deemed by this server that the experience and specialization of so-called “ScreenSharers” (other: PC Checker, Account Reviewer. . .) shall be tested and approved by ones in charge.

The following regulations defined in this charter present an extension of the General Charter of Red Lotus.

~ UnMonsieur | Headquarters of Red Lotus

<u>Requisites of ScreenSharers, Article I</u>

(1) Servers that hire or implement the institution of ScreenSharers shall include the age, country, past experience questionnaires in their “SS Applications”, for the purposes of:

a) ScreenSharers shall be over the age of 16 (at minimum), with at least 4 months of experience purely on the side of ScreenSharing to ensure general principles of RL Charter to the minimal degree.

b) ScreenSharer’s country shall be noted for any legal complications to be sorted with the most possible degree, if possible. IP or Geolocation, or State/City, Urban, MUST NOT be requested or kept.

c) ScreenSharers must have at least 4 months of experience purely on the side of ScreenSharing. By experience, it is implied to be verified by manual ScreenSharing.

(2) Servers that hire or implement the institution of ScreenSharers must verify the experience of ScreenSharers by conducting manual Bypass / Anti-ScreenShare techniques and testing them properly.

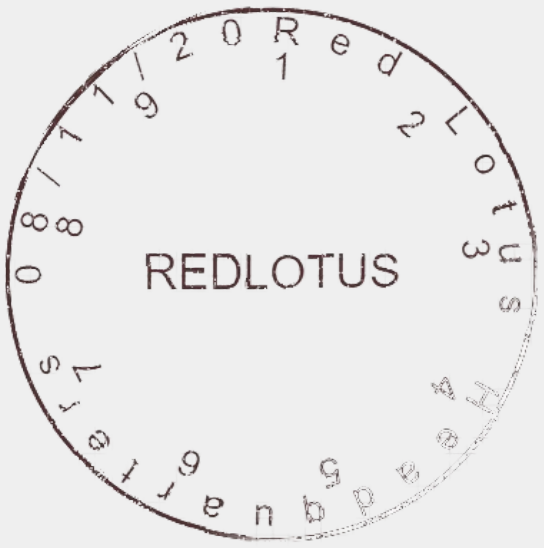
Protocols of ScreenSharing, Article II

(1) ScreenSharers must record the entirety of ScreenShares, regardless of the weakness of their devices, for the purposes of:
a) Safety of the Player; evidence must be previewed by SS Managers, Owners, or ones in charge of the institution.
b) Accuracy of the ScreenShare;
c) Truthness of the ScreenShare; to avoid malicious actors planting advantages on the user devices / framing them.
d) Legal requisite; this will serve a legal purpose, in accordance with data security protocols, acts, legislative norms, international privacy pacts.
e) This will also serve the purpose of verifying the skill set of ScreenSharers and their truthfulness.
(2) ScreenSharers not following these regulations may face a penalty in forms of demotion or suspension, or re-evaluation of their skill sets.
(3) Players banned in a ScreenShare with no video evidences shall immediately be unbanned.
(4) Screenshots instead of recordings shall be made acceptable if the ScreenSharer has proven their respect to the privacy and security of Players. However, in case of allegations, previous regulations shall also be applied.

Extended regulations, Article III

Servers or communities that decide to strengthen their bond with Red Lotus may request the Headquarters of Red Lotus to evaluate the specific custom of ScreenSharing & Player-base of theirs and extend these regulations in accordance with their server.
With mischief, servers and communities must not pretend to implement the following Charter to manipulate with corruption and unjust, malicious acts.





『Chapter I』

|++++|—First Phase—|++++|

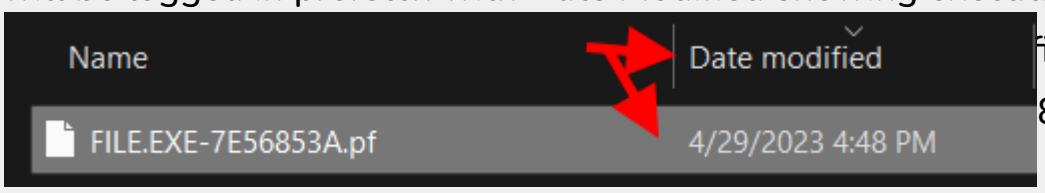
In the beginning of a ScreenShare, unlike many servers do - Red Lotus doesn't recommend or regulate checks for recorders. It is rather an extremely good self-defense mechanism for the Player themselves. Therefore, we'll move onto checking **Prefetch, Temp, Recent Items, Deleted Items and then we'll run an immediate PowerShell script** to speed up the starting process.

Prefetch

General Information about Prefetch

Prefetch is a Windows component that plays a huge role in memory management; executed applications leave a prefetch entry in C:\Windows\Prefetch directory that represents their execution time.

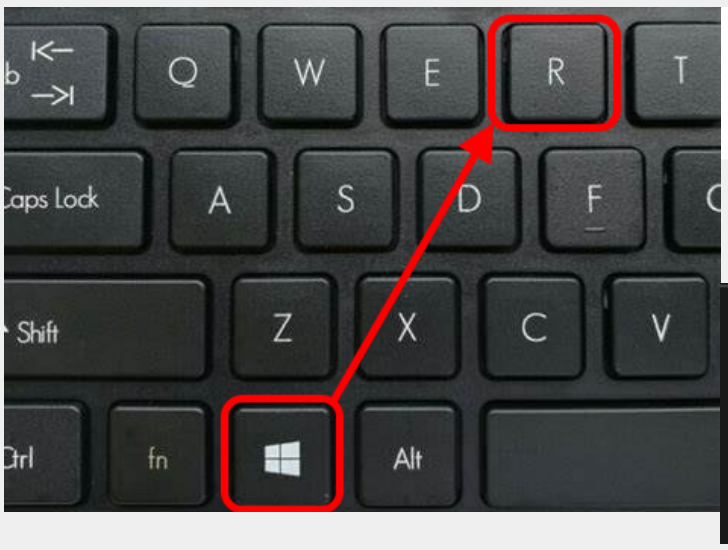
- Prefetch is only generated when a .exe is run, for java based applications or DLL files it'll instead generate java.exe and dllhost.exe / rundll32.exe / regsvr32.exe entries. Why is this important to know? Because this way we'll know if the Player has executed a renamed extension cheat. For example - Cheat.exe renamed to Cheat.dll.
- Prefetch is the evidence of execution, in other words, it logs the actual execution time of files. For example - executing FILE.exe at 04:32 will not update that time on the file's properties themselves, however, that execution will be logged in prefetch with Date Modified showing execution time.

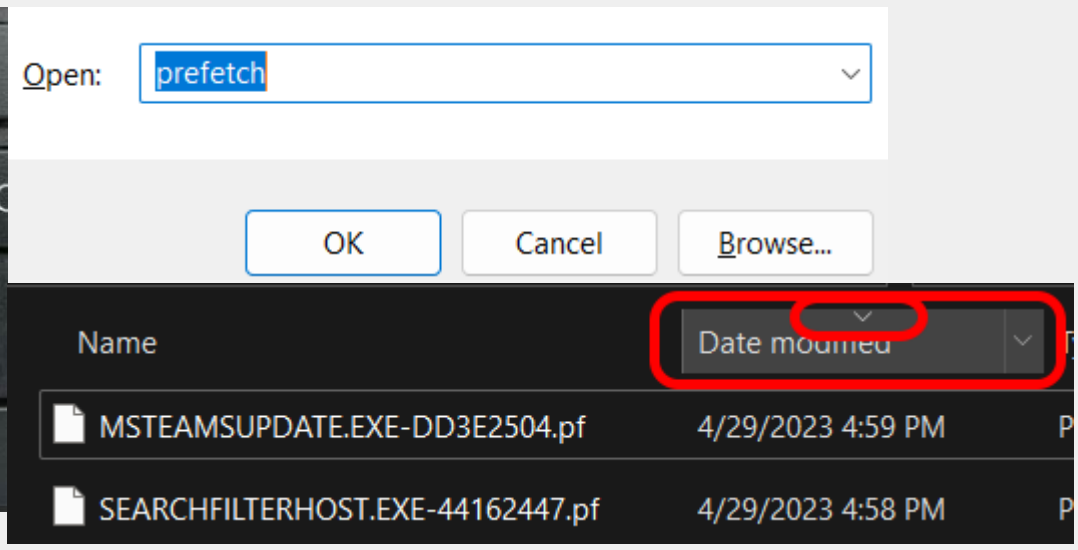


file will be left with a Hash number, like 853A.pf, for example.

- File's Hash represents their path - where was the file executed from. Therefore if you see the same file name with different Hashes, it is recommended you look more into it and mark down the file's name to not forget for later checks.
- Tools such as WinPrefetchView and Last Activity Viewer rely on Prefetch and Regedit. Which means if prefetch has been cleared / deleted - they will show no traces of execution. Therefore it is also important to check for deleted prefetch files.

To Access Prefetch you must press Win+R and type Prefetch in the Run Box. Sort by Date Modified.





Alternatively, you could use CMD to sort modified prefetch entries by date modified and applied attributes like hidden, read-only and so on. Run CMD as Administrator.

```
C:\Windows\System32>dir /a:h /o:-d C:\Windows\Prefetch
 Volume in drive C has no label.
 Volume Serial Number is 584E-358C

 Directory of C:\Windows\Prefetch

04/29/2023  05:04 PM            36,579 vape_lite.pf
```

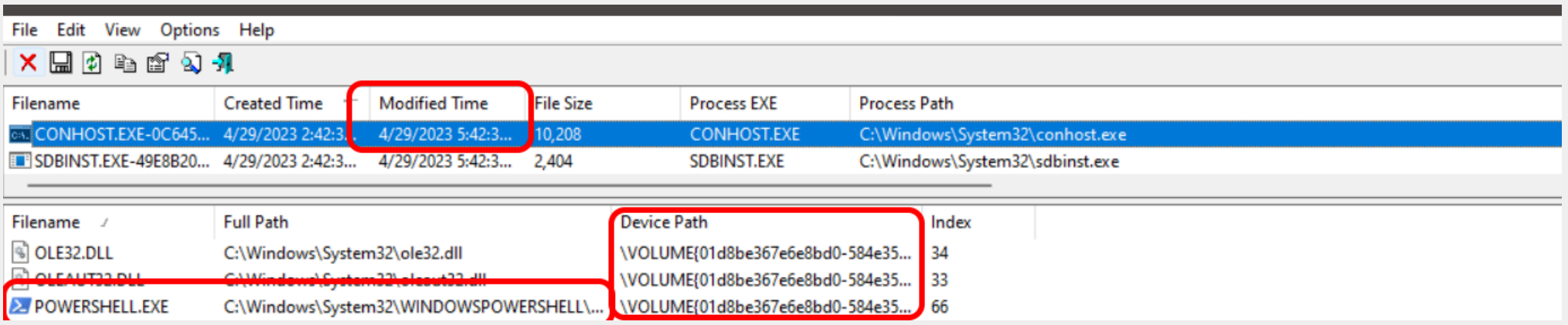
```
C:\Windows\System32>dir /a:r /o:-d C:\Windows\Prefetch
 Volume in drive C has no label.
 Volume Serial Number is 584E-358C

 Directory of C:\Windows\Prefetch

04/29/2023  05:05 PM             8,655 autoclickerlol.pf
```

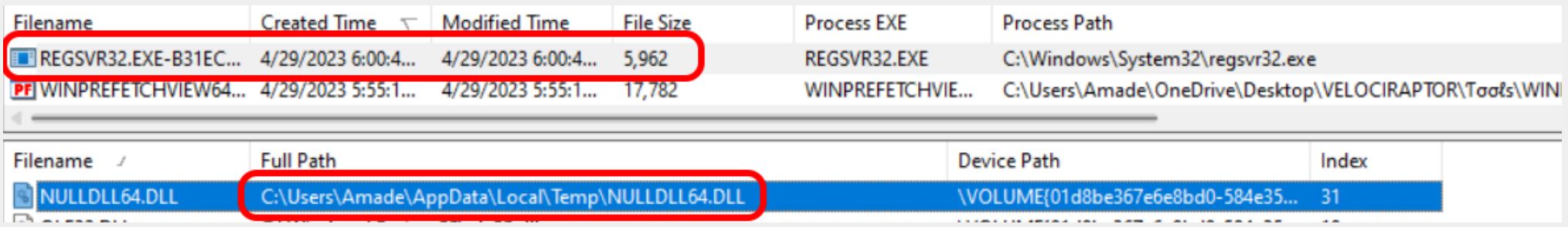
It is worth noting that we can use WinPrefetchView by Nirsoft to also analyze executions by Conhost & Consent. The

top part is where you look for Conhost or Consent and the bottom part of the tool is what shows which DLLs were loaded or .exes were launched. Note that in Conhost and Consent, we're only looking for executables and their directories.



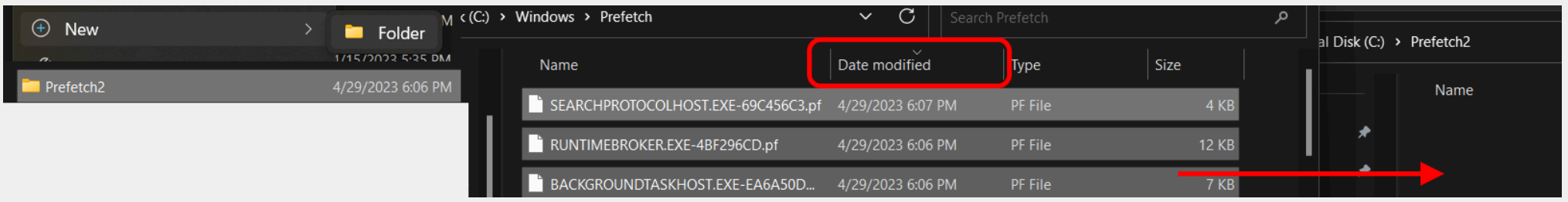
\VOLUME refers to the device path, or Disks to be exact (C:\, D:\, E:\, and so on).

- Pay attention to Modified Time as that refers to last write time.
- Pay attention to icons and directories, if Device Path of a .exe without an icon is gone, alongside with its modification dates and so on, that is quite suspicious.
- Paying attention to directories becomes more important with rundll and regsvr Injections.



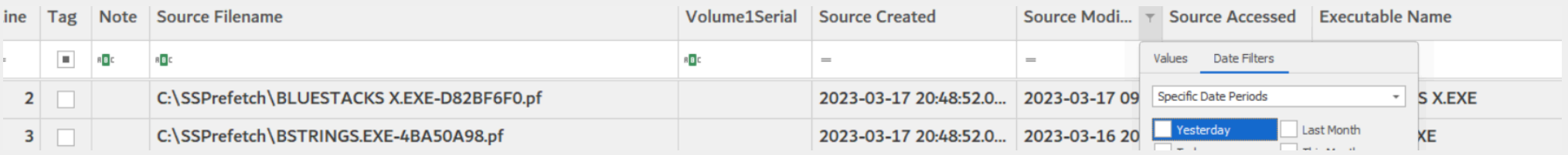
Regsvr and RunDLL services usually are never issued by the system itself, and even if they were to be issued by the system - they'd never load anything from users directories.

> Some users (or Windows versions) may "break" their prefetch and make WinPrefetchView be unusable. This will make a ScreenShare very painful to perform, however we can still use Zimmerman's PECmd to parse the whole Prefetch. In order to do so, create a folder and name it whatever you want (I'll name it Prefetch2) wherever you want (I'll create it in C:\Prefetch2), copy prefetch entries of the current date that you want to parse and move them there (Or copy paste with CTRL+C CTRL+V).



Use Zimmerman's PECmd.exe and run the following command:

`PECmd.exe -d "C:\Prefetch2" --csv .` Note that after –csv we specify an output directory, thus the output file will be saved to whichever directory PECmd is located in. The output will be in CSV (comma separated values) which can be opened with either Excel or Timeline Explorer by Zimmerman.



We can sort the search results with a lot of options, for extended navigating options visit this demonstration link;

Note that it is recommended you use the navigating options on mentioned prefetch entries (RunDll / Regsvr / DLLHost / Conhost / Consent)

Temp

Temp is a windows temporary files directory, in other words, it is a folder located in C:\Windows\%username%\AppData\Local\Temp that stores temporary actions performed by Windows or applications, such as browser / downloaded files cache (it may include images, executables, dlls and so on).

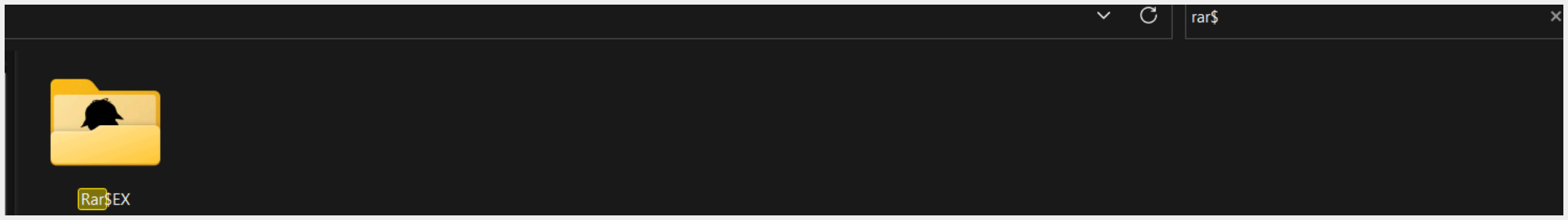
- By pressing Win+R and typing %temp%, you'll access this directory.
- Look for things like JnativeHook - Rar$ - Scoped - Zz in the search to find the following:

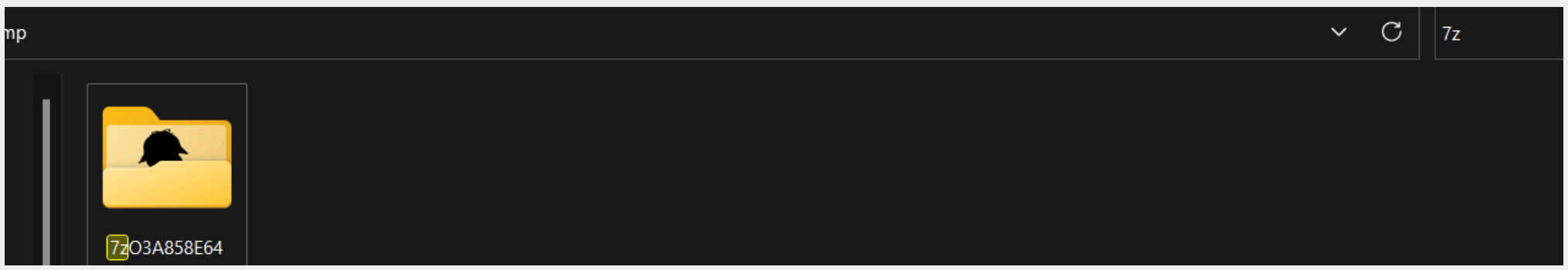
  - JnativeHook stands for 7Clicker, it is a .dll file that generates in %temp% upon the clicker's launch, date modified represents its execution time.

  - Rar$ represents executions from WinRar compressed files

  - Scoped represents scoped directories, they usually store image caches from Windows apps (often Discord profile pictures and server icons)

  - 7z is the same as Rar$ (7zip)





In the past this directory was also being used to store .DLL injectable cheats as it'd be extremely difficult to distinguish regular DLLs. So using a Powershell script to check unsigned executables in the directory and its sub-directories might be useful.

Get-ChildItem -Recurse $env:temp*.dll -ea SilentlyContinue| ForEach-object {Get-AuthenticodeSignature $_ -ea SilentlyContinue} | Where-Object {$_.status -ine "Valid"}|Select Status,Path

```
PS C:\WINDOWS\system32> Get-ChildItem -Recurse $env:temp\*.dll -ea SilentlyContinue| ForEach-object {Get-AuthenticodeSig
nature $_ -ea SilentlyContinue} | Where-Object {$_.status -ine "Valid"}|Select Status,Path

   Status Path
   ------ ----
NotSigned C:\Users\Amade\AppData\Local\Temp\NullDLL64.dll
```

You could modify the script to look for unsigned .exe files in the %temp% directory as well, since it may have tons of folders that include a cheat. (Detects rar$ex subdirectories as well)

Get-ChildItem -Recurse $env:temp*.exe -ea SilentlyContinue| ForEach-object {Get-AuthenticodeSignature $_ -ea SilentlyContinue} | Where-Object {$_.status -ine "Valid"}|Select Status,Path

```
PS C:\WINDOWS\system32> Get-ChildItem -Recurse $env:temp\*.exe -ea SilentlyContinue| ForEach-object {Get-AuthenticodeSig
nature $_ -ea SilentlyContinue} | Where-Object {$_.status -ine "Valid"}|Select Status,Path

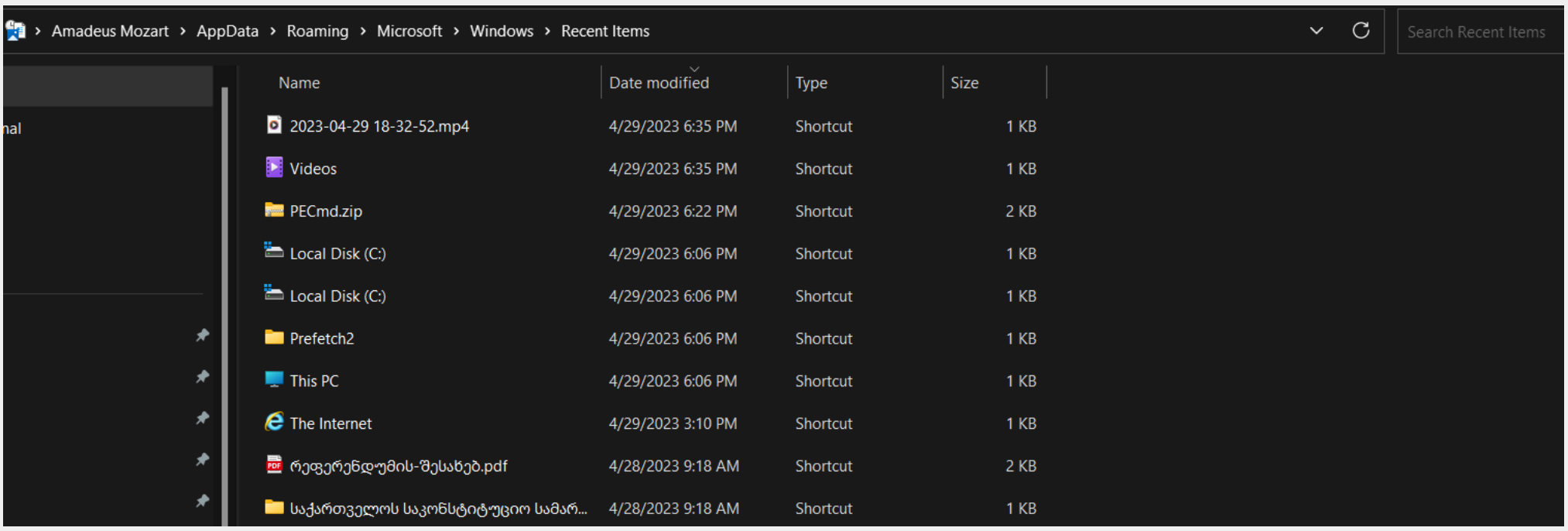
   Status Path
   ------ ----
NotSigned C:\Users\Amade\AppData\Local\Temp\Rar$EX\Vape Lite.exe
```

You must remember that JnativeHook and JNA are not the same. JNA-[Random Numbers] refers to modified Minecraft launchers (such as Lunar client, badlion, feather client, salwyrr and so on.)

shell:recent

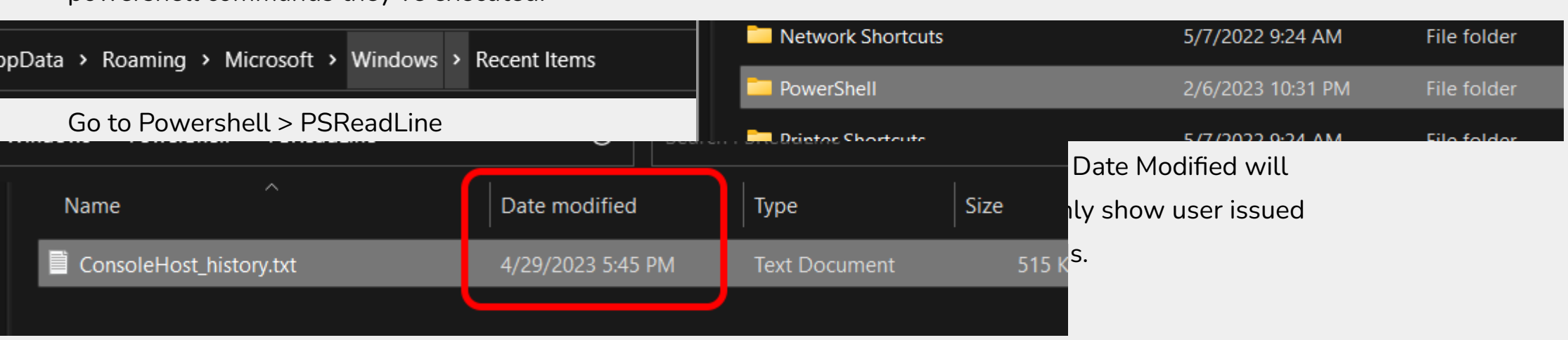
Recent Items is one of the most useful targets on Windows, not only can they show virtual disk launches, .bat cheats and so on, but it also shows recently visited directories. It's located in C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Recent

You may access the folder with Win+R then typing shell:recent



From here, you can view last accessed folders, zip files and so on. Note that data collected here derives from RecentDocs registry path which can be cleared. If that happens, rest assured that the whole directory will be blank, thus you can then check the deletion of registry values (explained later);

From this directory you can head back to the [Windows] directory and check [PowerShell] Folder to see what powershell commands they've executed.



Date Modified will
nly show user issued
s.

SS tools like Echo might flag "user accessed powershell" which is mostly a false flag as it launches powershell upon self execution. Use this method to check when the user has actually accessed powershell.

- Recent directory holds traces of JumpLists, that may remain even if the user has cleared RecentDocs from regedit;
- Use Zimmerman's JumpList Explorer; Run it as Administrator.
- Follow the Tutorial; Directories include [CustomDestinations] and [AutomaticDestinations]

If the tool doesn't work for whatever reason, you can use LECmd, same way as PECmd works, except here the command is:

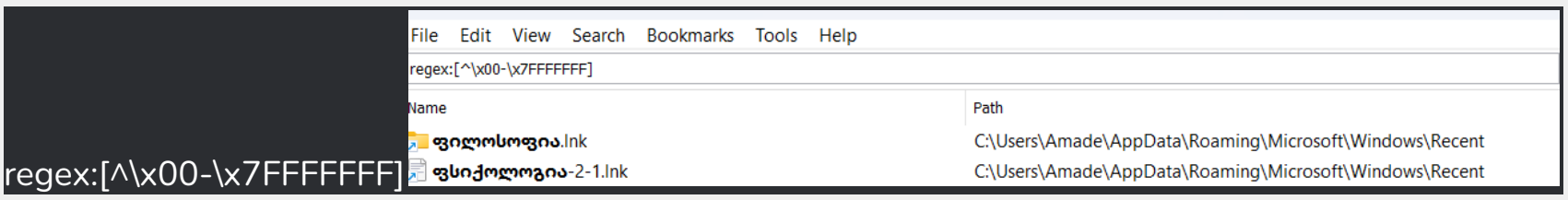
```
LECmd.exe -d "C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Recent" --csv .
```

Here's how it works

Search Everything & System Informer

Before starting any PC Check, make sure you have these two important softwares installed on the user's device, as Search Everything is a tool you'll be having to work with in most cases.

- ❖ Download Search Everything from voidtools
- ❖ Upon installation, the software will be just called "Everything" - in case you closed it & need to reopen it
- ❖ Sort Everything by Date Modified and input the following regex before you start the PC Check: (this shows hollow characters)



➢ Install **System Informer** from Nightly - System Informer

➢ Here's a list of some useful Regex patterns to use in System Informer:

1) `^(?!C:)[A-Z]:[\\](?!\+|(u){FFFF}|rkr)` Displays other disks than C:\ (search in Explorer)
(How to dump processes in System Informer)
2) `^!![A-Z]((?!Exe).)*$` Displays modified extension in DPS service
`^[0-9a-zA-Z]{6}\.json$` Theoretically displays Skript's .json in Search Indexer

More regex patterns can be checked here:
https://cdn.discordapp.com/attachments/1118212828592091236/1121040716555096135/Regexes.txt

EVERYTHING & SIGNATURE CHECKER

Download Everything (re-download if the user has it to avoid bypasses)
https://www.voidtools.com/Everything-1.4.1.1022.x86.zip

1) Filter to sort by size limit **size:5mb..25mb ext:exe**

You may change the extension to etx:dll for .dll files; most DLL cheats don't exceed 5MB.

2) Export the results **ctrl+s**

3) Sort the exported file paths using https://binarysignaturetools.000webhostapp.com/, follow the video below.

Download:
https://github.com/Si13n7/BinarySignatureStatus/releases/download/20.1.15/BinarySignatureStatus_2020-01-15.zip

.\BinarySignatureStatus <PASTE WHAT YOU HAD>

https://youtu.be/VxYRG_O8rm4 (Demonstration) ~ Kendo

Alternatively, name the exported .txt "paths.txt" and use Baca's Signature Checker.

Deleted Items

This paragraph will cover USN Journal tracks, prefetch and recentdocs deletions, macro deletions, FAT32 USB deletions.

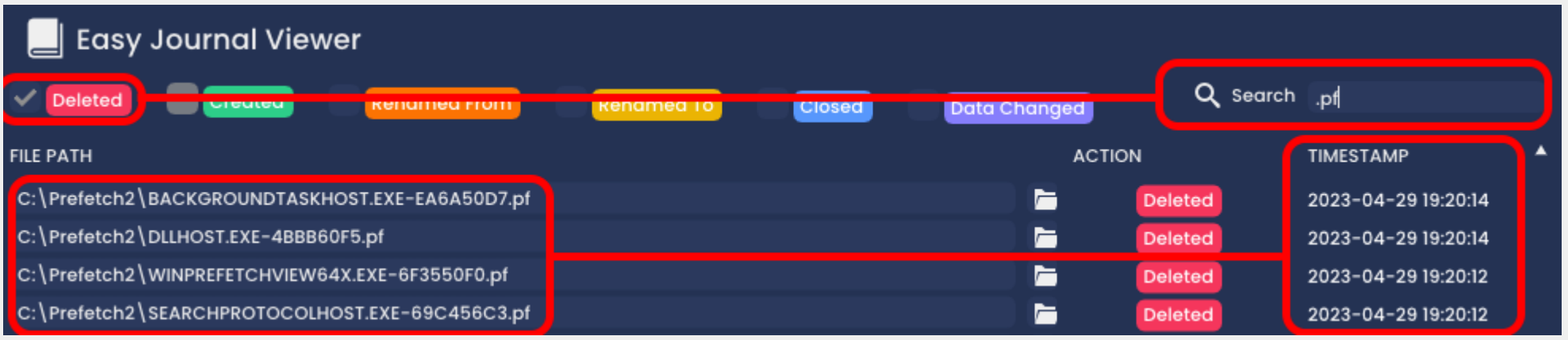
Before we proceed, you must first know what FAT32 deletions mean.

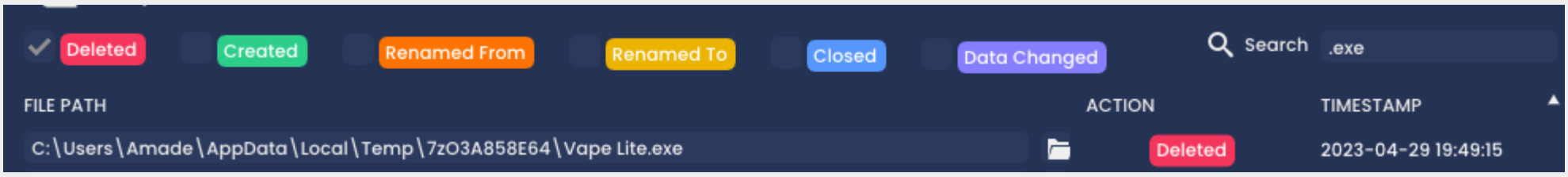
- **FAT32 is a File System mechanism, a data organizer. There's NTFS File System which is Windows specific - it includes USN Journal traces and it is often easier to recover files from the NTFS file system. FAT32 on the other hand, does not include USN Journal, which makes it one of the most complicated file systems.**

Beginning with USN Journal

We have two useful versions of a USN Journal parser, one is JournalTrace and the other one is Journal-Tool
https://cdn.discordapp.com/attachments/1014632218183860334/1014637785694621706/journal-tool.exe

Made by Echo. We'll start with Journal Tool by Echo to cover deleted Macros, Prefetch and Executables. Open the tool, mark "Deleted" and type .exe or .pf for executables and .prefetch.





Here's a list of Macro extensions that the Players might delete before SS, make sure you pay attention to the directory they were deleted from and check further:

.mcf .amc2

.GMAC .cuecfg

.MA32AIY .MSMACRO

record.ini

sequence.dat

macro_list.dat

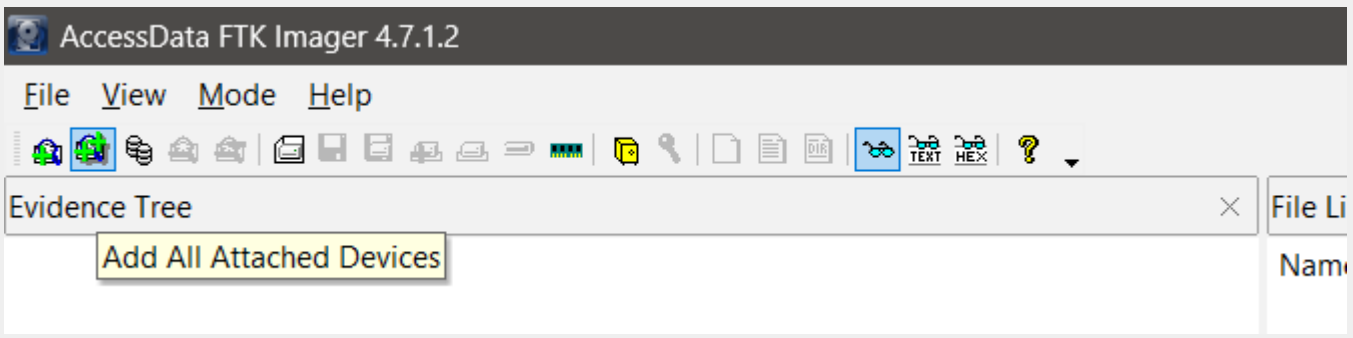
custom_macro_list.dat

FAT32 Deletions

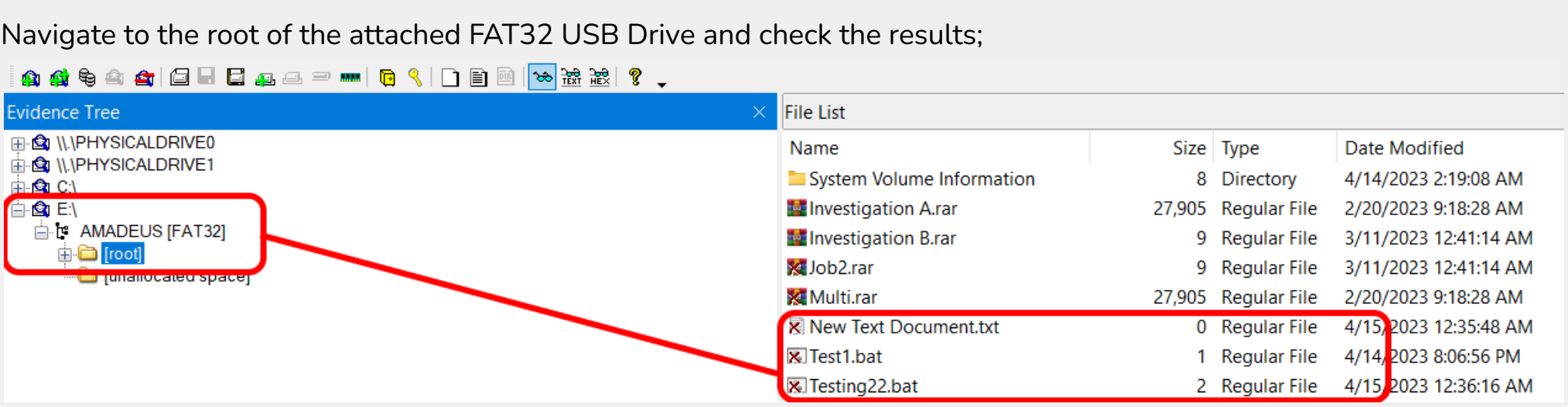
In order to view FAT32 Deletions, we'll need to download FTK Imager.

https://d1kpmuwb7gvu1i.cloudfront.net/AccessData_FTK_Imager_4.7.1.exe

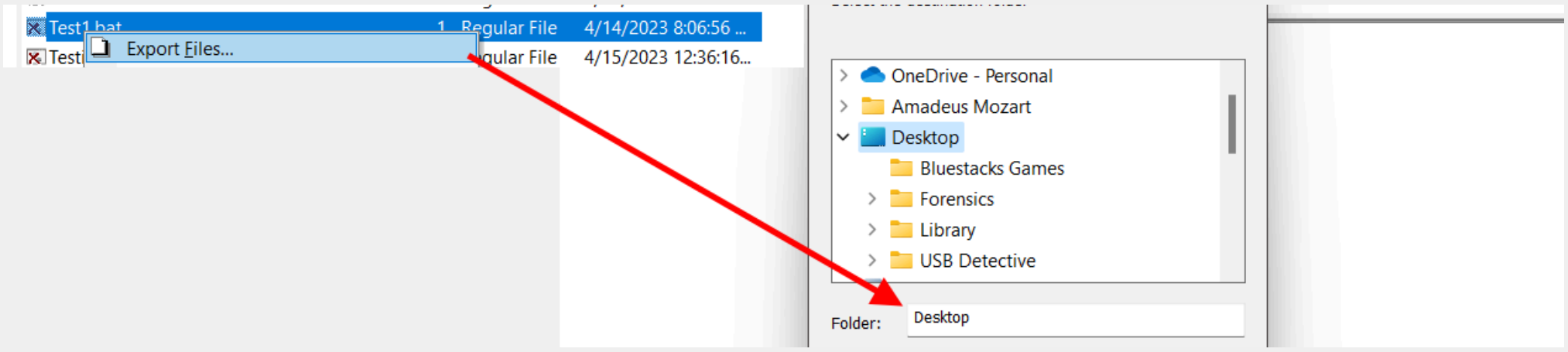
Open the FTK Imager as Administrator and load all attached Devices:



Navigate to the root of the attached FAT32 USB Drive and check the results;



Note that this does NOT show the date of deletion, so you'll need to find evidence that the file had been executed after the last system boot / restart. Sometimes you might also be able to recover the deleted files from FTK Imager by right clicking on them, if you perform this process right away in the SS, there's a higher chance you can potentially recover an executable properly. However, that isn't a guarantee.



MiniSS Powershell Script

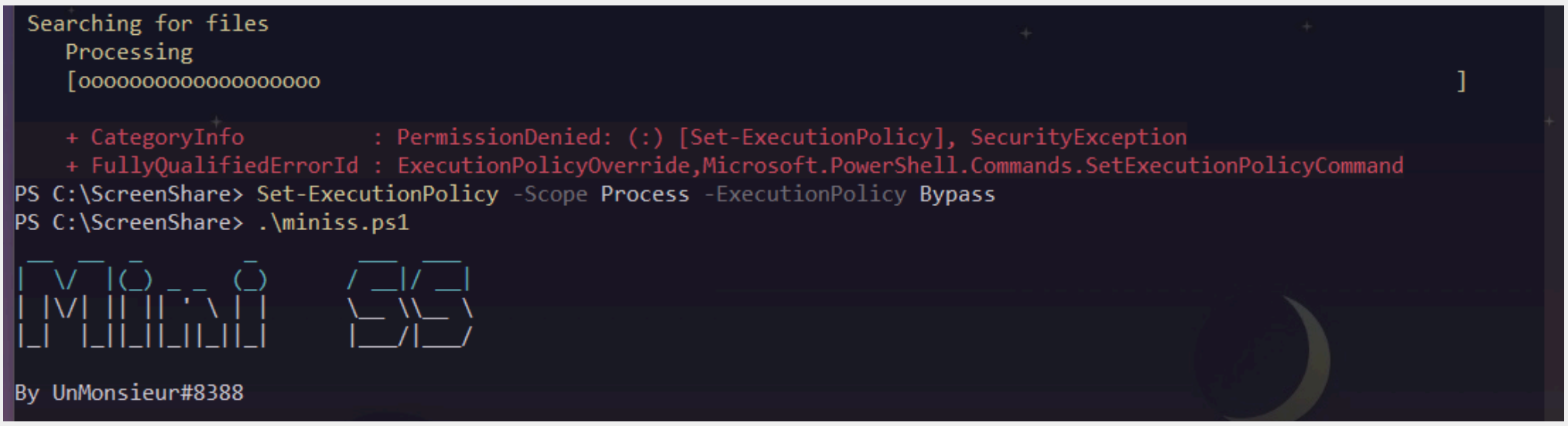
Red Lotus Unity delivers some PowerShell utilities to cooperate servers that we've mostly not shared or included in our other guides. We'll start off with MiniSS powershell Script.

MiniSS Powershell Script

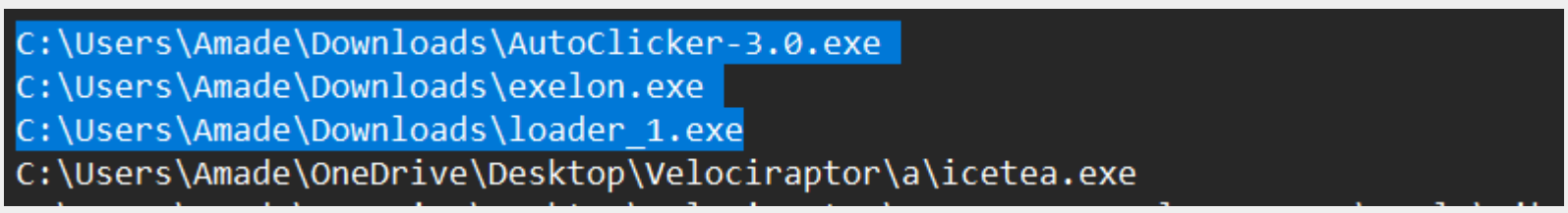
- Open CMD as Administrator and type mkdir C:\ScreenShare && cd C:\ScreenShare
- Download the powershell Script by pasting this command:
  curl -O "https://cdn.discordapp.com/attachments/1083546232745644152/1087887260948631652/miniss.ps1"
- Type powershell in CMD and run the following commands:
  Set-ExecutionPolicy RemoteSigned -Scope LocalMachine
  Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
  Press [A] on each notification.
- Run the MiniSS.ps1 by typing ./MiniSS.ps1 (or type mini and press [TAB])

Here's how the process should look

The scan will take some time to load (based on how good Player's PC is)

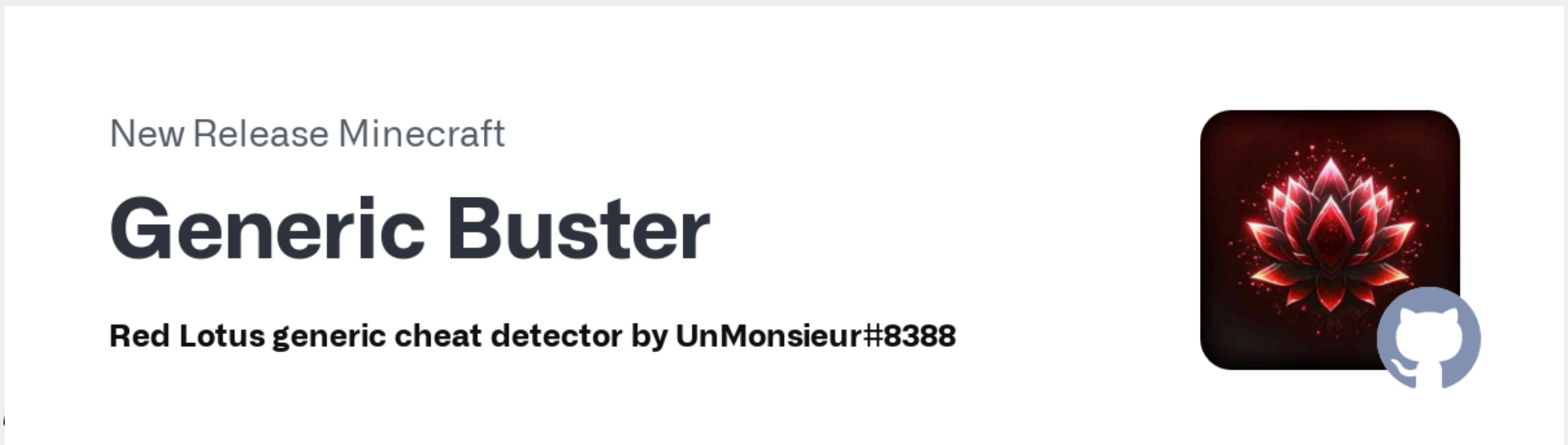


After the scan is finished, it'll output found results in C:\ScreenShare current directory as a .txt file.



There's an alternative .exe version of this script in my repositories.

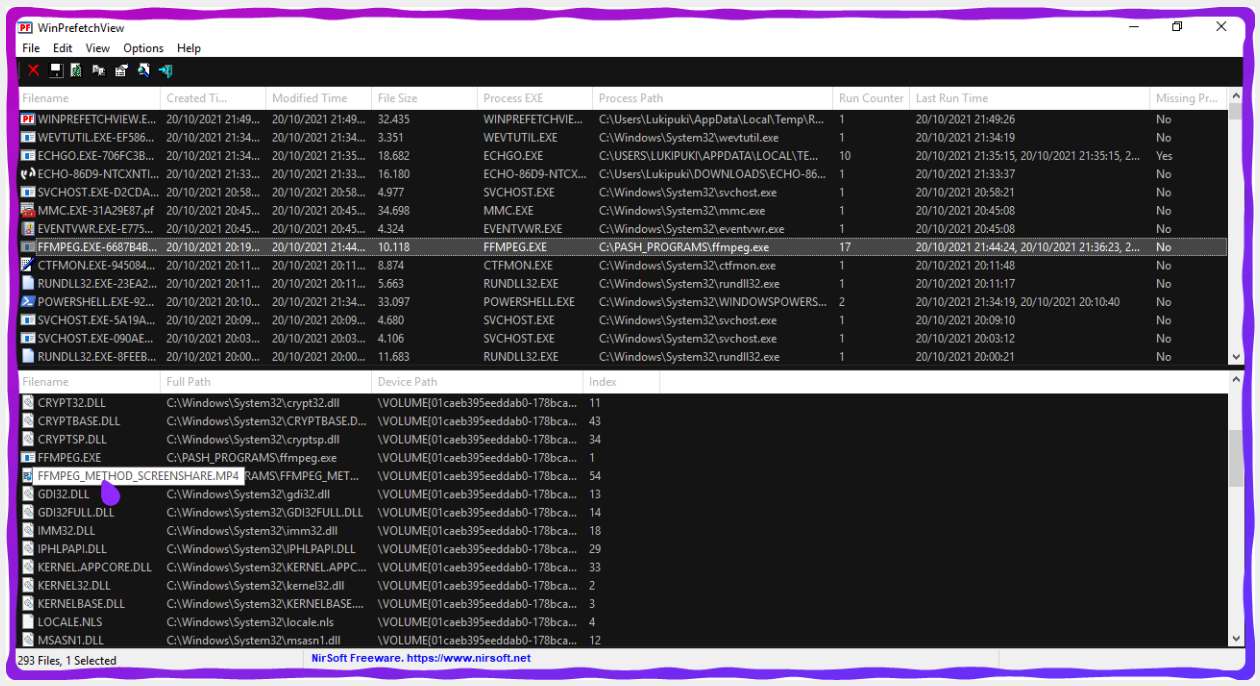
[https://github.com/Amadeus411/Red-Lotus-buster/releases/tag/Minecraft]



WinPrefetchView

Analyzing prefetch traces is the easiest with the help of WinPrefetchView. Can be downloaded from [View the content of Windows Prefetch (.pf) files (nirsoft.net)]
Last Activity Viewer can be used as a side-tool with WinPrefetch, since it also checks data from registry entries. LAV can be downloaded from [LastActivityView - View the latest computer activity in Windows operating system (nirsoft.net)]

- WinPrefetchView is split into two separate windows. The top part - which shows a prefetch trace and the bottom half - which shows the loaded contents of the prefetch trace. It can show if a recorder is currently in use and so on.

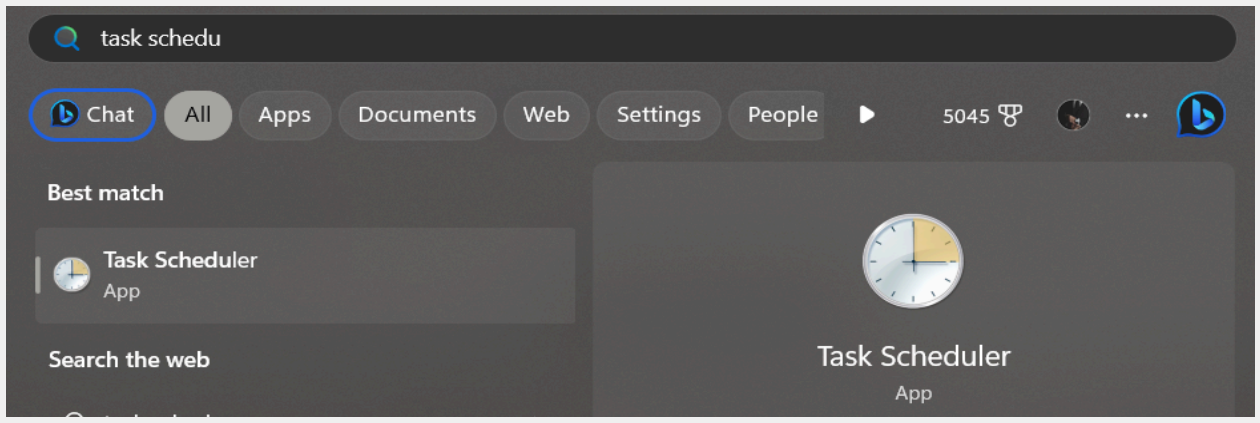


- It can also prove executions, simply analyze prefetch traces like DllHost, Conhost, Consent, even FiveM's prefetch trace can show what files were loaded into the game - in case they're placed in the FiveM's directory. I recommend you check at least 10 before & after prefetch traces on FiveM (10 traces before fivem and 10 traces after)

Task Scheduler

(Explanation by - Requiem)

Task Scheduler is a built-in feature in Windows operating systems that allows users to schedule tasks or programs to run automatically at specified times or on specific events.



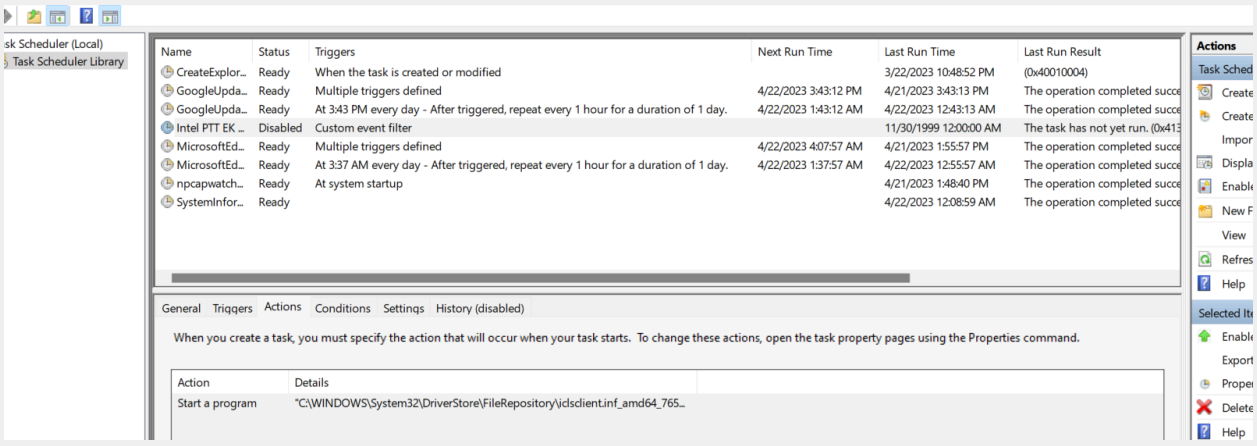
Since tasks can be started at boot time, it may bypass string logging.

To make a detection, we will consider two cases:

The task is currently in the system, not deleted:

1. We can detect its presence by analyzing every task in the Task Scheduler program, and checking it "Actions" parameters

This way, we can check if the file is a cheat or not, and you may only check tasks that are executed at boot/logon time

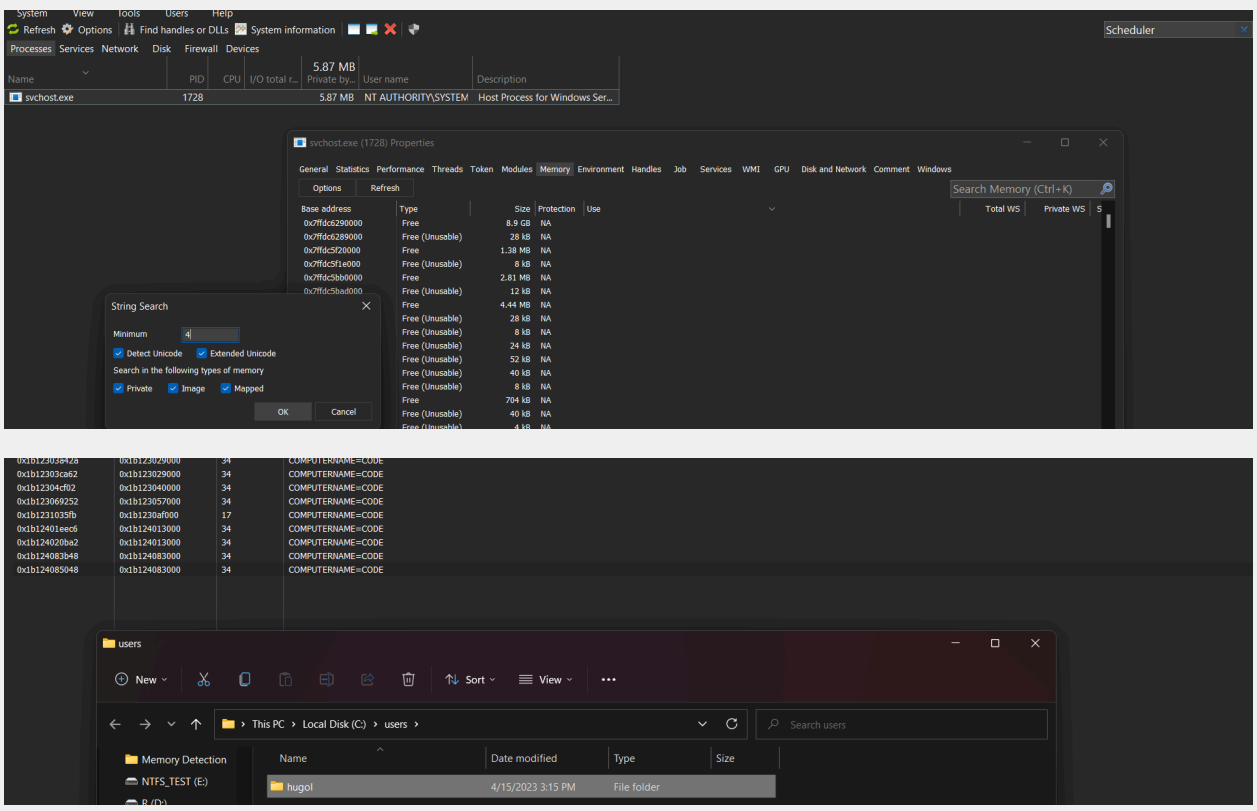


2. We can detect its presence by analyzing files logged by the Scheduler process, or xml artifacts registered when the task was made.

To detect artifacts traces, we may filter this string:

COMPUTERNAME=\Windows Username

Replace Computername= with the value that you get in the Scheduler process after filtering "computername=, and Windows username with the current username being used, that you can get in the task manager or in the C:\Users path



To detect files executed by Schedule, [we may filter](#) this regex (contains case-insensitive):

`([A-Z]:\\.+\.(dll|exe)|"[A-Z]:\\.+\.(dll|exe)")$` - Look for suspicious files, for example .exe files from directories other than C:\Windows\System32;

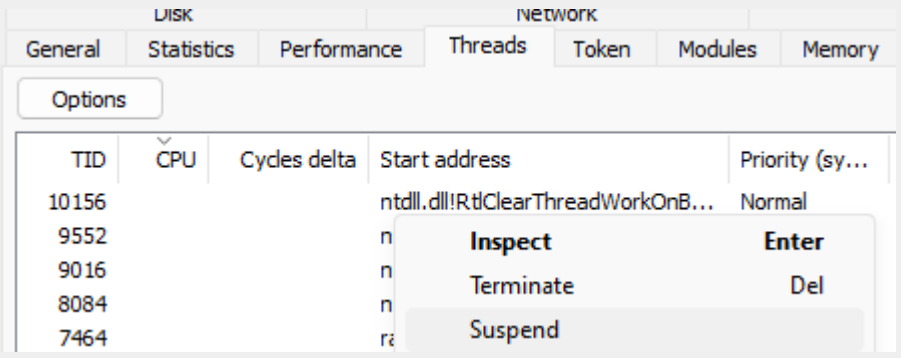
- Windows tasks are located in C:\Windows\System32\Tasks directory, use JournalTool to detect deleted task names if the above methods are too complicated, but you'll still need to perform them.
- After reading what task was deleted before / near SS, use System Informer and filter the Scheduler by the name of that specific task, copy the results in a .txt and read </Command>

Suspended services & threads

PC Optimizers, FPS Boosters, Custom Operating Systems stop useful services that are important for ScreenSharing. Here's a list of actually bannable services:

- **Pcasvc** (Responsible for Program Compatibility Assistant; Pcaclient)
- **SysMain** (Responsible for Prefetch)
- **DPS** (Used to detect renamed extensions, executions, two different string softwares)
- **CDPU_(RandomNumbers)** (Often used to determine external drives)
- **EventLog** (Event Viewer service)
- **DcomLaunch** (Responsible for logging .jar executions)
- **Activities Cache** (Tracks applications, copy pastes, app switches and so on)

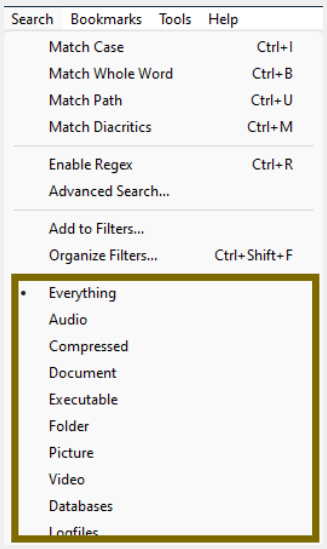
And sometimes, users only suspend **threads** of these Services to not make them stop but also log anything.

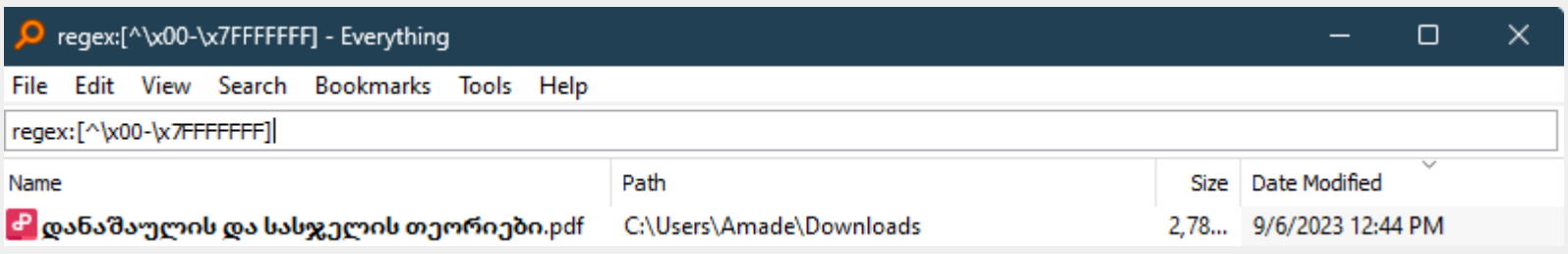


All of this is directly linked to accessing process hacker / system informer if done by a user. That's why we ban for using it. **They can also restart services before ScreenShare.**

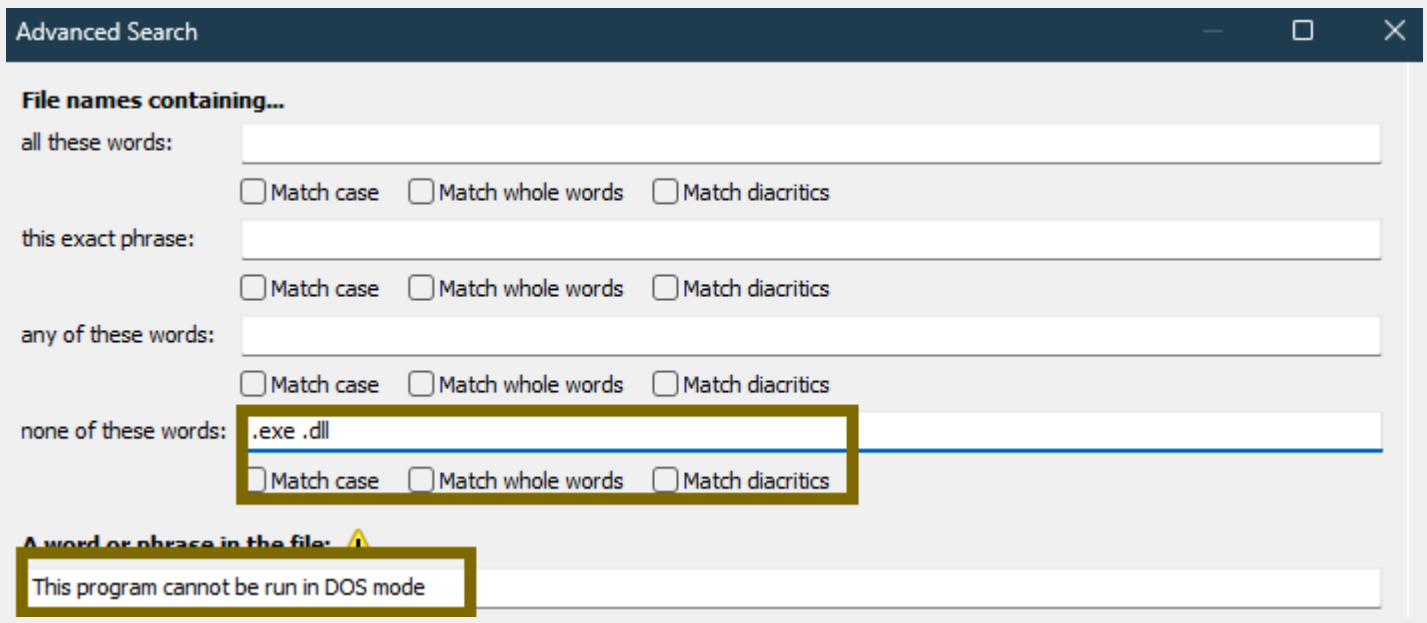
Search Everything

Search everything will be the most useful tool you can utilize for ScreenSharing, it is basically a 100 times better version of the default windows search. The ScreenShare Community has shared a regex for ASCII Character files, they've made a Software as well as this regex pattern to find it all. **regex:[^\x00-\x7FFFFFFF]**

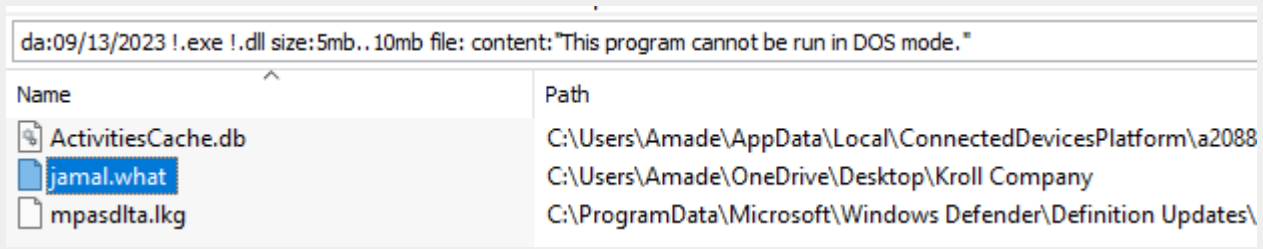




By clicking "search" you can select to look for specific file types only. That can be used to select "Folder" to check the last folder update dates, especially if they're suspected of using version clients, like Wolfram, Wurst. . .
If you press **ALT+P** you will enable previews, that will allow you to see the contents of the files you click on.
Search Everything also allows us a string-based search in files, which my recommendation is to only utilize for detecting renamed .exes and .dll files by going to Advanced Search:



Or you may simply use this in the search itself:
da:[currentDate] !.exe !.dll size:5mb..10mb file: content:"This program cannot be run in DOS mode."

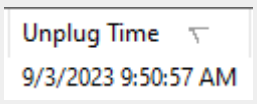


USBDriveLog

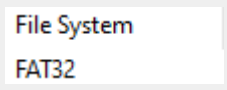


USBDriveLog is probably a better version of USBDeview, that has been being used by the community for quite some time now. Unlike USBDeview sometimes, USBDriveLog shows:

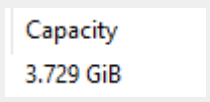
- **Last unplug time**



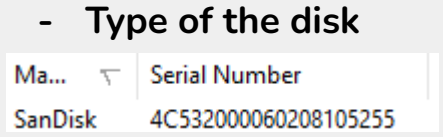
- **File system**



- **Capacity of the disk**



- **Type of the disk**



It is also capable of identifying and detecting virtual disks with these actions.

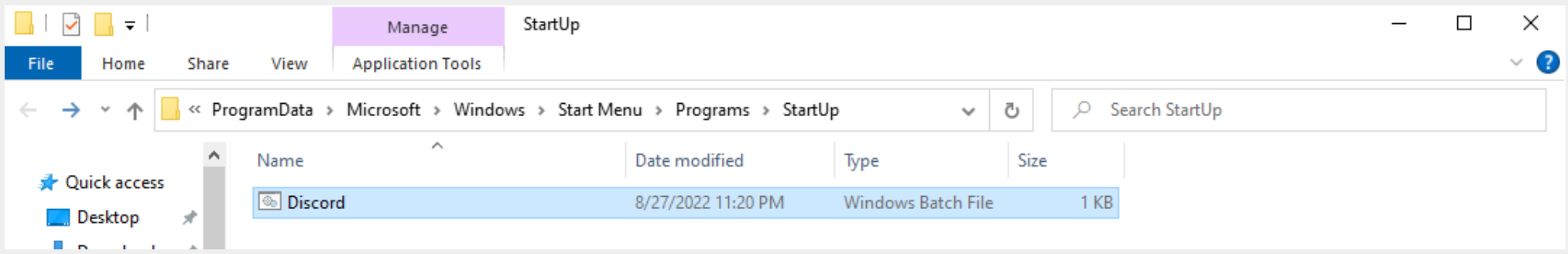
Run it as Administrator.

X-Mouse button control & Startup items

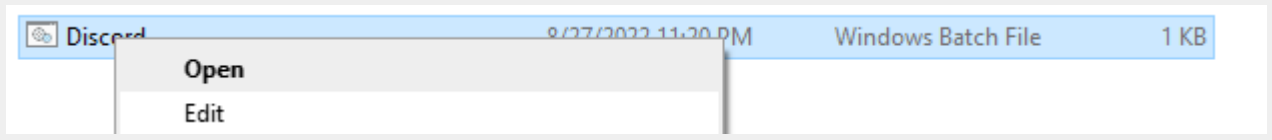
This'll teach you everything you need Regarding X-Mouse button & Startup items, since X-Mouse downloads and sets as a Startup program.

To check for custom added startup items, go to

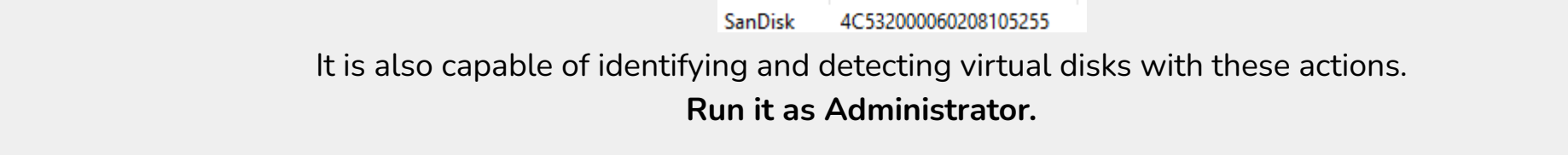
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp

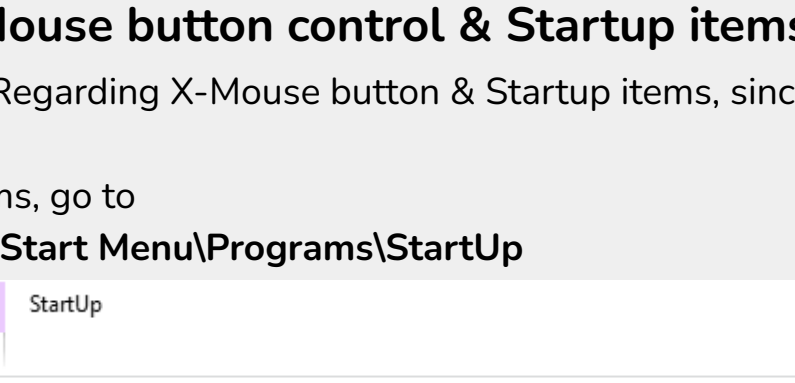


Right click on the .bat file and select Edit

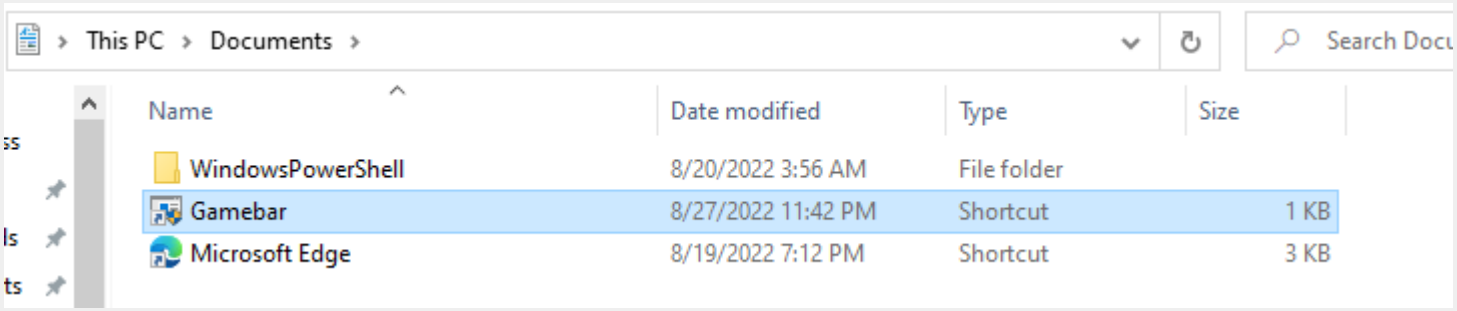


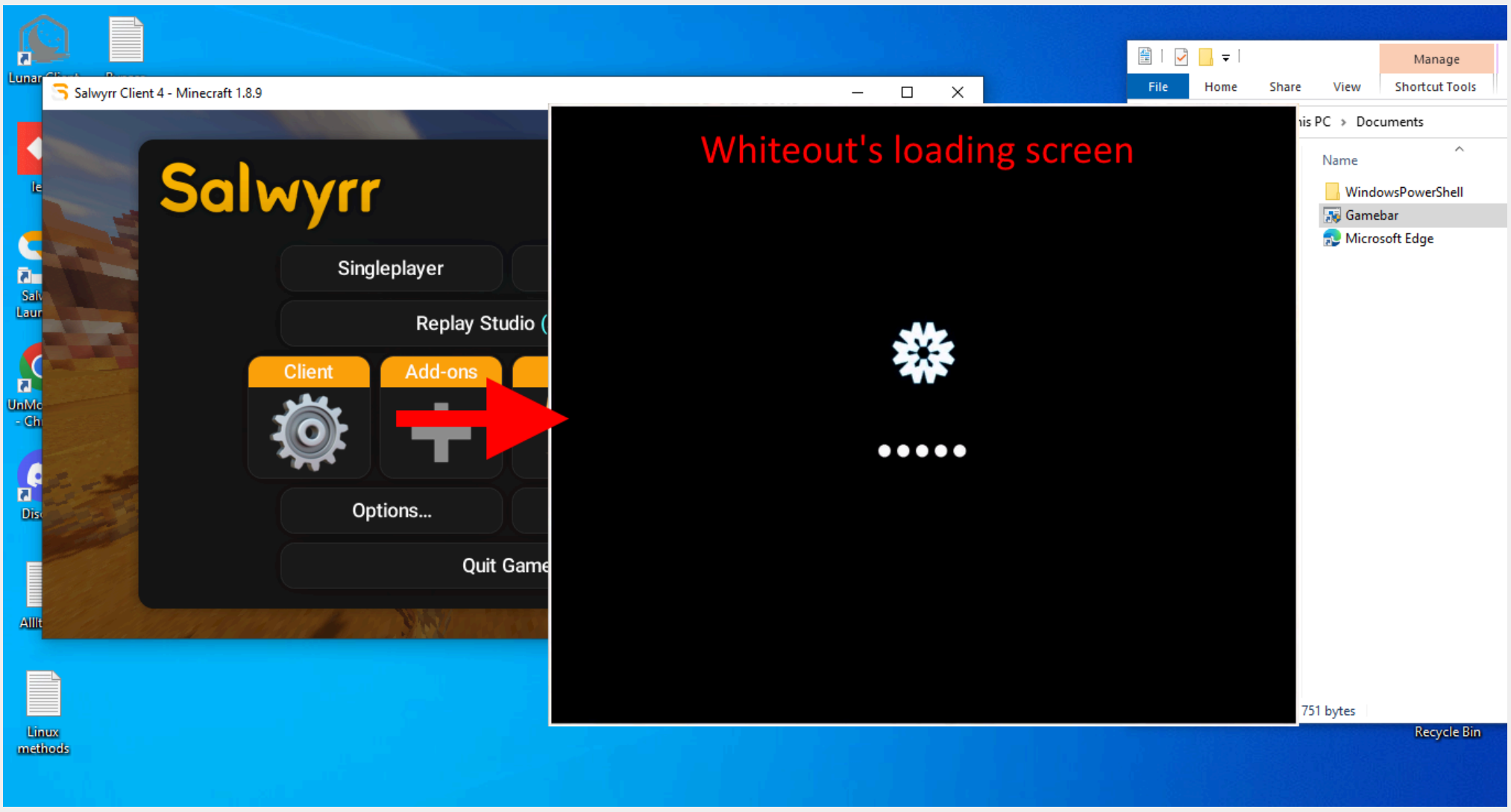
Analyze the results of the txt file





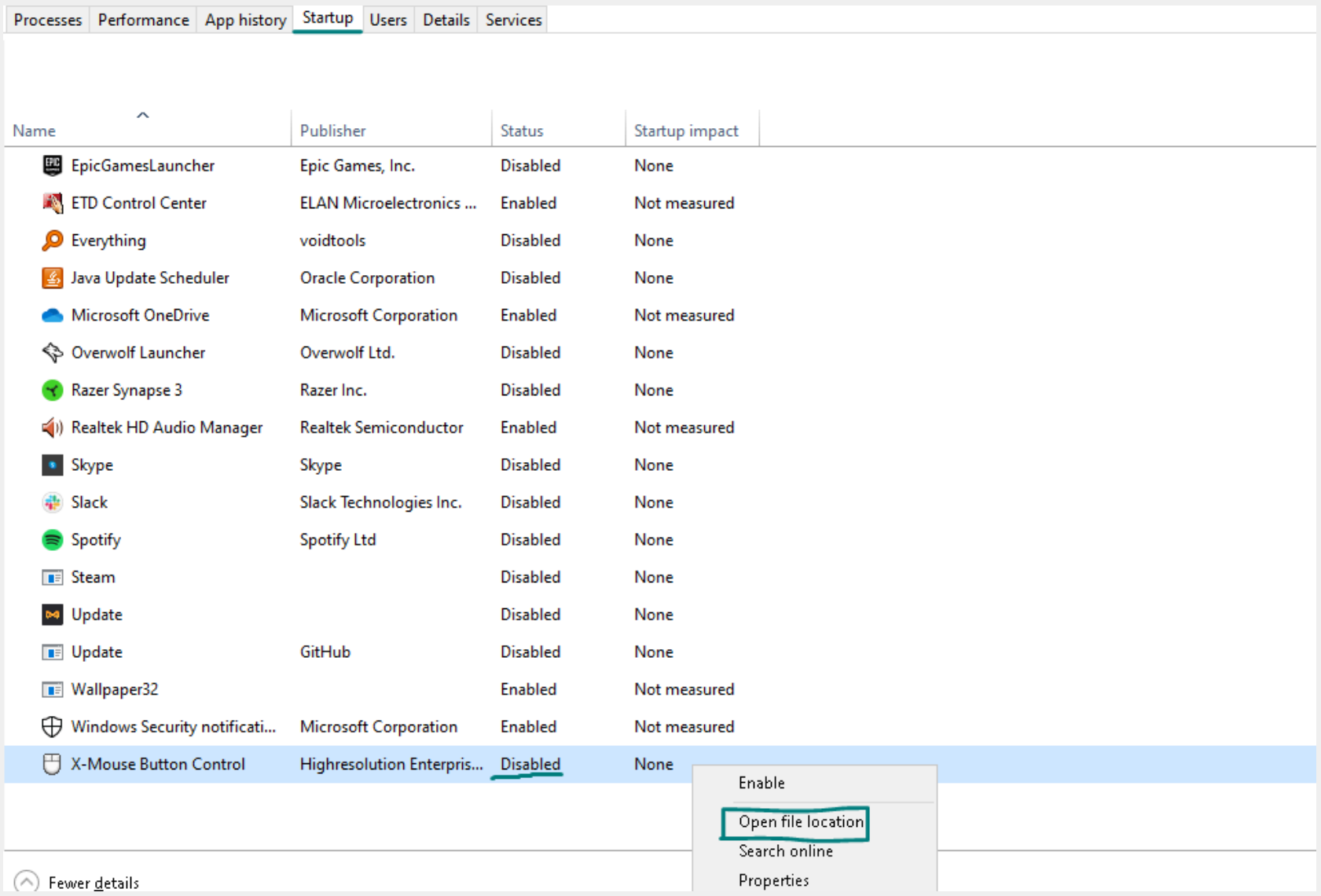
Go to the directory it shows (after cd)
Check the file by opening it.



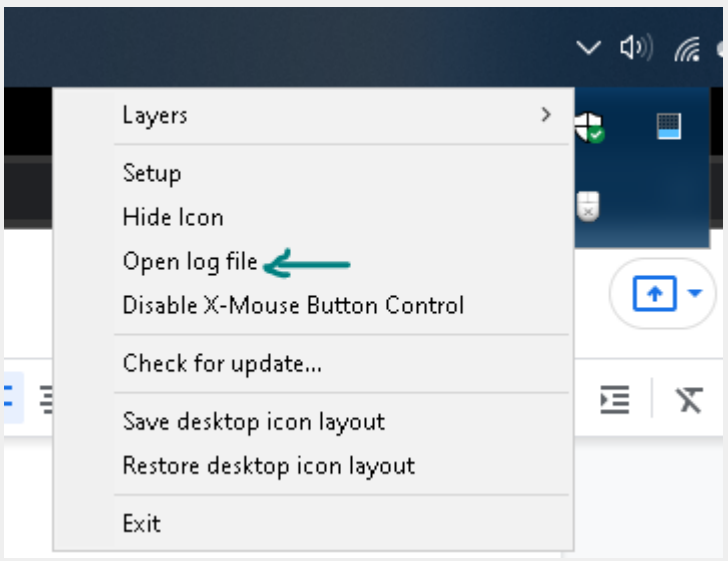


The reason you may ban for this is that clients like Vape lite, Astolfo, Whiteout etc usually wait to detect Minecraft and it lets you inject. Which means, they can inject 5 hours after the computer has started up to try and bypass with it.

If you froze someone based on Suspicious Clicks, or someone came to you for a Console ban, check for Startup items from Task Manager very first since the X-Mouse button can be Enabled.



Click on “Open file location” and run X-Mouse Button Control in case it’s Enabled and Run X-Mouse manually.



Open hidden icons again and Right Click on X-Mouse control, Open log file and check its logs.

```
**********************************************************************************************************************************
17-12-2021 00:47:27.760> X-Mouse Button Control v2.19.2 (x64) Startup. Commandline ''
17-12-2021 00:47:27.776> Running in high integrity mode (0x00003000)
17-12-2021 00:47:27.792> Loaded 0 application specific profiles (0 normal and 0 custom window).
17-12-2021 00:47:27.792> Running on Microsoft Windows 10 Professional Edition (build 18363), 64-bit, Hook Timeout: 200 ms
17-12-2021 00:47:27.792> Warning: Low Level Hook Timeout is low. If you have problems try increasing this value.
17-12-2021 00:47:27.792> Startup folder: 'C:\Program Files\Highresolution Enterprises\X-Mouse Button Control\'
17-12-2021 00:47:27.792> Settings (and log) folder: 'C:\Users\Noobking\AppData\Roaming\Highresolution Enterprises\XMouseButtonControl\'
17-12-2021 00:47:27.823> HookThreadProc[0x183C]: Using keyboard language 0x04090409
```

Concentrate on the time. If X-Mouse control was closed Before ScreenShare, as well as if its change logs were applied before SS, ban for X-Mouse button.

```
17-12-2021 00:49:33.196> CMyMessageWnd::OnSetup    -   Opening setup dialog.
17-12-2021 00:49:33.274> Loaded 0 application specific profiles (0 normal and 0 custom window).
17-12-2021 00:49:40.718> Applying changes to X-Mouse Button Control settings...
17-12-2021 00:49:40.770> Loaded 0 application specific profiles (0 normal and 0 custom window).
17-12-2021 00:49:45.874> Applying changes to X-Mouse Button Control settings...
17-12-2021 00:49:45.896> Loaded 0 application specific profiles (0 normal and 0 custom window).
17-12-2021 00:49:47.477> CMyMessageWnd::OnSetup    -   Setup dialog closed.
17-12-2021 00:49:47.488> Loaded 0 application specific profiles (0 normal and 0 custom window).
```

Setup Dialog already means they have changed settings of X-Mouse. Loaded 0 Application means they've also edited settings. If anything was edited/modified before SS - Ban for X-Mouse button, if nothing was edited/modified but they closed X-Mouse before SS, don't ban.
If they already have X-Mouse open when you enter Anydesk, just check for the Log file, if anything was modified before SS, ban for X-Mouse.

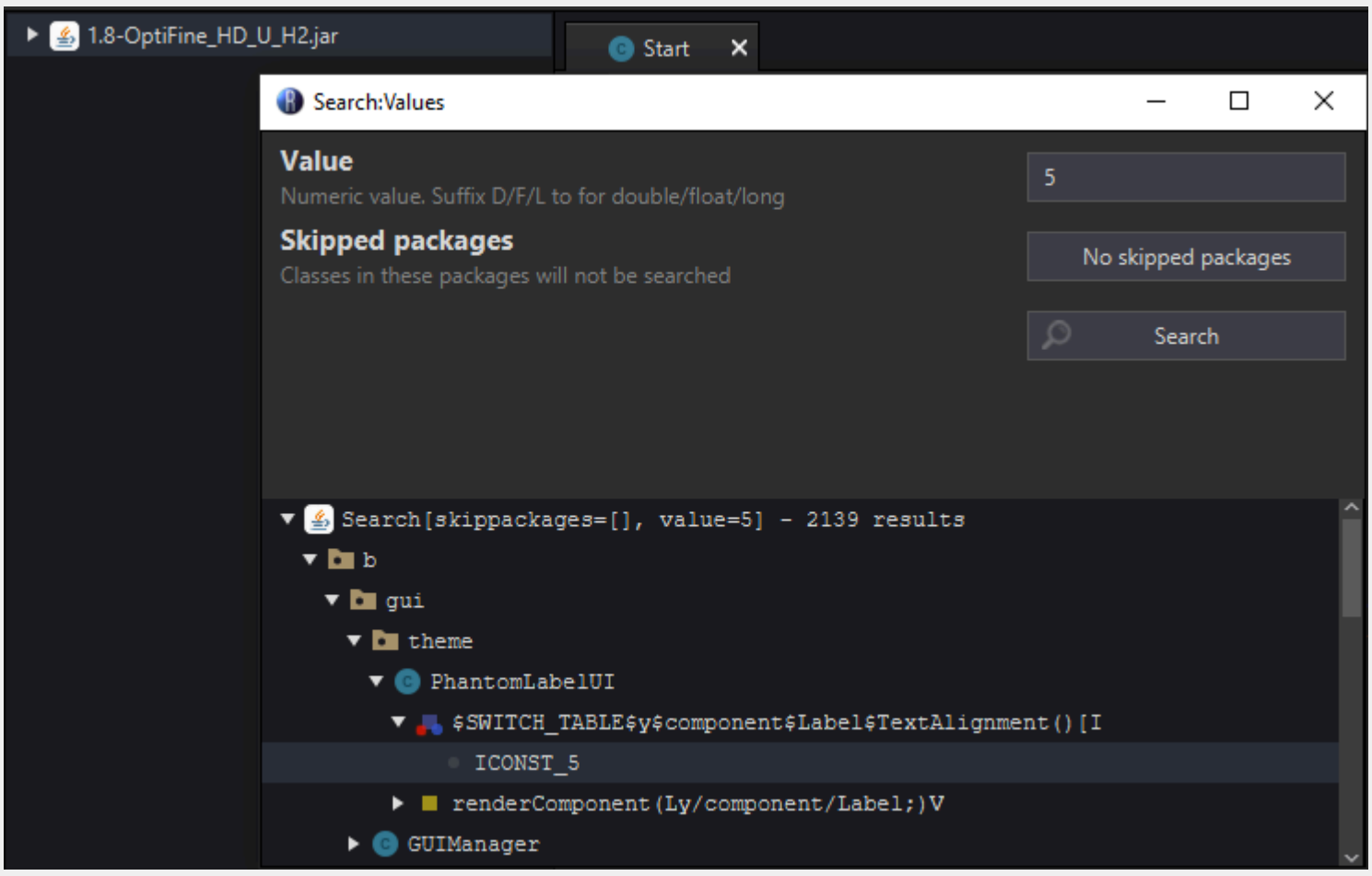
If you scroll all the way up and see that there is no Installation date shown in X-Mouse logs,

```
**********************************************************************************************************************************
28-11-2021 18:58:29.141> X-Mouse Button Control v2.19.2 (x64) Startup. Commandline '/installed /notportable'
28-11-2021 18:58:29.234> Running in high integrity mode (0x00003000)
```

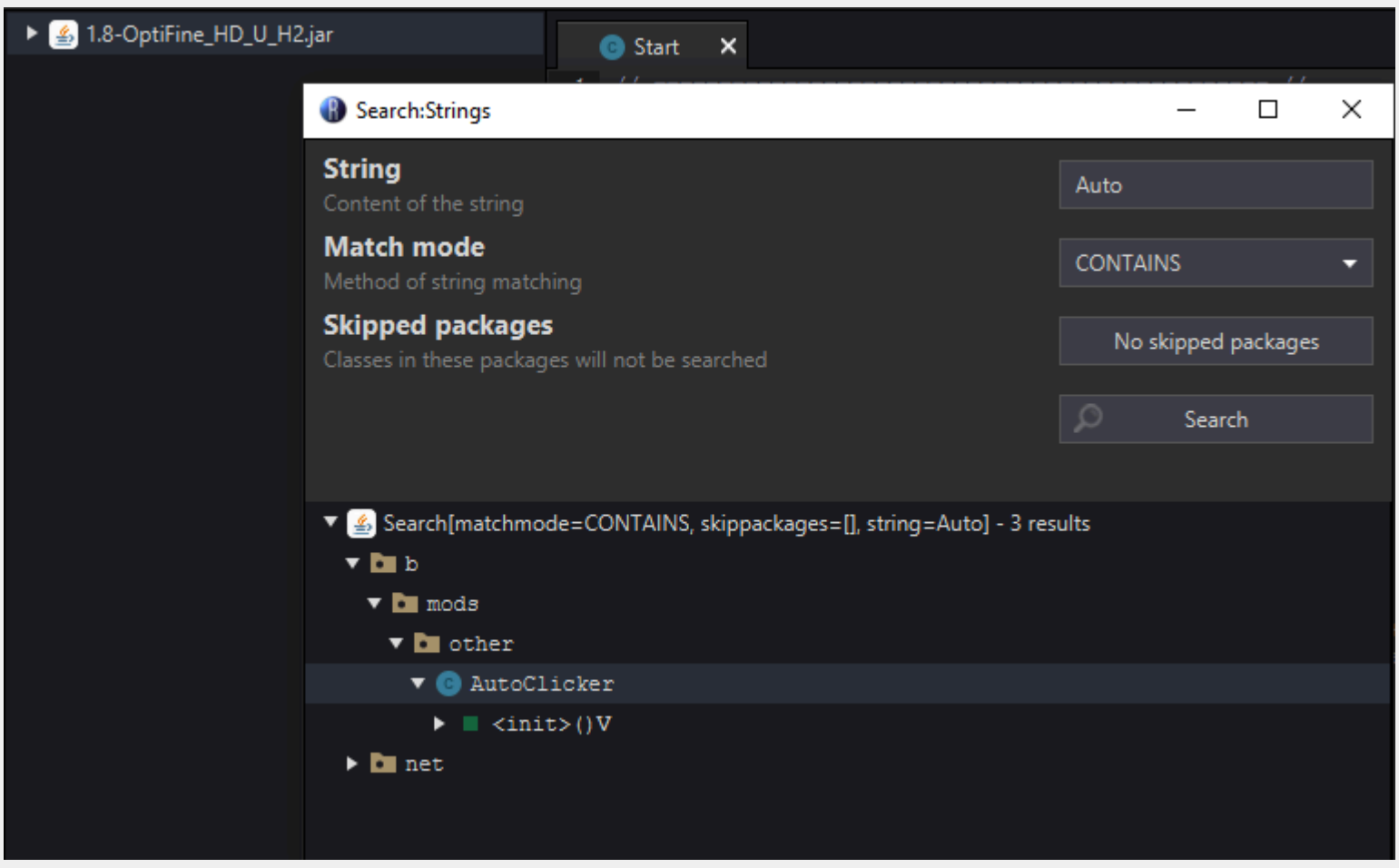
It means they've deleted/cleared X-Mouse entirely. Make sure to use Fsutil to check when they deleted X-Mouse, if it was before SS or after their Console ban for example, feel free to ban.

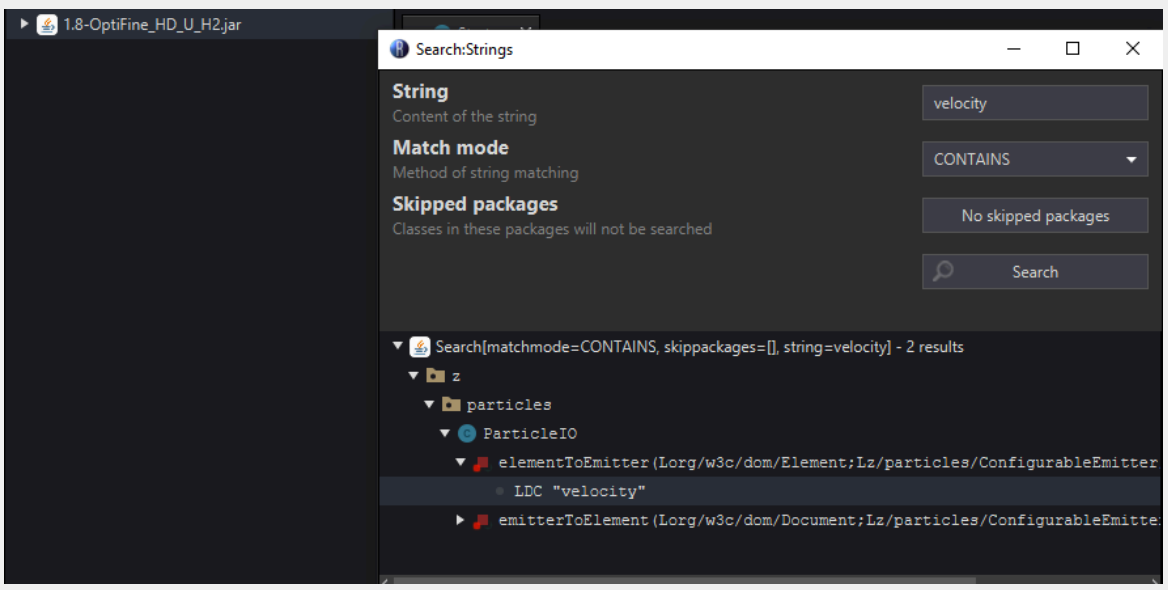
Mod Clients & .Jars

To find mod clients generally, as well as to analyze .jar files further, use recaf. Use the Search Values function, filter “5”

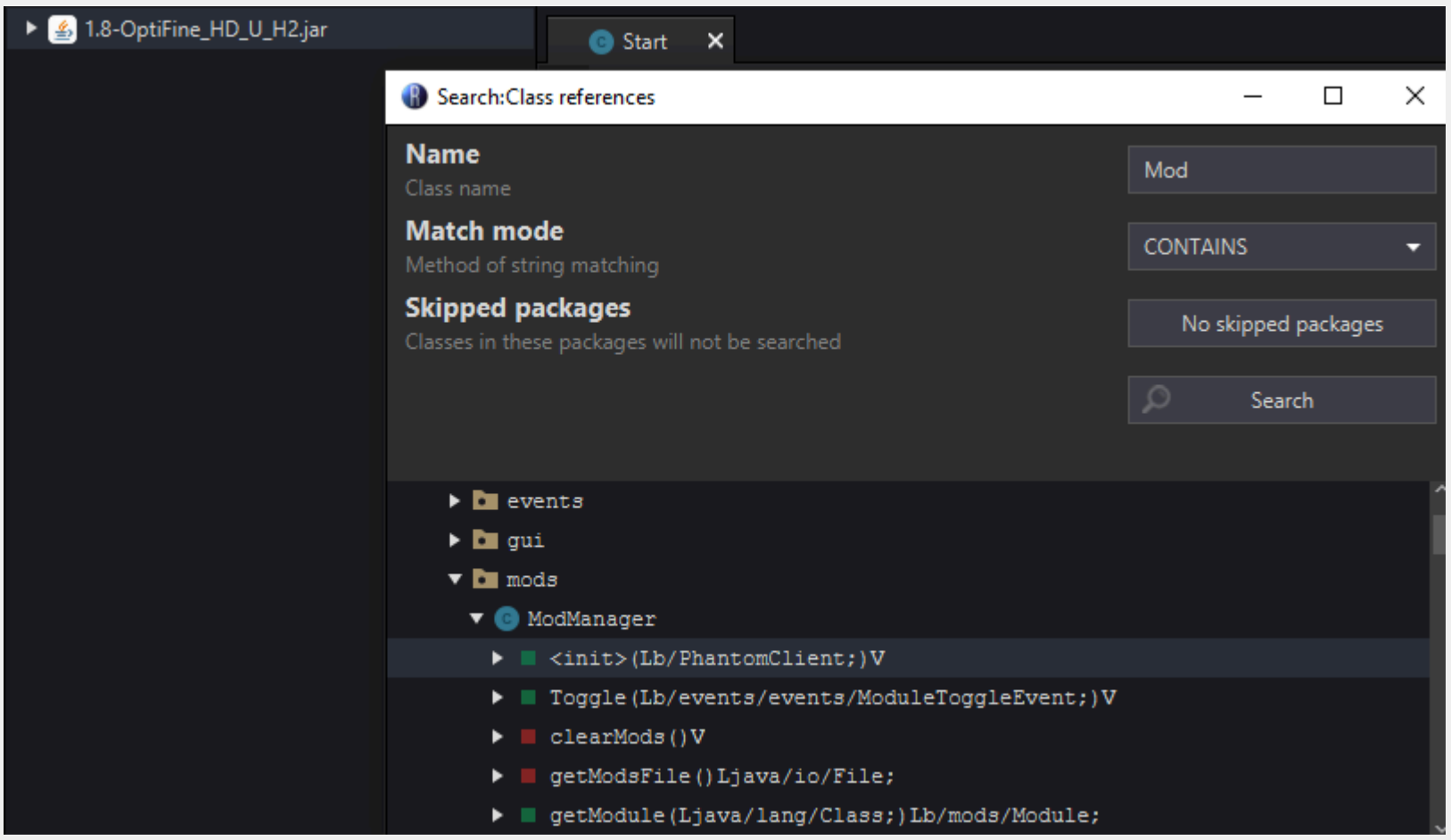


Alternatively, use the Strings function and filter for things like “Auto”, “Hitbox”, “Reach”, “Velocity” etc





You may as well use “Class references” and filter “Mod”

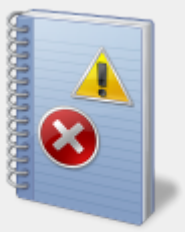


Analyze everything given.

|++++|—Second Phase—|++++|

If nothing was found by following the First Phase steps, assuming the ScreenSharer has paid attention to every small detail, there's a slight chance that the Player isn't cheating - or they're using some sort of a bypass method to hide most traces. In this phase, we'll cover Event Logs, Task Scheduler, Search Everything, System Informer, Maceta, OSForensics, and SherlockFinds.

Event Logs



EventViewer is a Windows tool that stores troubleshooting logs, in other words, they store traces of errors, crashes, system modifications and more. Players might clear USN Journal, Change system Time, mount and dismount virtual disks, restart their services... Or, their Explorer might crash & restart itself which is when we'll be able to use EventViewer to make sure it wasn't done by the Player to avoid false bans!

Deleted USN Journal, Modified System Time, Virtual Disks, Restarted Services

I am glad to tell you that RLU has created a Powershell script that will check if Event Logs Service is running, parse and output results with dates of deleted USN journal, modified system time, virtual disks, USB activity, restarted services, user Logon Time, powershell downloads (that don't leave traces in their downloads history) and event logs cleared!

- Copy this whole script and you may keep it to paste in Powershell.

```
$EventIDs = @(3079, 1102, 7, 7036, 7045, 1001, 1002, 104, 1074, 4663, 7036, 4688, 4660, 4616,)
$EventLogs = @("Application", "System", "Security", "Windows PowerShell", "Windows PowerShell Operational", "Setup", "ForwardedEvents", "Internet Explorer")
$Results = @()
foreach ($EventLog in $EventLogs) {
    $Events = foreach ($EventID in $EventIDs) {
```

```
    Get-EventLog -LogName $EventLog -InstanceId $EventID -Newest 3
  }
  $Results += $Events
}
$Results | Export-Csv -Path ".\RLUEvents.csv"
```

To understand what you're looking at, you must understand what these numbers stand for.

3079 - USN Journal has been deleted.

1102 - Audit logs (event logs) have been cleared.

7 - Unexpected system shutdown (PC died due to low energy or it has been shut down by the user)

7036 - Modified service state (started, stopped)

1001 - Service has crashed

1002 - Service crash too

104 - User logon

1074 - System shut down by user

4663 - Folder permissions change

4688 - Executed a script

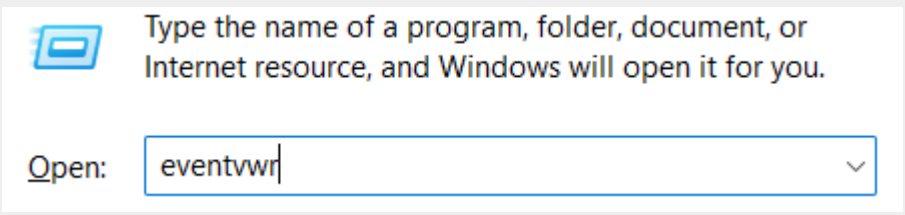
4660 - File's permissions menu was accessed

4616 - System time change

The output is in csv so use Timeline Explorer. [Here's a Video Demonstration] of the Script. Don't lose your time while the script is in progress, do some regular SS checks after minimizing the CMD window and get back to the output later.

Even though the script automates several checks, there are still Kernel-PnP Configurations and Management logs we'll need to check, assuming the user has mounted a Virtual Disk, skip otherwise.

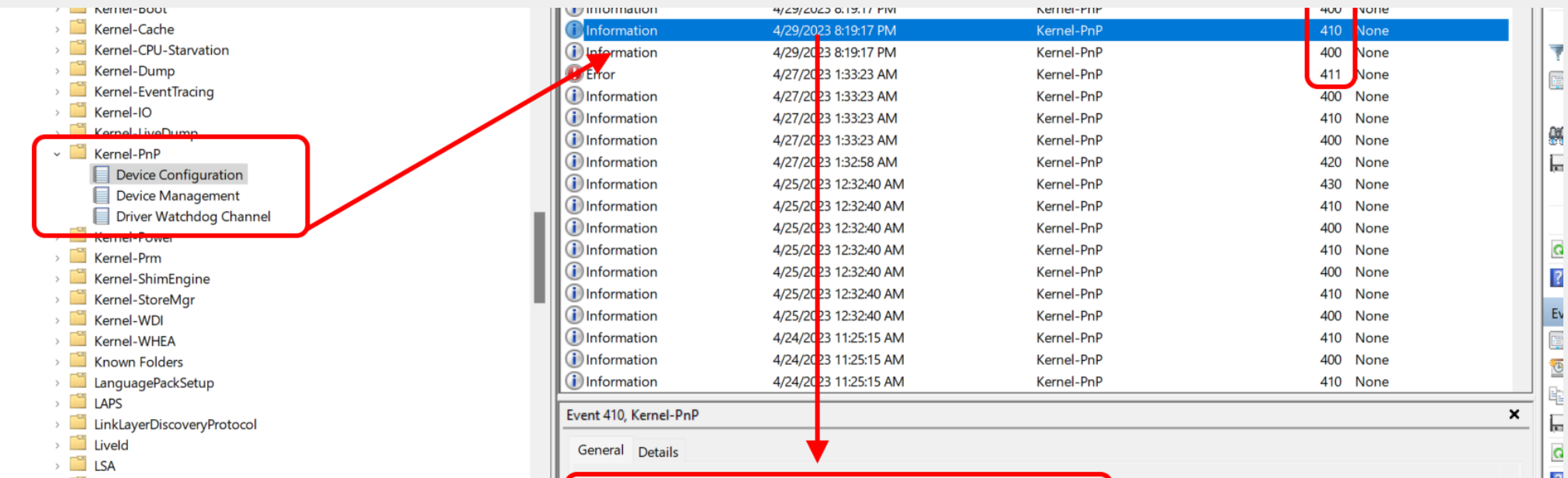
Press Win+R and type Eventvwr



Go to Applications and Services Logs > Microsoft > Windows > Kernel-PnP/Configurations

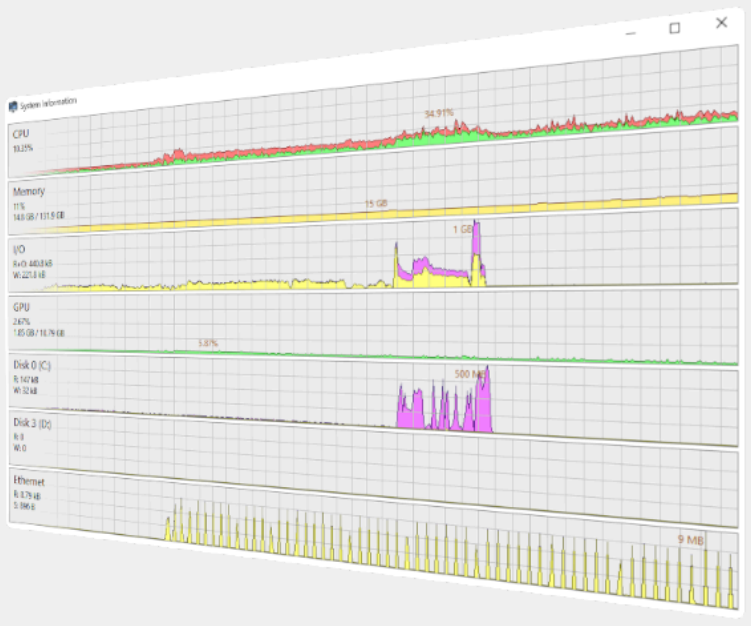


Specifically check event IDs of 400, 410, 411 that were recently generated (current date)



System Informer

System Informer (SI) is a new version of Process Hacker - As per say, it is “A free, powerful, multi-purpose tool that helps you monitor system resources, debug software and detect malware.”

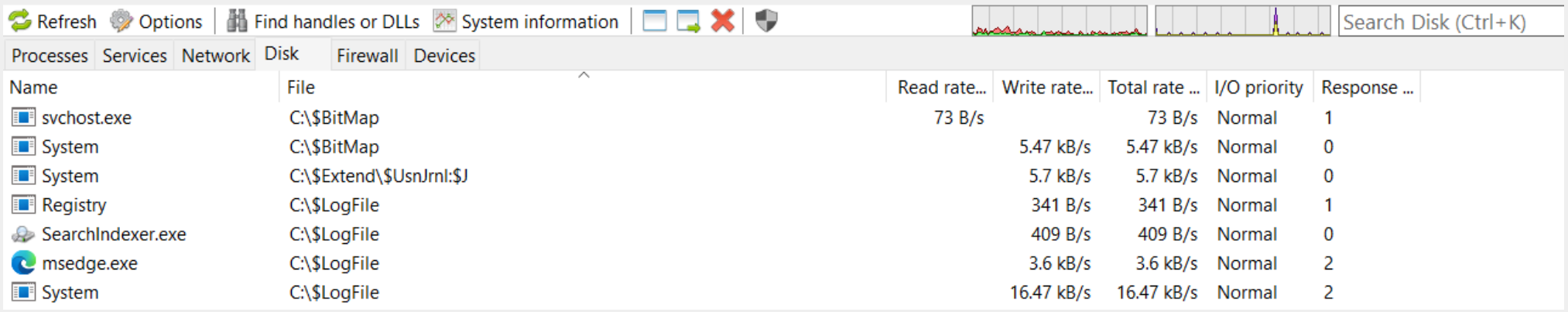


There are many alternatives to this tool that search for given specifics in services memory, such as strings64, bstrings, volatility and so on, however, none of them come close to the possibilities of this tool as all of them require a memory sample to be usable.

It is also the only version of Process Hacker that has the option to actually toggle Kernel Driver option on Windows11, although some of its functions mentioned in this guide might bluescreen the Player’s PC... (those methods will include a **!WARNING!** notice)

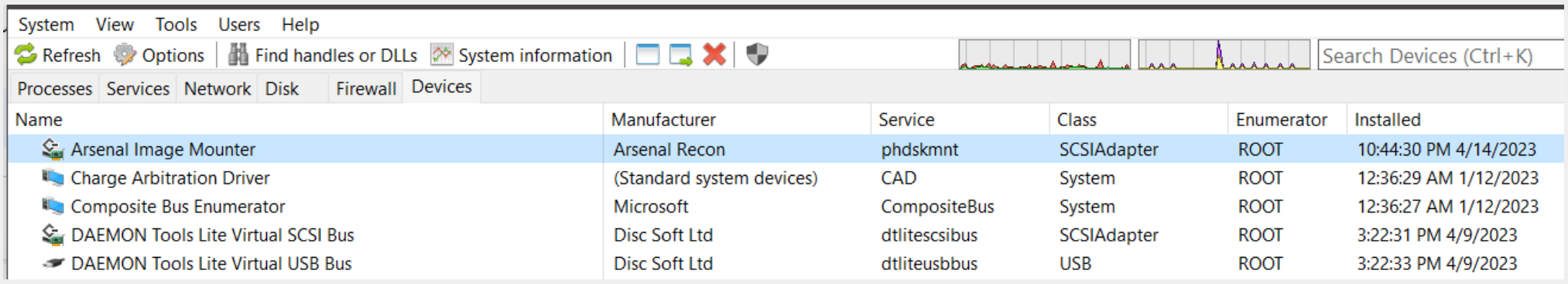
To toggle Kernel Driver on Windows, go to Options, select it and Close. You’ll need to restart SI.

The “Disk” option in SI will display currently working tasks, for example, recordings or background VPNs.



Name	File	Read rate...	Write rate...	Total rate ...	I/O priority	Response ...
svchost.exe	C:\$BitMap	73 B/s		73 B/s	Normal	1
System	C:\$BitMap		5.47 kB/s	5.47 kB/s	Normal	0
System	C:\$Extend\$UsnJrnl:$J		5.7 kB/s	5.7 kB/s	Normal	0
Registry	C:\$LogFile		341 B/s	341 B/s	Normal	1
SearchIndexer.exe	C:\$LogFile		409 B/s	409 B/s	Normal	0
msedge.exe	C:\$LogFile		3.6 kB/s	3.6 kB/s	Normal	2
System	C:\$LogFile		16.47 kB/s	16.47 kB/s	Normal	2

The “Devices” option will display all the currently plugged in Devices, it may also leave tracks of once recognized devices (such are Virtual disks or USBs).



Name	Manufacturer	Service	Class	Enumerator	Installed
Arsenal Image Mounter	Arsenal Recon	phdskmnt	SCSIAdapter	ROOT	10:44:30 PM 4/14/2023
Charge Arbitration Driver	(Standard system devices)	CAD	System	ROOT	12:36:29 AM 1/12/2023
Composite Bus Enumerator	Microsoft	CompositeBus	System	ROOT	12:36:27 AM 1/12/2023
DAEMON Tools Lite Virtual SCSI Bus	Disc Soft Ltd	dtlitescsibus	SCSIAdapter	ROOT	3:22:31 PM 4/9/2023
DAEMON Tools Lite Virtual USB Bus	Disc Soft Ltd	dtliteusbbus	USB	ROOT	3:22:33 PM 4/9/2023

We’ll be mainly concentrating on the Processes section, as that’s what we need to use. We’ll be using System Informer for the Task Scheduler section of this guide.

Anti-Scheduler



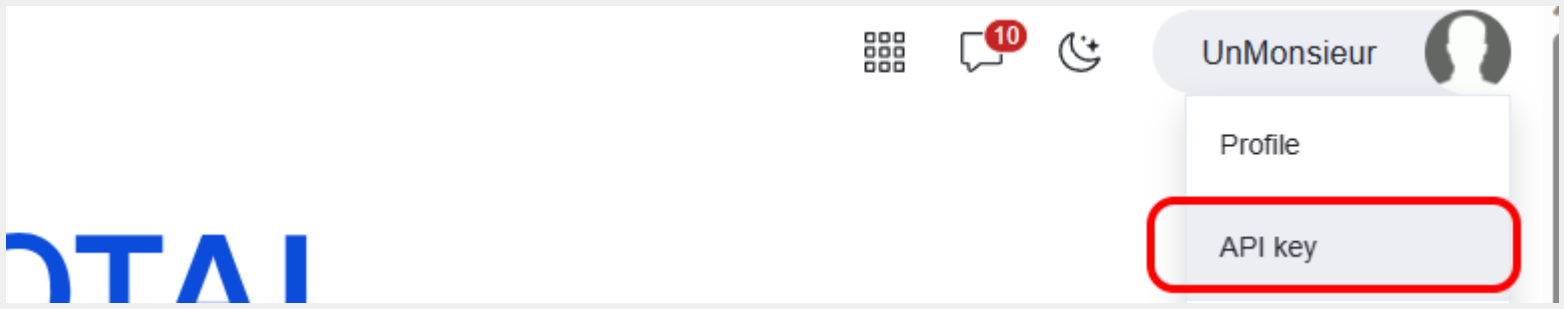
Red Lotus has released a Task Scheduler bypass detection tool, “Anti-Scheduler”, which scans all the scheduled events, calculates Entropy values (above 6> is most likely a virus = most likely a cheat) and outputs suspicious events that are, also most likely, set-up by the User.

- ❖ The tool will not work on files that include hidden characters in them, this bypass has caught popularity lately, in-depth review of this will be included in the “Red Lotus [Academy] Handbook, Tome II” release.

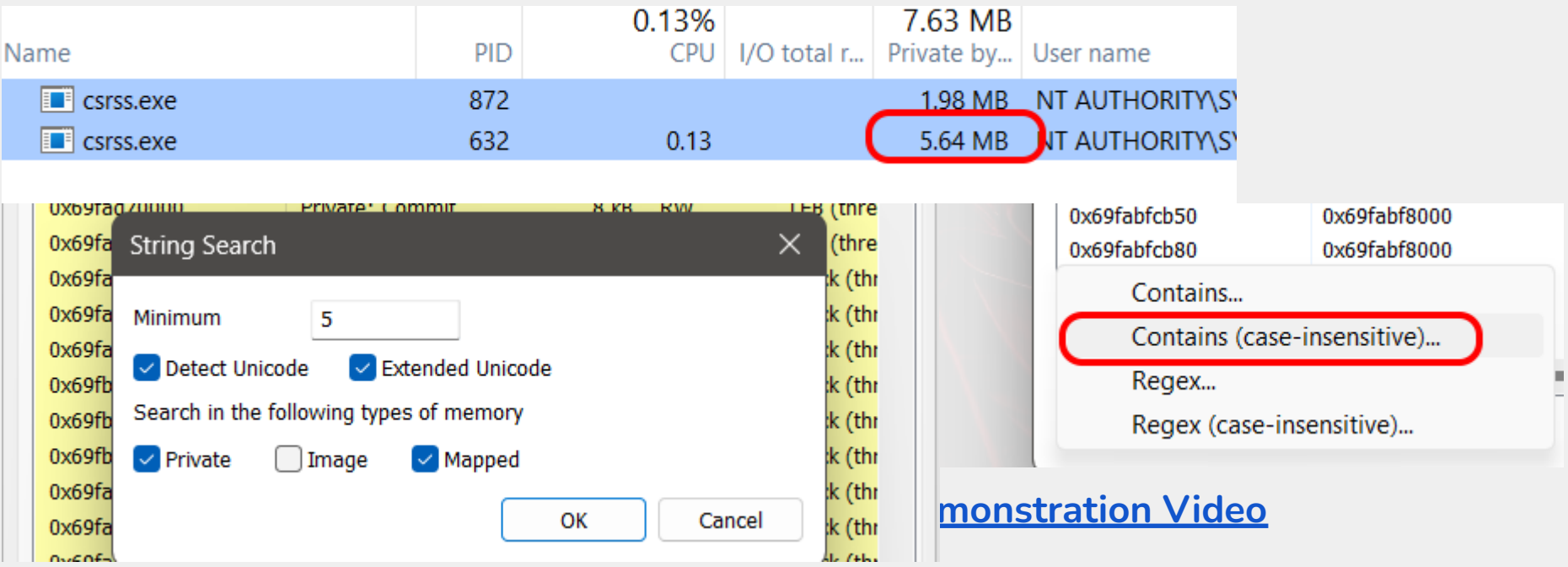
Maceta

Maceta is a SS Tool created by Rancio, it uses virustotal API to detect unsigned malicious executions by specified files. Let’s break it down:

- You may Download Maceta using this link: [https://github.com/RRancio/Exec/raw/main/Files/maceta.exe]
- Make a VirusTotal account and generate your API key



- Before executing Maceta, open System Informer and dump csrss with highest memory with strings: :\ and .exe | save results as a .txt.



[monstration Video

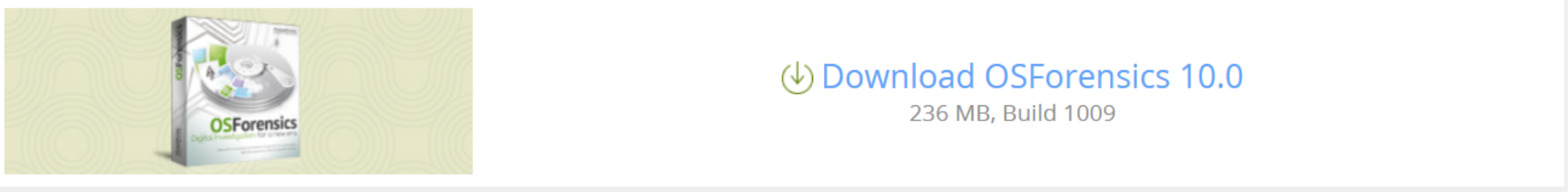
Maceta will output detected files in a folder & then it'll open the folder itself. Make sure you execute it as Admin.



OSForensics

▬Download▬

OSForensics - Download



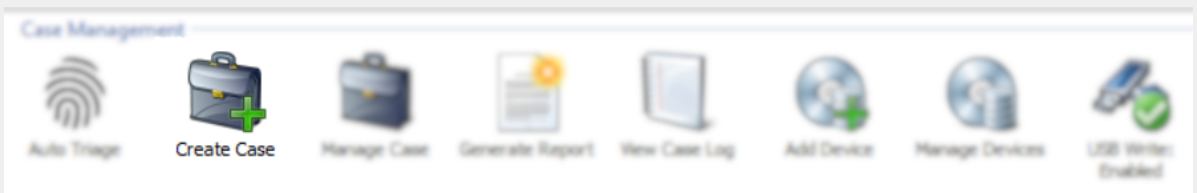
▪▪▬Introduction▬ ▪ ▪

OSForensics is a Forensic toolkit that has a Trial version available for FREE. The tool is ~240MB, which might be impractical in some cases but it's definitely a tool that has many uses. This guide will only cover some of the most useful features of the tool.

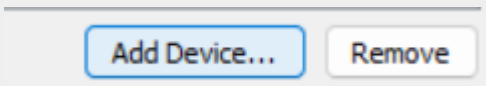
▬ ▪ ▬

[__Processing the Case

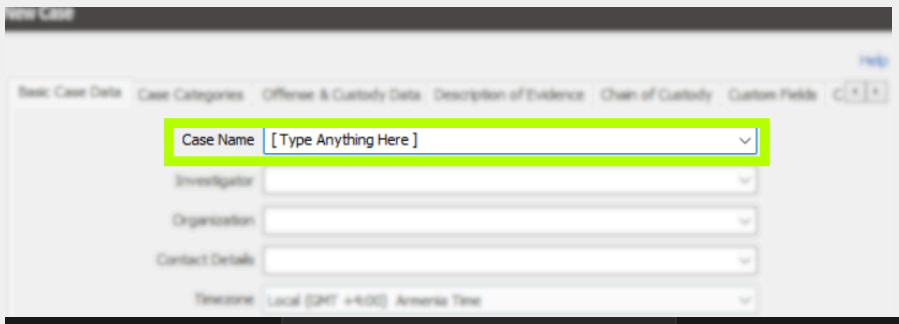
To start off, you'll need to create a new case instead of doing a Triangle Acquisition. To do so, navigate to the **Create Case** option under **Case Management.**



From here on, you'll be met with a menu to select case options. In Case Name you'll type anything as it plays no significant role in this.

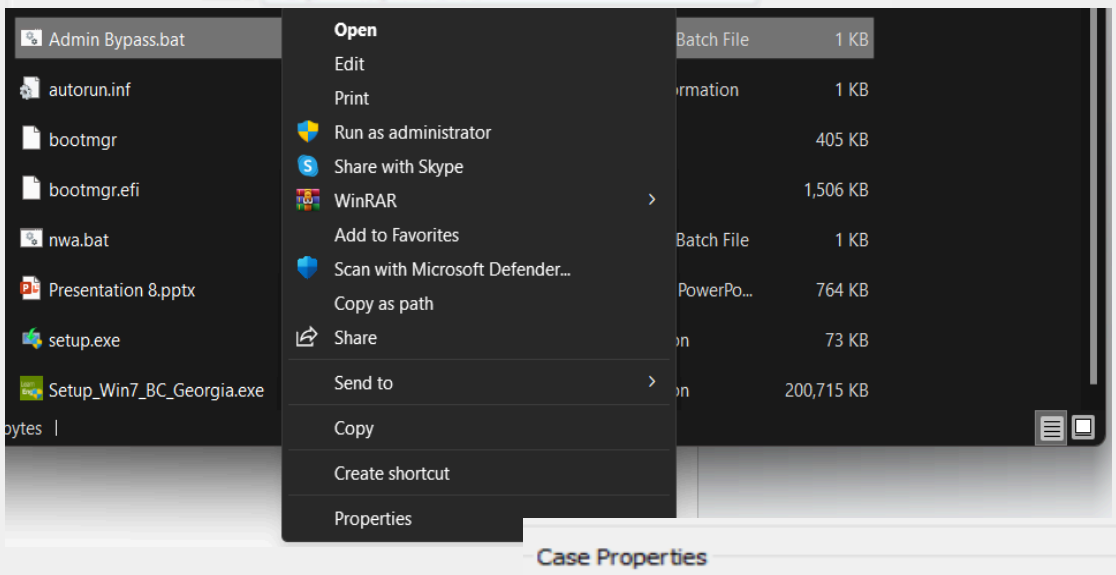


For Acquisition Type - You'll select **Investigate Disk(s)**

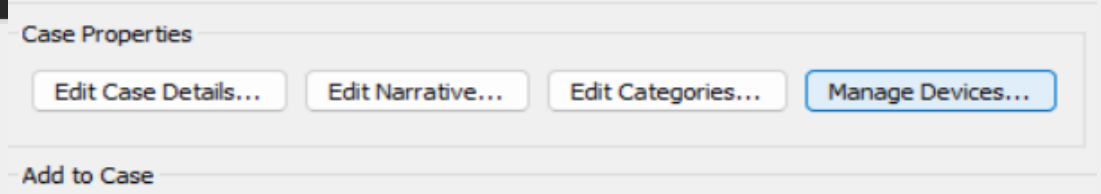


For Case Folder - You'll enable **USB Write-Block,** with this you'll rest assured that any external drive modifications **will not be done by you,** any data alteration / deletion is safe to ban for. Important Note - **Enable this only if the user has disconnected a drive** & ask them to reconnect it. Make sure it's the right drive number.

This way, any file on the drive will lose **"rename" / "delete"** options as the drive will be Write protected.



and add all the Drives connected to the



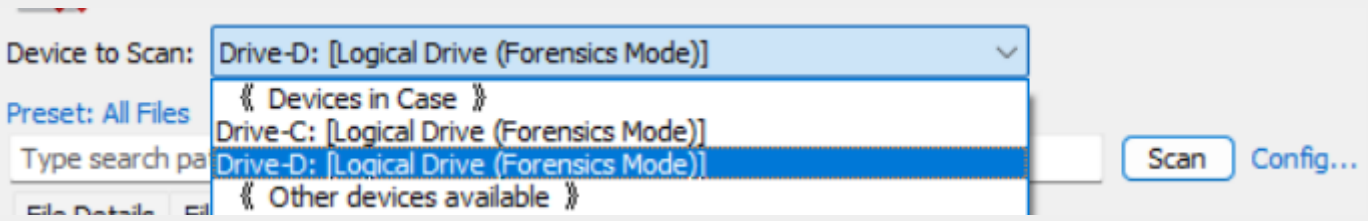
In **Case Properties,** select **Manage Devices** that are currently device.

[__File System Browser

After you've set up the case drives, you can easily navigate through the utilities without having to create a new case for each new drive entry. First, we'll be looking for **Deleted Files Search.**

⛓_____⛓

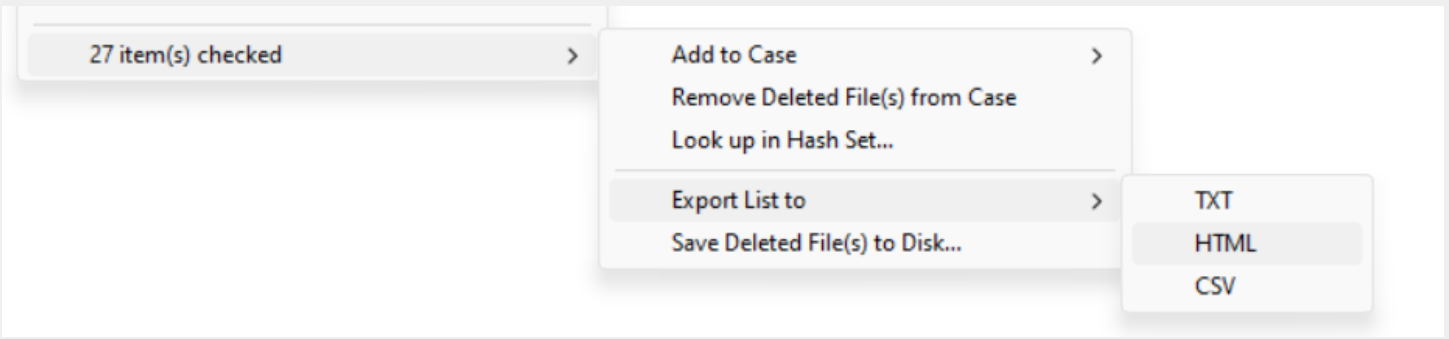




Select a **Device in the Case** (or Other devices that you need to parse) and press **Scan.**

After the Scan is finished, right click to a blank space and press **Select All**

Right click again and save selections as **HTML | Note - Only do this if you need to acquire recently deleted files that are .exe / .dll / suspicious enough.**



This HTML output will show an **estimated restoration quality,** if it's above 70% there's a high chance of successfully recovering them; otherwise a lower than 60~50% estimation might indicate multiple things, such as:

➖The deletion space has been overwritten (may indicate a replacement)

➖The file has been corrupted (may indicate an anti-forensic technique of file deletion)

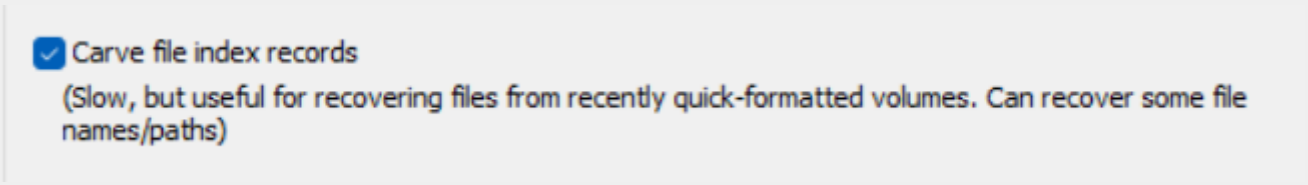
➖May indicate the file has been deleted ages ago, but a similar file has been put in its place

➖The deleted file has been scattered across the device (may indicate a fragmentation technique which makes it nearly impossible to recover them)

Keep in mind - You can filter to only look for deleted .exe files in the search patterns.

Configuration

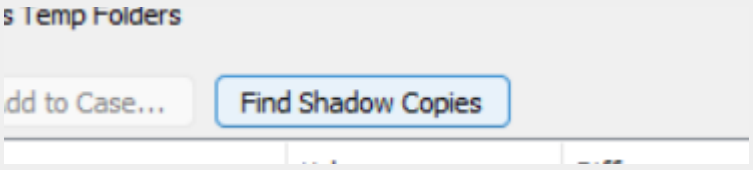
To configure Deleted Files Search, press **Config...** option next to the scan button. Here, you can choose to also scan **Index Records, File Size limits, Minimum Quality** of the files and so on. Note that usually, excellent quality will refer to fresh deletion (not overwritten / corrupted)



[__Volume Shadow Copies

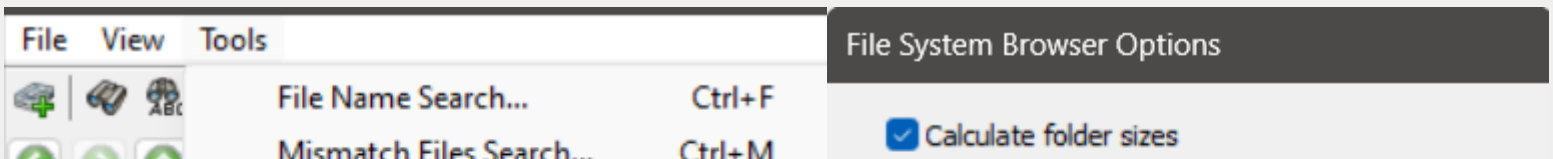
Analyze Volume Shadow Copies

While analyzing Volume Shadow Copies might be useless, sometimes they can be used to recover freshly deleted files (such as python scripts / bat / css scripts). Scroll slightly down and click **Analyze Shadow Copies.** Press **Find Shadow Copies** and reopen OSForensics. This time you should be able to see several VSS mounts to your active Case.

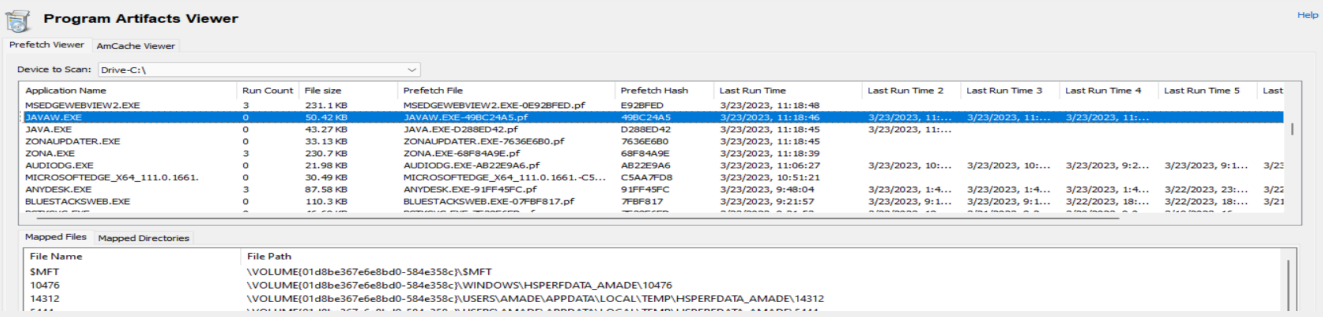


[__Program Artifacts Viewer

Program Artifacts Viewer is a Prefetch & AmCache data collector, This can practically be used if WinPrefetchView faces an error. Simply select



this option, select the drive and analyze the results.

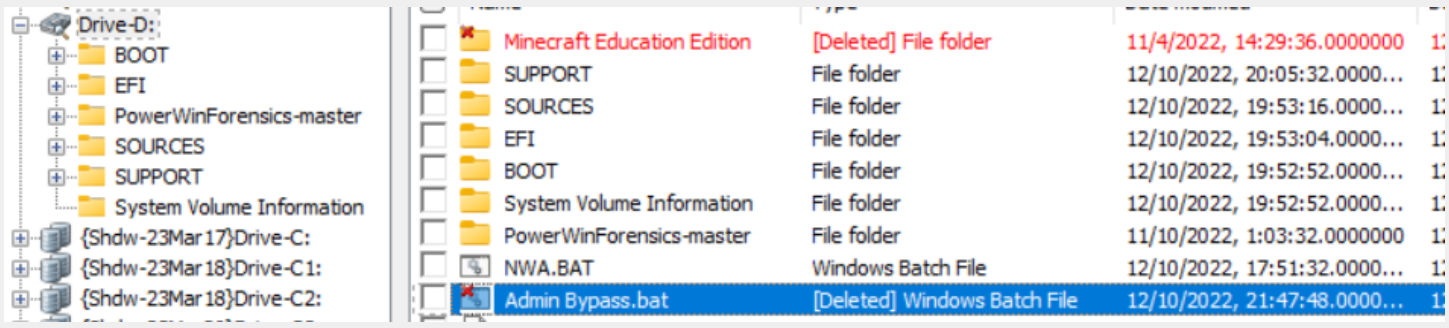


⛓_____⛓

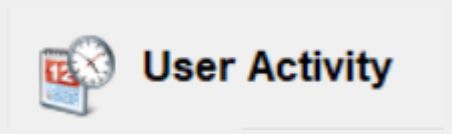
[__Deleted Files Search

File System Browser is basically more fun to use FTK Imager, with this, we should (in theory) be able to detect deletions on **FAT32 / external Drives.** But before you begin, go to File & Options to check everything for better results.

Select the drive (FAT32 in my case) you want to analyze and you'll see Deleted files with their estimated size. Note that this will not show a deletion date of the file but rather its Access date; if there's a cheat on the drive that has been recently accessed (you can use System Informer to see if the file has been executed since last reboot), the ban will be valid.



[__User Activity Timeline



One of the most useful features of OSForensics is a Timeline viewer. This isn't Timeline Explorer, it's an option that lets you view timeline data of the state of the device. In other words, you're simply able to navigate through **only** executables, **only** DLLs and **only** any data you're looking for. Most options have this setting, I'll be using User Activity to demonstrate it.

For this scan, recommended options are:

➖**Most Recently Used**

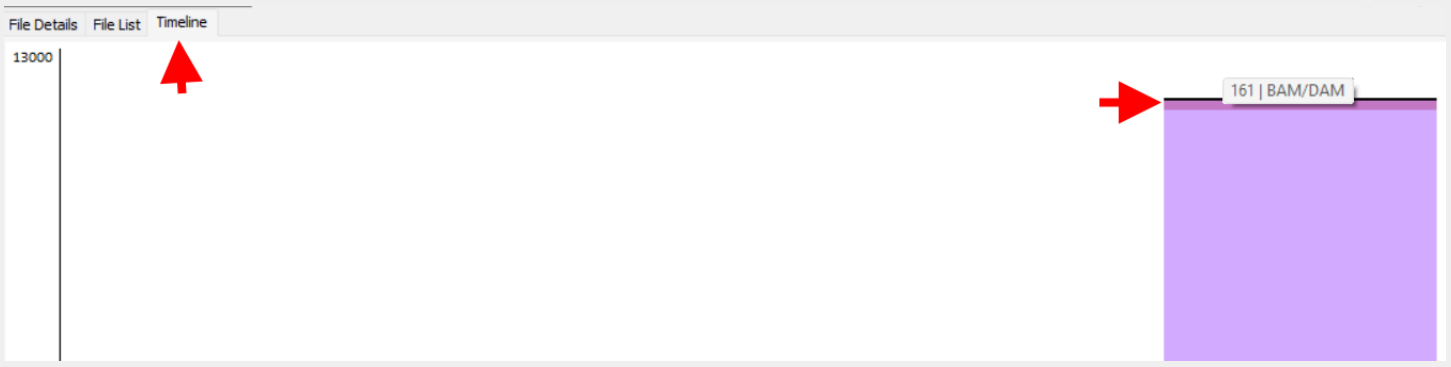
➖**Autorun commands**

➖**Clipboard**

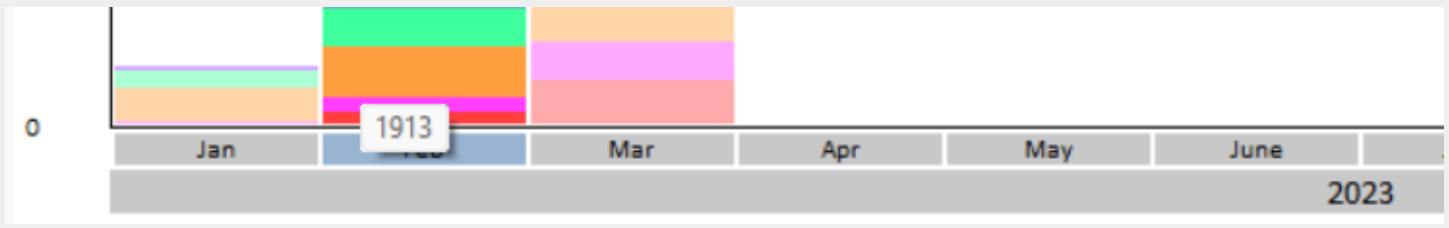
- **User Assist**
- **Jump Lists**
- **Shellbags**
- **Windows 10 Timeline**
- **Recycle bin**
- **Windows search**
- **Anti-Forensics artifacts**
- **BAM/DAM (DAM for Windows8)**
- **USB / Mounted Volumes**

Simply select a device and proceed to a **quick scan.**

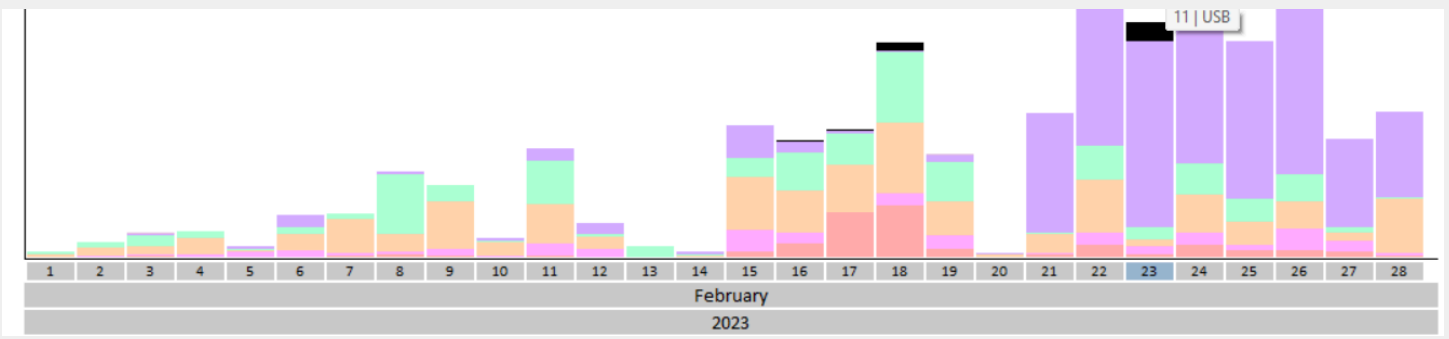
- - - - - Timeline - - - - -



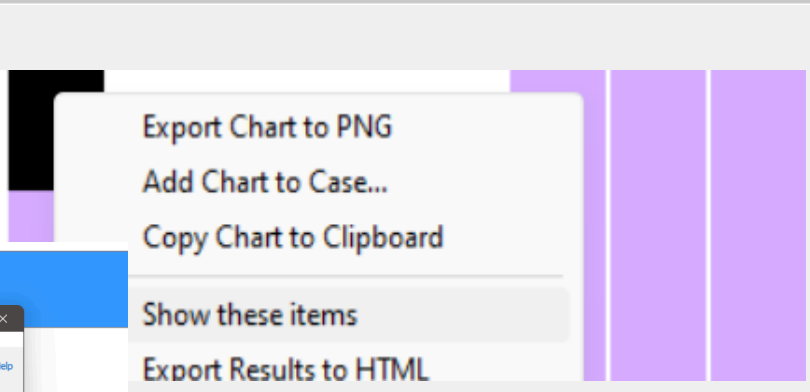
The Timeline output will display a chart of parsed output which you can navigate through by simply clicking on them.
The data you'll be redirected to will be more oriented on specific dates of the acquisition.



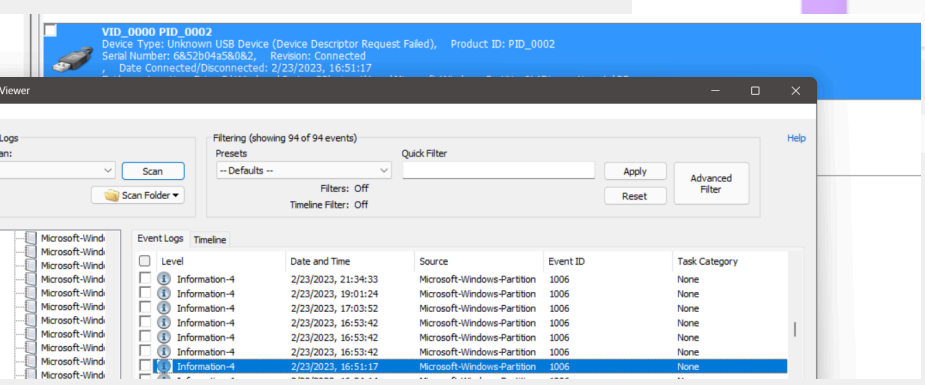
Clicking the same acquisition twice will spread the data into more specifics.



After you've gone directly to for, right click and press **Show** specific output only.



whatever data you're looking **these items** to be left with a



Sherlock Finds

SherlockFinds is a File Signature Checker for NTFS and FAT Drives, it displays Date of Modification on output results. The tool is made by the Red Lotus SS Team.

Note that the tool is getting updated from time to time, it detects unsigned files but the Date of Modification isn't sufficient. Thus we'd rather recommend you filter Directories to search (for example files in C:\Windows\System32) and so on.

- Download https://github.com/SherlockHolmesv/SherlockSignatureChecker/releases/tag/DFIR



Execute the file and wait. It'll output results in a .csv which you'll check in Timeline Explorer. Recommended filtering methodology;



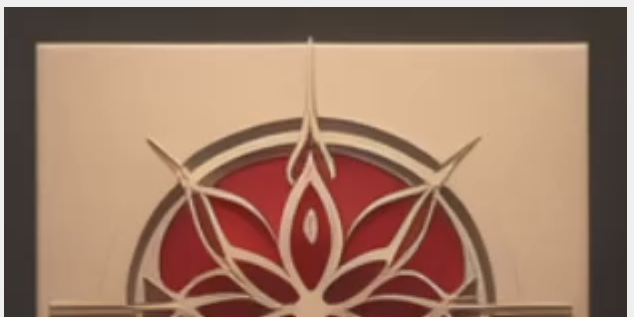
You may ignore filtering by the Dates at all and only use directories search.

It detects .jar and .bat files too.

|++++|—Third Phase—|++++|

This phase will include only the most useful manual SS-ing methods and tools, explained in mere details. It'll mainly cover a powershell toolkit.

Red Lotus Collector



RedLotus Collector is a DFIR Artifact Collector based on Eric Zimmerman

UnMonsieur | Red Lotus Manual (Public Collection)

Tools; the Collector is made by UnMonsieur,

Parse time takes quite some time, around 4~5 minutes

- Parses all Activities Cache entries that are found on the user
- Parses Jump Lists and .lnk files
- Parses Prefetch (Including Prefetch traces found in VSS)
- Parses **$MFT**
- Parses Regedit - Important registry entries such as TaskCache, USBStor, USB and more.

https://cdn.discordapp.com/attachments/1083164927696588921/1140744067077120110/RedLotus_DFIR.zip

A demonstration of this utility is in the official Red Lotus server, access level; or visit this link to view a clip of its usage. I recommend running it with a CMD instead of double clicking.

Velociraptor NTFS

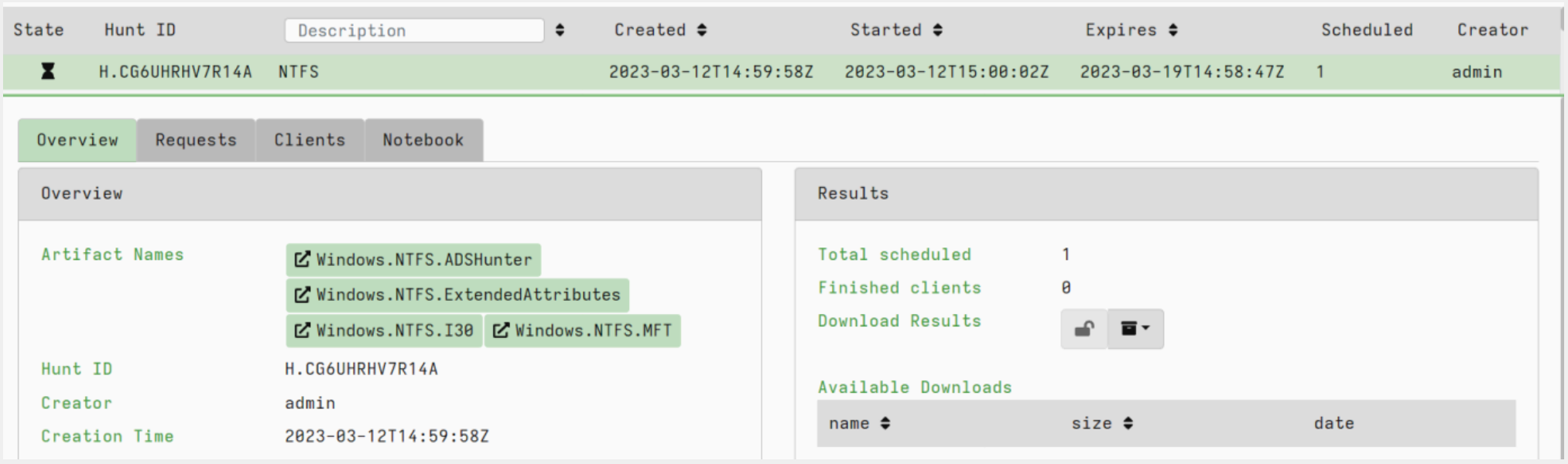
Velociraptor offers a compilation of NTFS parsers, with a combination of these 4:

Windows.NTFS.ADSHunter - Alternate Data Stream in NTFS

Windows.NTFS.ExtendedAttributes - Malicious Attributes trackdown

Windows.NTFS.I30 - $i30 Parse

Windows.NTFS.MFT - $MFT Parse



Note that it is recommended you either use only Windows.NTFS.ADSHunter & Windows.NTFS.ExtendedAttributes, or either one of the four since all 4 at once will drastically slow down the PC due to how much data it's parsing. For the tutorial, visit this link.

Rancio's Tools

Rancio has also made multiple extremely useful SS Tools to help you with ScreenShares; They can be viewed in his github repository. Here they are:

DoomsDay Client Finder

Executable file capable of detecting the DoomsDay client internally.

This was only tested on version 1.16.5, I have no idea about other versions.



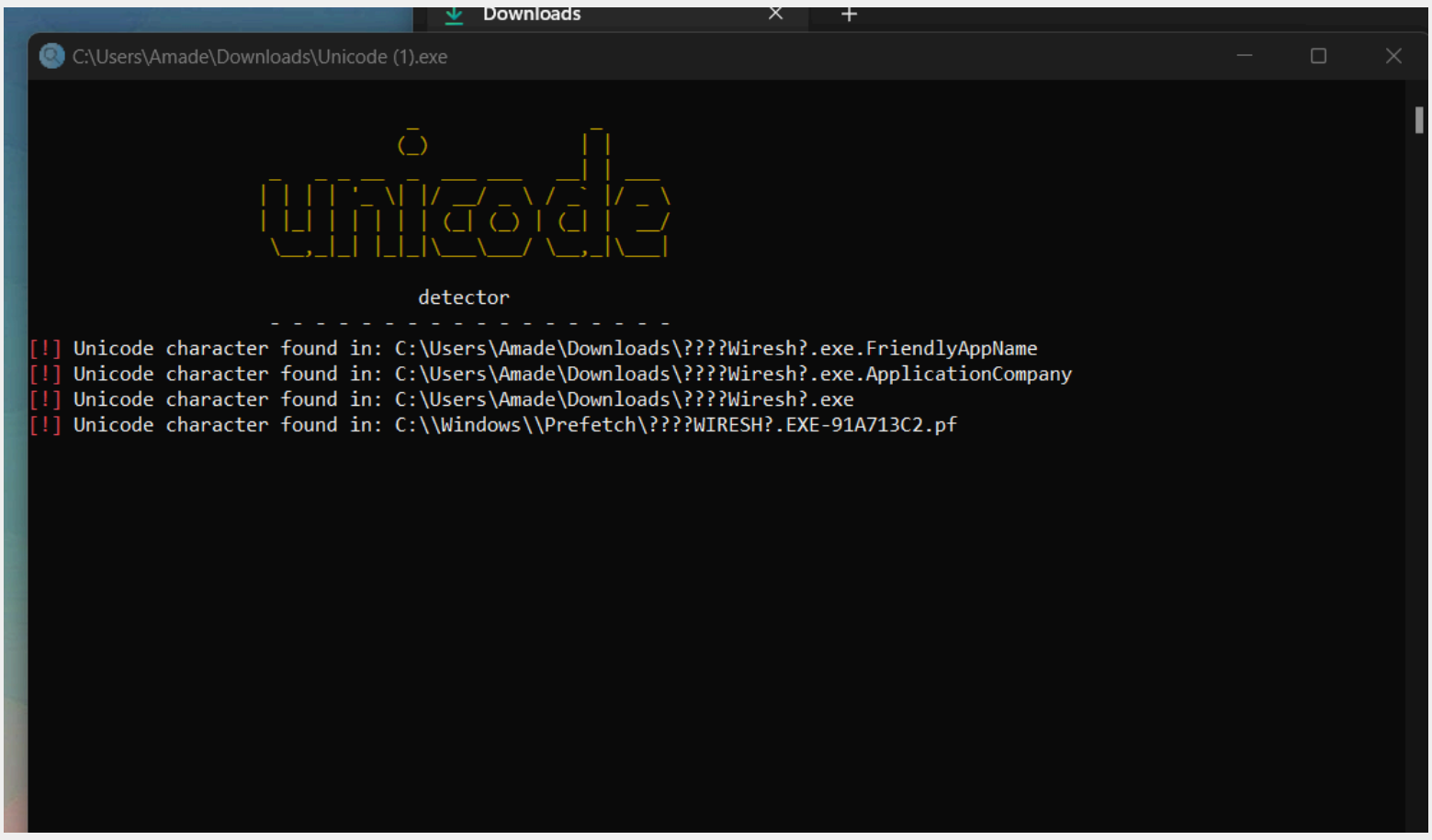
Download the executable from the following link: https://github.com/RRancio/Exec/raw/main/Files/DDFO.exe

Video demonstration;

Unicode Detector

There are several ways for detecting Unicode character files on the user's device, Rancio has made a tool that detects the most suspicious unicode files (executables, DLLs and so on)

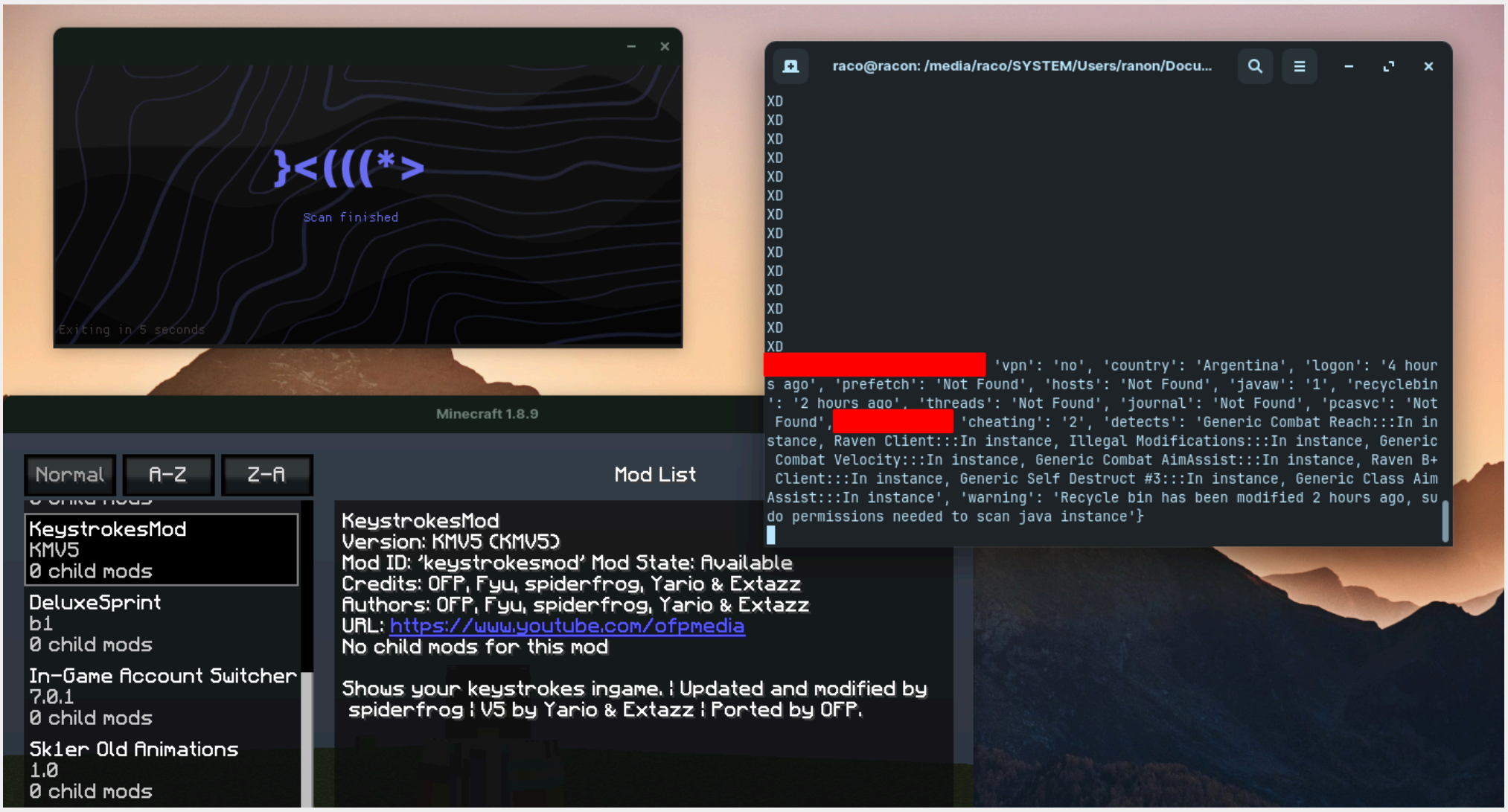
Download the tool from the following link: https://github.com/RRancio/Exec/raw/main/Files/Unicode.exe



Ocean SS Tool

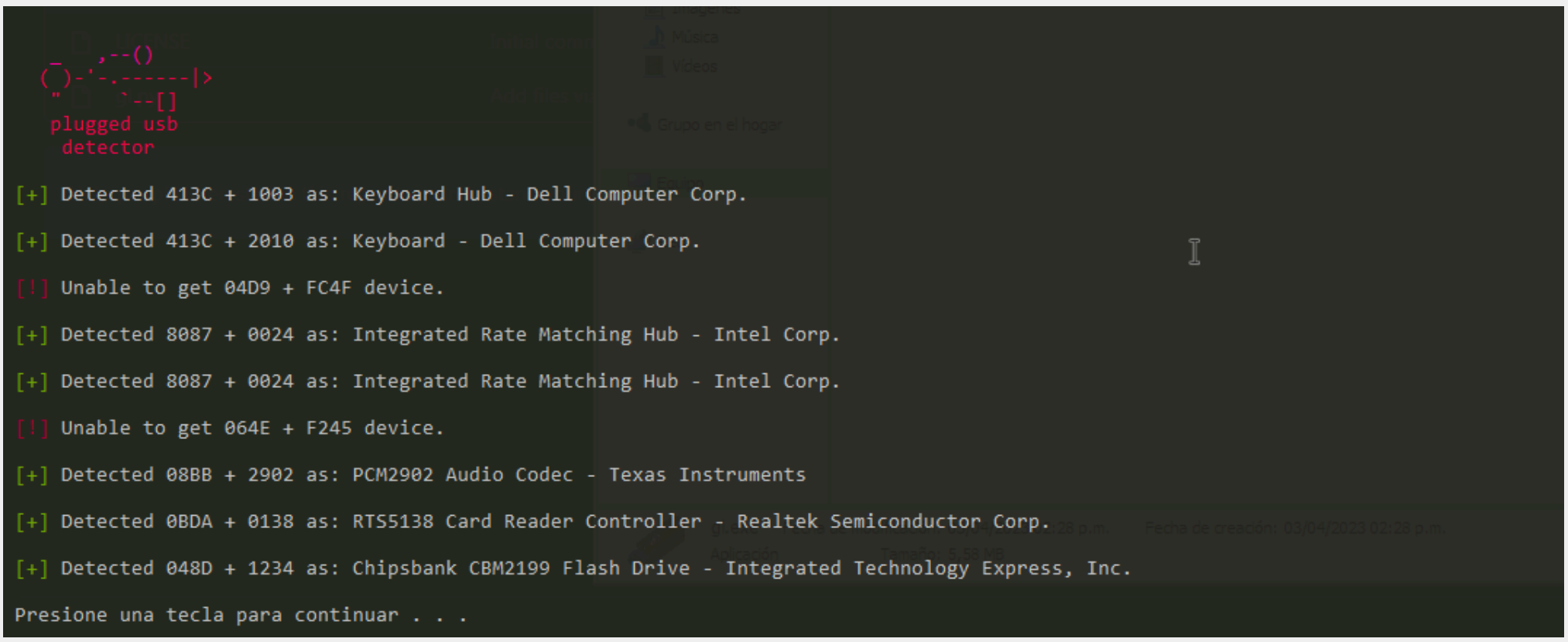
Ocean is by far the biggest Free SS Tool project ever done; the tool is a free version of Golden SS Tool. To download the tool and use it with pins, visit Ocean's Discord server.

The tool supports Windows 7/8/10/11 and Linux operating systems.



GlobalLister

queries DeviceID to a website to check specification and original product names.



Download from here: [Releases · RRancio/GlobalLister (github.com)]

Powershell WinPrefetchView

To make your job slightly easier, we've created a powershell script to Download and Execute WinPrefetchView (by Nirsoft) according to your specific dates. [Here's a Demonstration;]

- Simply Input the date in MM-DD-YYYY format (Month, Day, Year)

  Here's a downloadable version of the script:

https://cdn.discordapp.com/attachments/1086389114577895545/1086392746090168380/WinPrefetch_Powershell.ps1

Note that you will have to execute Powershell policies to run the .ps1 file (like with MiniSS.ps1)

『Chapter II』

<>!<> Forensics Stage I <>!<>

You've reached the Forensics part of this guide.

Magnet EDD

Magnet EDD stands for Magnet Encrypted Disk Detector. It detects active encrypted disks like VeraCrypt, TrueCrypt, BitLocker and so on. Download the tool Here

```
Drive C: (PhysicalDrive0), Drive Type: Fixed, Filesystem: NTFS, Size: 511 GB, Free Space: 235 GB
Drive E: [Label: AMADEUS] (PhysicalDrive1), Drive Type: Fixed, Filesystem: FAT32, Size: 10 GB, Free Space: 10 GB

* Completed checking logical volumes on system. *

* Running Secondary Bitlocker Check... *
* Completed Secondary Bitlocker Check... *

* Checking for running processes... *

* Completed checking running processes. *

*** No TrueCrypt, PGP, Bitlocker, VeraCrypt, SafeBoot, BestCrypt, Checkpoint, Sophos, or Symantec encrypted volumes dete
ctable by EDD were found. ***
```

You'll only need to execute the detector and it'll start scanning. As you can see, it scans NTFS, exFat and FAT32 file systems.

INDXRipper

INDXRipper is a Forensic utility created in Python, by Harel Segev, it is an extremely heavy and powerful utility for carving metadata from NTFS $i30 indexes.

- NTFS consists of $FILE_ALLOCATION and $INDEX_ALLOCATION attributes. By parsing both, INDXRipper will

output information of file's Modification and Access date. Unlike $MFT parsing, this tool will display slack data of deleted files as well.

To use the tool, Download it from here and extract it. Open CMD and navigate to the extracted folder, run the following command:

INDXRipper.exe //./C: --deleted-dirs output.csv

The output will be in .csv format which you may view through TimeLine Explorer. Sort by last access date to view in the latest UTC time format.

Here's a Demonstration Clip. Note that parsing will take more than 3 minutes.

$MFT

MFT, in short for Master File Table, is a database in NTFS file systems that keep track of file activity, including their streams (where they were downloaded from).

- We'll parse $MFT of the specified NTFS system (C:\ disk D:\ disk or whichever) by using Zimmerman's MFTECmd.exe. Use CMD to navigate to the same Directory and execute command:

  MFTECmd.exe -f C:\$MFT –csv .

```
C:\Users\Amade\Downloads\KAPE [-Amadeus-]\KAPE\Modules\bin>MFTECmd.exe -f C:\$MFT --csv .
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f C:\$MFT --csv .

File type: Mft

C:\$MFT is in use. Rerouting...

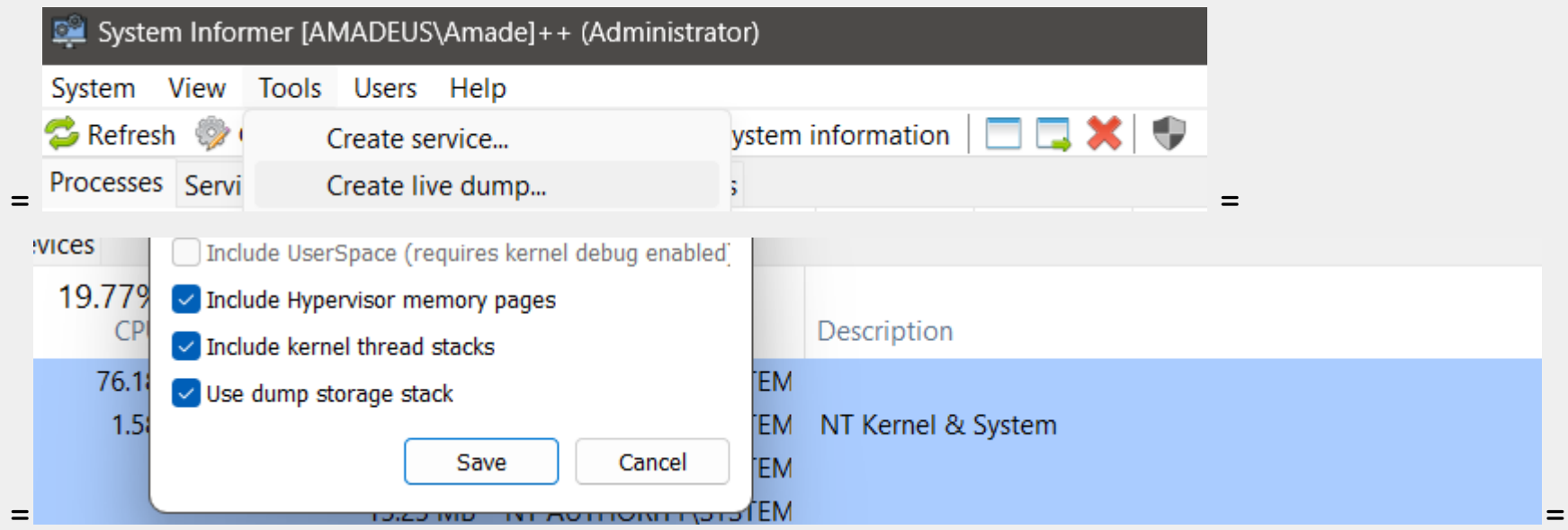
Processed C:\$MFT in 40.4817 seconds
```

- follow the video clip here. I suggest you practice these tools mentioned above on your device.

Kernel Live Dump

To perform this check, we'll need to use System Informer and create a Kerlen Live Dump .dmp file first. !WARNING! - this may bluescreen Player's PC so use it only if you suspect WMIC, Regsvr, Rundll or any hollow bypasses with it **in the very end of the SS.**

1) **Go to System Informer and Dump Live Kernel Memory**



2) **Use Strings64.exe by SysInternals to parse the .dmp output. Here's option variants:**
   - **To dump Rundll / Regsvr Injections, use a command:**

     strings64.exe -accepteula -s -n 4 .*.dmp | findstr "RunDLL32.exe " > Output.txt

     It will create a .txt output of results. Navigate through it with ctr+f

```
-55.dmp: \Users\Amade\Downloads\NullDLL64.dll
-55.dmp: \Device\HarddiskVolume3\Users\Amade\Downloads\NullDLL64.dll
-55.dmp: rundll32.exe "NullDLL64.dll"
```

- **There's a java library Injection technique that is based on CMD, to detect that using this kernel live dump - search for "java -jar " instead of "RunDLL32.exe " (note that there's a space after -jar)**

```
5.dmp: java  -jar JLoadLibrary.jar 26236 NullDLL64.dll
5.dmp: java  -jar JLoadLibrary.jar 26236 NullDLL64.dll
5.dmp: java  -jar JLoadLibrary.jar 11384 NullDLL64.dll
```

  - **To detect process hollowing, look for keywords like "Injecting" or "Encrypting File" or "Target File "**

```
dmp: process_hollow.exe .\Icetea.exe AnyDesk.exe
dmp: [+] Injecting . . .
```

```
C:\Program Files\WindowsApps\Microsoft.ScreenSketch_11.2302.4.0_neutral_split.scale-150_8wekyb3d8bbwe
C:\Users\Amade\OneDrive\Desktop\Velociraptor\a>     - Creating Target File "icetea.exe.khe"...Success
\Program Files\paint.net\System.Threading.Thread.dll
```

<>!<> Forensics Stage II <>!<>



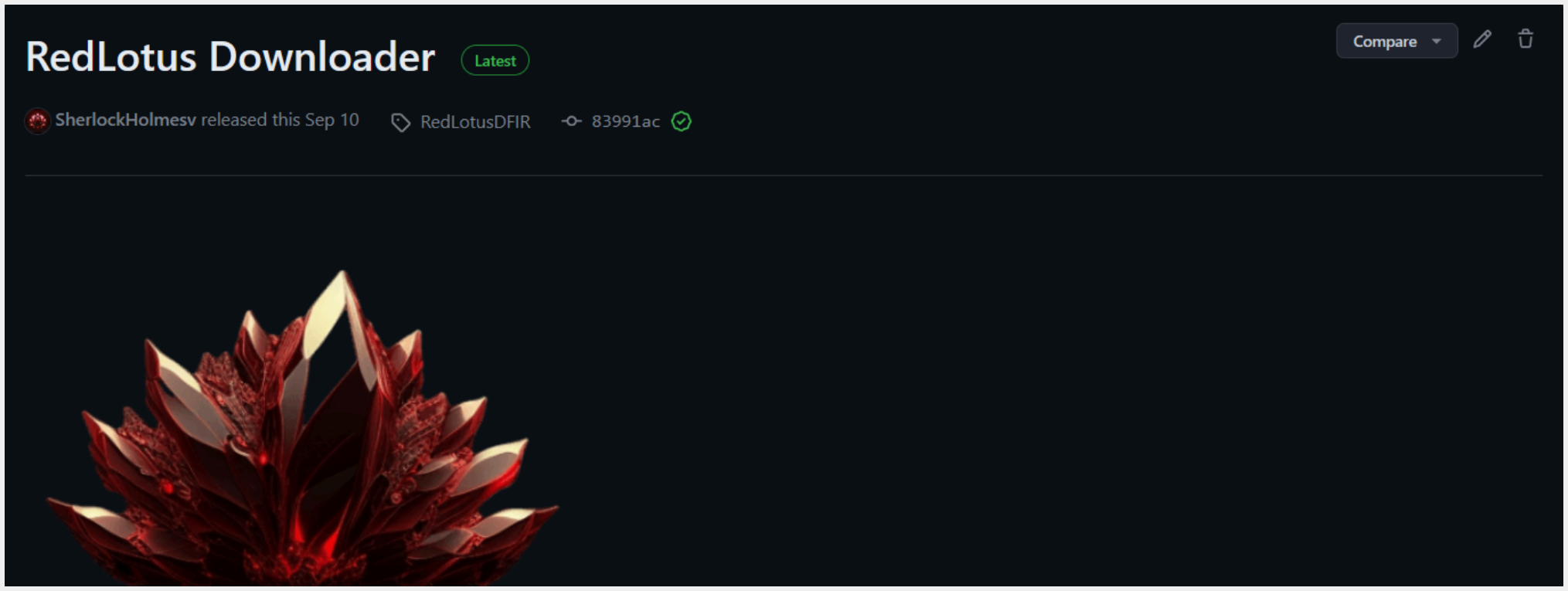
Red Lotus

Red Lotus FREE Downloader is a tool that adapts a miniature interpretation of KAPE Forensics. I've gotten permission from Eric Zimmerman to distribute a stripped version of Kape under my delivered GKape, which I have implemented in the above mentioned tool - Red Lotus FREE Downloader. (Premium Buyers get a more advanced utility)

To get a basic idea of what Red Lotus tool is, here's a demonstration video of it.

The tool will keep getting updated with newer utilities. For now, it includes everything that Red Lotus offers for Minecraft ScreenSharing.

- **Download # Red Lotus DFIR**

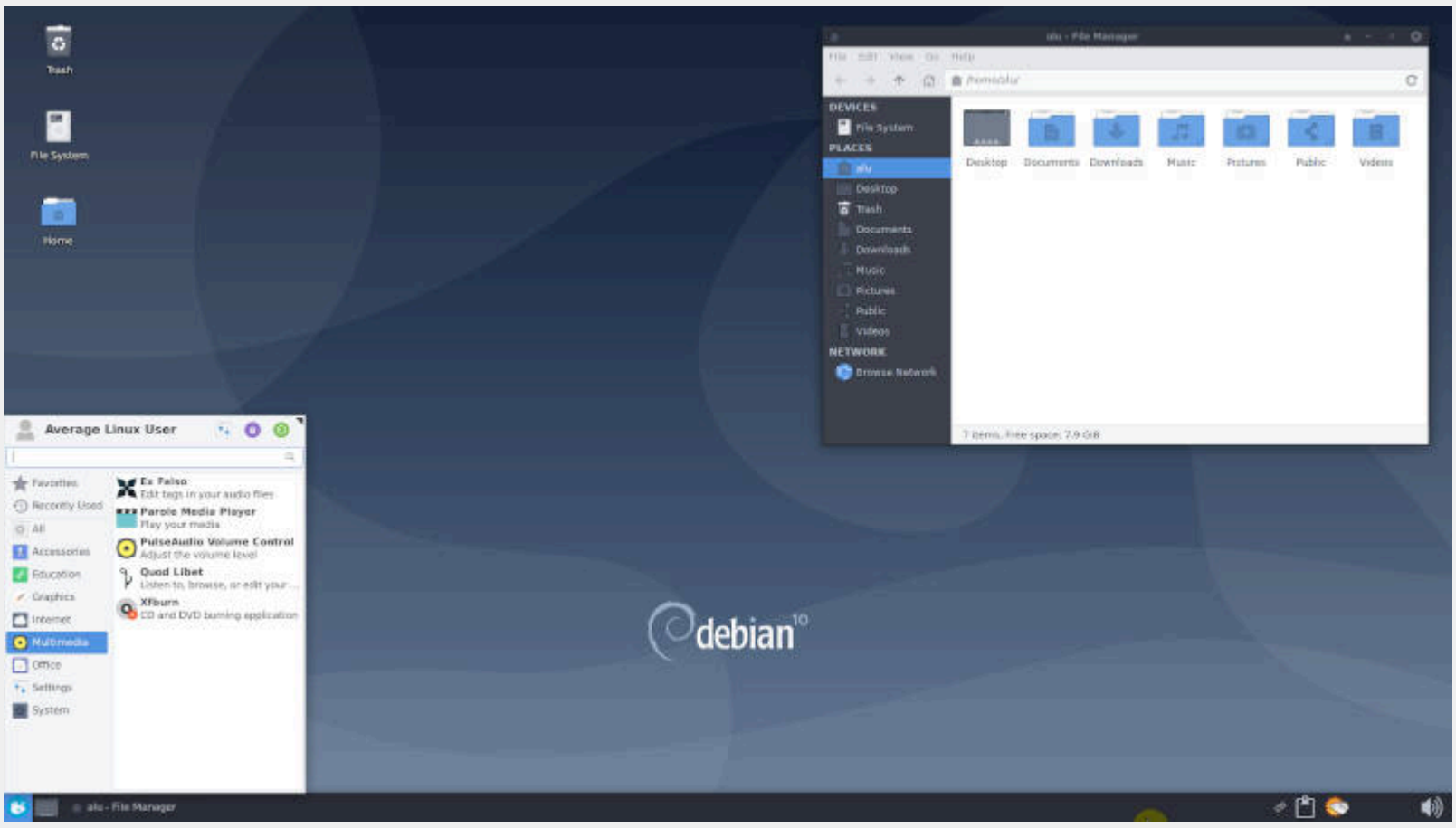


- **Join the [Red Lotus Official Discord] for more Forensic methods and collaboration with the ScreenShare community mainstream - for even greater experience, as these are methods we in Red Lotus were simply willing to publicly share. A lot of private methods still exist...**

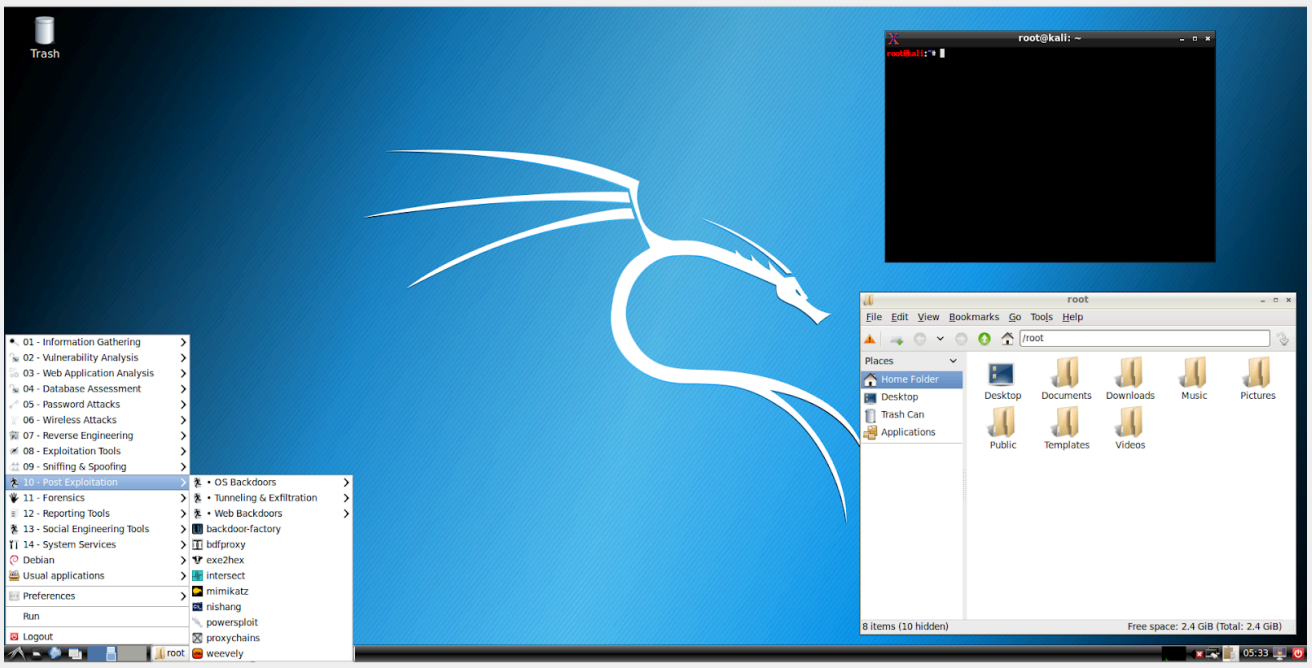
Linux ScreenSharing

Linux ScreenSharing is by far the least investigated due to the numerous operating systems. (Debian, Ubuntu, Kali, Mint, XFCE, Fedora, etc)... Here are some example desktops from chrome:

XCFE Debian



Kali Linux



The guide concentrates on the most commonly used systems, as well as the most working methods.

Terminal

CMD on Linux is called the “Terminal”, opens by pressing **Ctrl+Alt+T** and is mainly what we’ll be using for ScreenSharing. Alternatively, simply press the terminal button in their taskbar/wherever you see it. It looks like CMD or something like this >

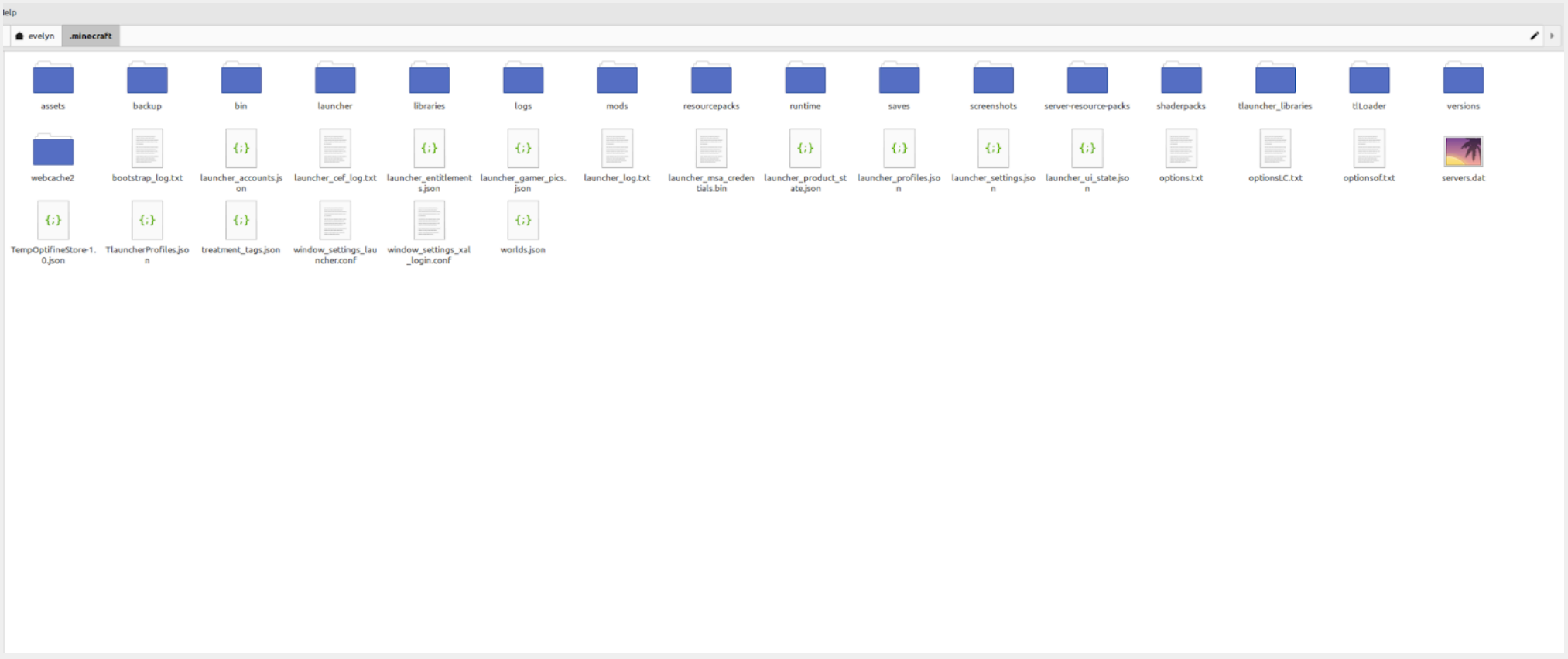




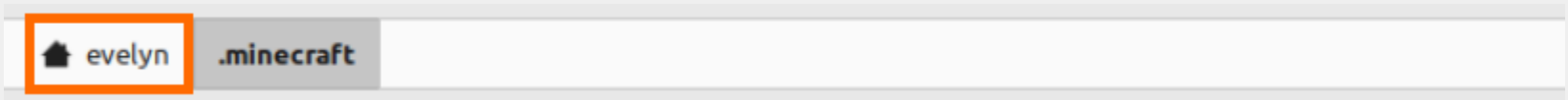
Directories

Everything on the Linux is stored in users folders, for example, if we want to go to the .minecraft folder, we’ll go to the users menu and type the directory there, in this case we type **/home/user/.minecraft** - note that instead of “user” we put

their system's username. We'll check everything in .minecraft starting from user caches to Mods, logs, etc.



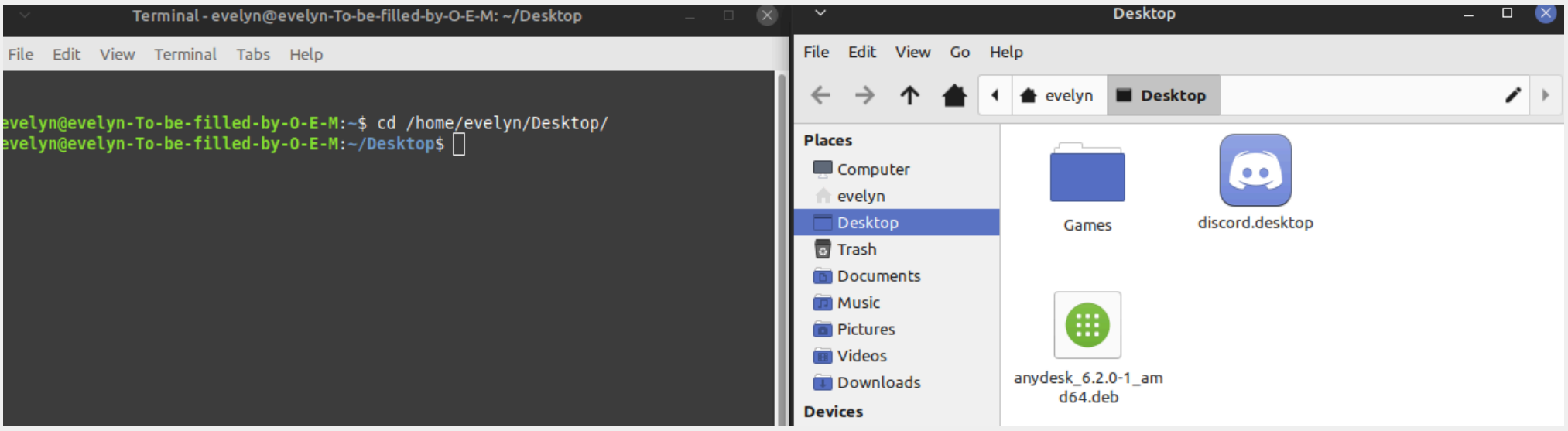
We'll know the Username by paying attention to the search bar. The search can be edited by double clicking it.



You can also simply do this by using the Terminal (way easier than this if you don't understand), we open Terminal and type the following

cd /home/user/.minecraft

It works with any directory, for this instance we'll use Desktop for it.



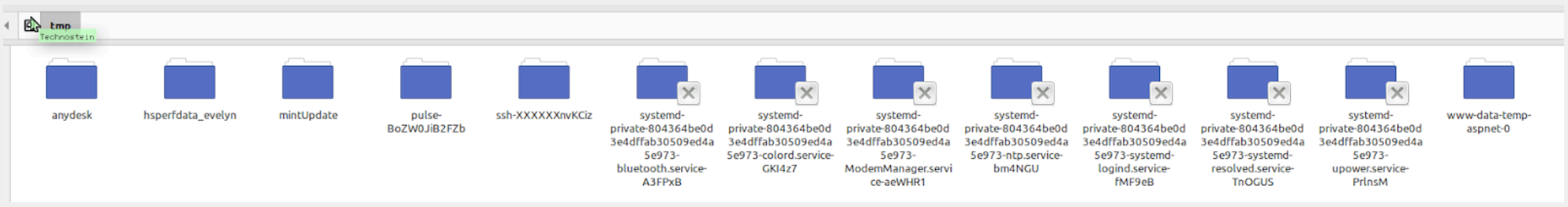
This obviously works with any directory in .minecraft, as well as for Lunar client, etc.

Directory for Lunar is

cd /home/username/.lunarclient/

For accounts/ban evasion you check the **game folder** while for chat logs/related you check the **offline folder.**

Go to **/tmp/** in the search bar to check temp files. JnativeHook can be checked here as well.



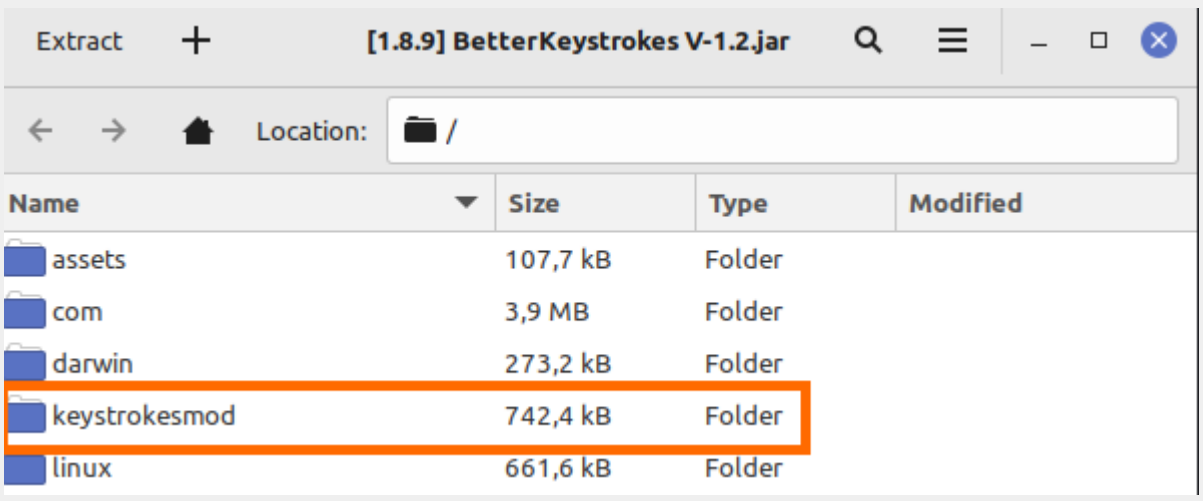
Opening Minecraft/.Jar files

While some SS tools indeed do work on Linux, downloading them might be a bit complicated in some scenarios as we use the Terminal to download files as well. However, there's a bit simpler way to do that which also works on Windows.

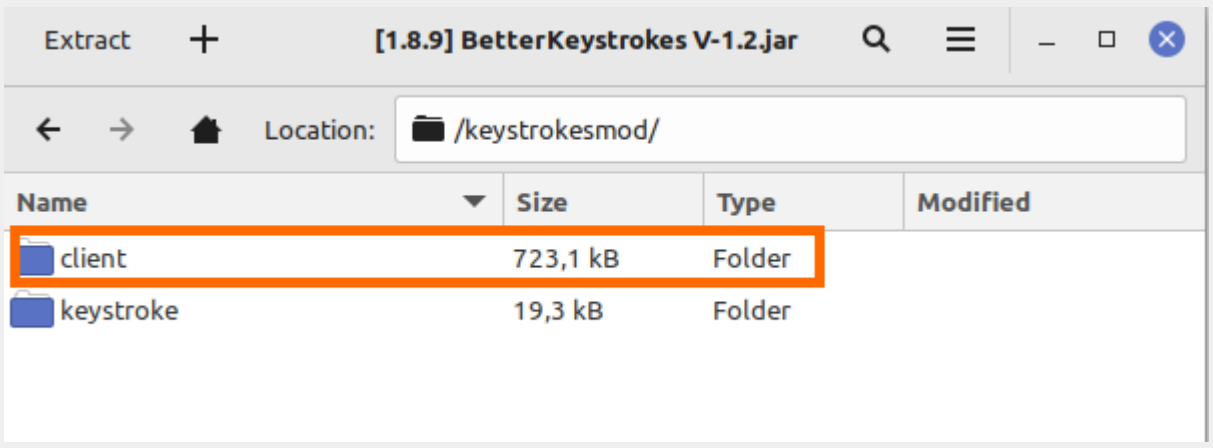
We'll begin with opening Minecraft, for Lunar and Tlauncher, you'll have to right click them and open them with Java.



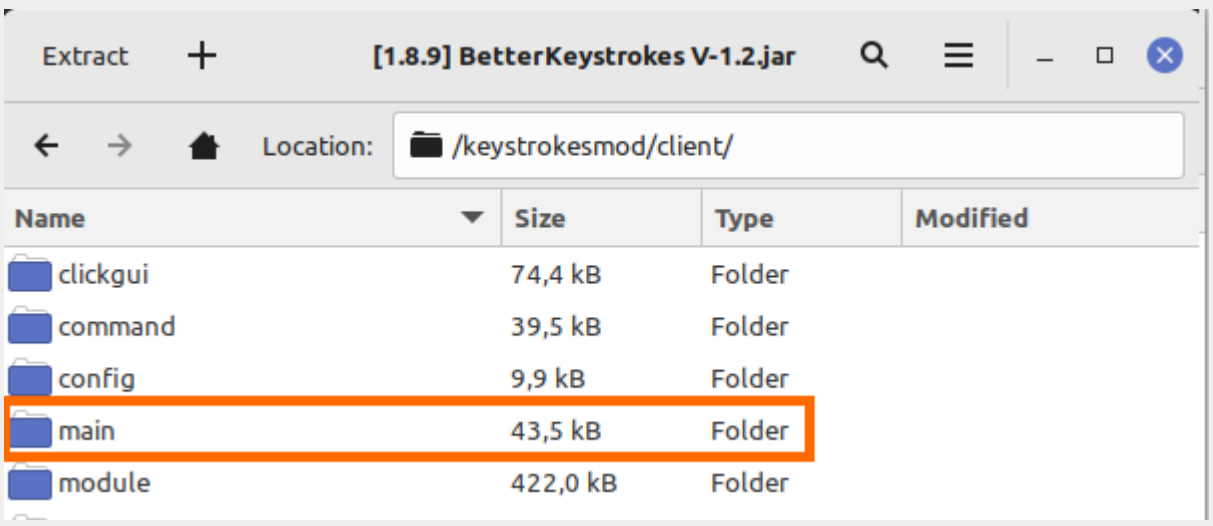
For opening .jar mods, you'll hover over "Open With" and press anything regarding open with Archive/Rar/Zip. Analyze everything shown there, imagine they're simply classes of the mod.

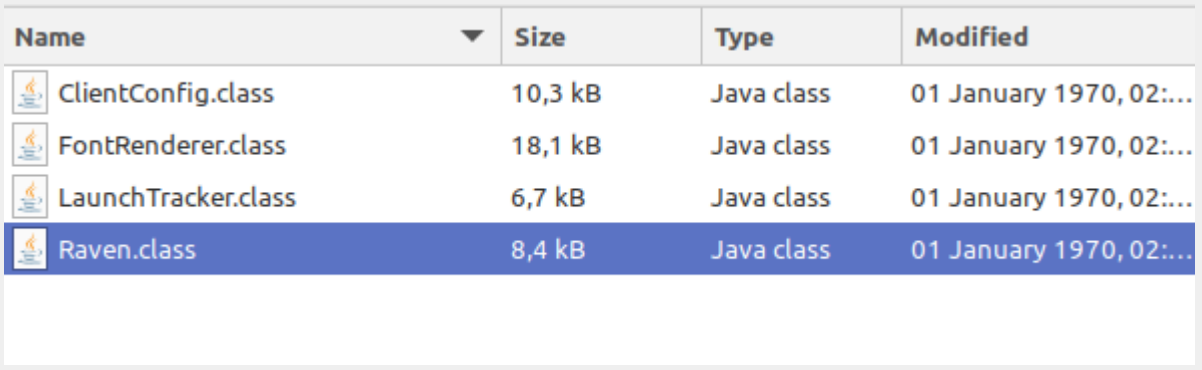


You're mainly looking for the mod name and a folder called **client** in these cases.



Usually this will leave you with a bunch of folders that represent mod classes, for example here you're looking for the **main** folder as we're investigating Raven.

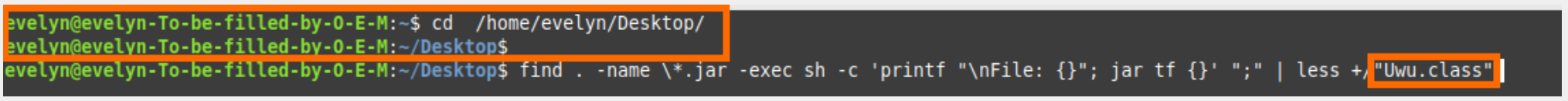




This process is a bit more generalized if you encounter different .jar files and you're unsure of what to look for, use Terminal. Cd the directory of the mods folder in .minecraft (if they're on forge) and type the following command in:

find . -name *.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"name.class"

Note that instead of the "name" in .class we'll filter already known to us classes. Check the Intermediate Windows guide to know what to look for in multiple jar mods. We'll use Desktop as an example for now.



You can also go to the mods directory manually and right click > Open in Terminal to open the terminal in that specific directory.
You may filter classes such as **AutoClicker, AimAssist, Reach, Destruct** etc as these are commonly in the jar cheat mods. If anything pops up in the results that indicates this class is from that specific Jar file (for example it shows the directory of the mods folder with the mod name and UwU.class etc) you can ban.

Modified recycle bin

To check the last modified date of the recycle bin, easiest way is by using the Terminal and typing the following commands:
To get to the recycle bin directory, we type the following first:
cd ~/.local/share/
Then we'll check the last modified time of the bin by typing:
ls -lt ./Trash/

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ cd ~/.local/share/
evelyn@evelyn-To-be-filled-by-O-E-M:~/.local/share$ ls -lt ./Trash/
total 12
drwx------ 2 evelyn evelyn 4096 Aug 29 08:56 expunged
drwx------ 2 evelyn evelyn 4096 Aug 29 08:56 files
drwx------ 2 evelyn evelyn 4096 Aug 29 08:56 info
```

Installed applications

Checking installed might help us catch users with a VPN or anything related, possibly even clickers as well. To access the list of currently running programs, we type the following command in the Terminal:
dpkg -l

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                                  Version                                                  Architecture Description
+++-=====================================================-========================================================-============-=============================================================
ii  accountsservice                                       22.07.5-2ubuntu1.3                                       amd64        query and manipulate user account information
ii  acl                                                   2.3.1-1                                                  amd64        access control list - utilities
ii  acpi-support                                          0.144                                                    amd64        scripts for handling many ACPI events
ii  acpid                                                 1:2.0.33-1ubuntu1                                        amd64        Advanced Configuration and Power Interface event daemon
ii  add-apt-key                                           1.0-0.5                                                  all          Command line tool to add GPG keys to the APT keyring
ii  adduser                                               3.118ubuntu5                                             all          add and remove users and groups
ii  adwaita-icon-theme                                    41.0-1ubuntu1                                            all          default icon theme of GNOME (small subset)
ii  adwaita-icon-theme-full                               41.0-1ubuntu1                                            all          default icon theme of GNOME
ii  alsa-base                                             1.0.25+dfsg-0ubuntu7                                     all          ALSA driver configuration files
ii  alsa-topology-conf                                    1.2.5.1-2                                                all          ALSA topology configuration files
ii  alsa-ucm-conf                                         1.2.6.3-1ubuntu1                                         all          ALSA Use Case Manager configuration files
ii  alsa-utils                                            1.2.6-1ubuntu1                                           amd64        Utilities for configuring and using ALSA
ii  amd64-microcode                                       3.20191218.1ubuntu2                                      amd64        Processor microcode firmware for AMD CPUs
ii  anacron                                               2.3-31ubuntu2                                            amd64        cron-like program that doesn't go by time
ii  anydesk                                               6.2.0                                                    amd64        The fastest remote desktop software on the market.
ii  apg                                                   2.2.3.dfsg.1-5build2                                     amd64        Automated Password Generator - Standalone version
```

This may give us an idea of what to look for. To check when they last ran an application, type **-mtime +1** in the Terminal, compare the dates to minecraft logs.

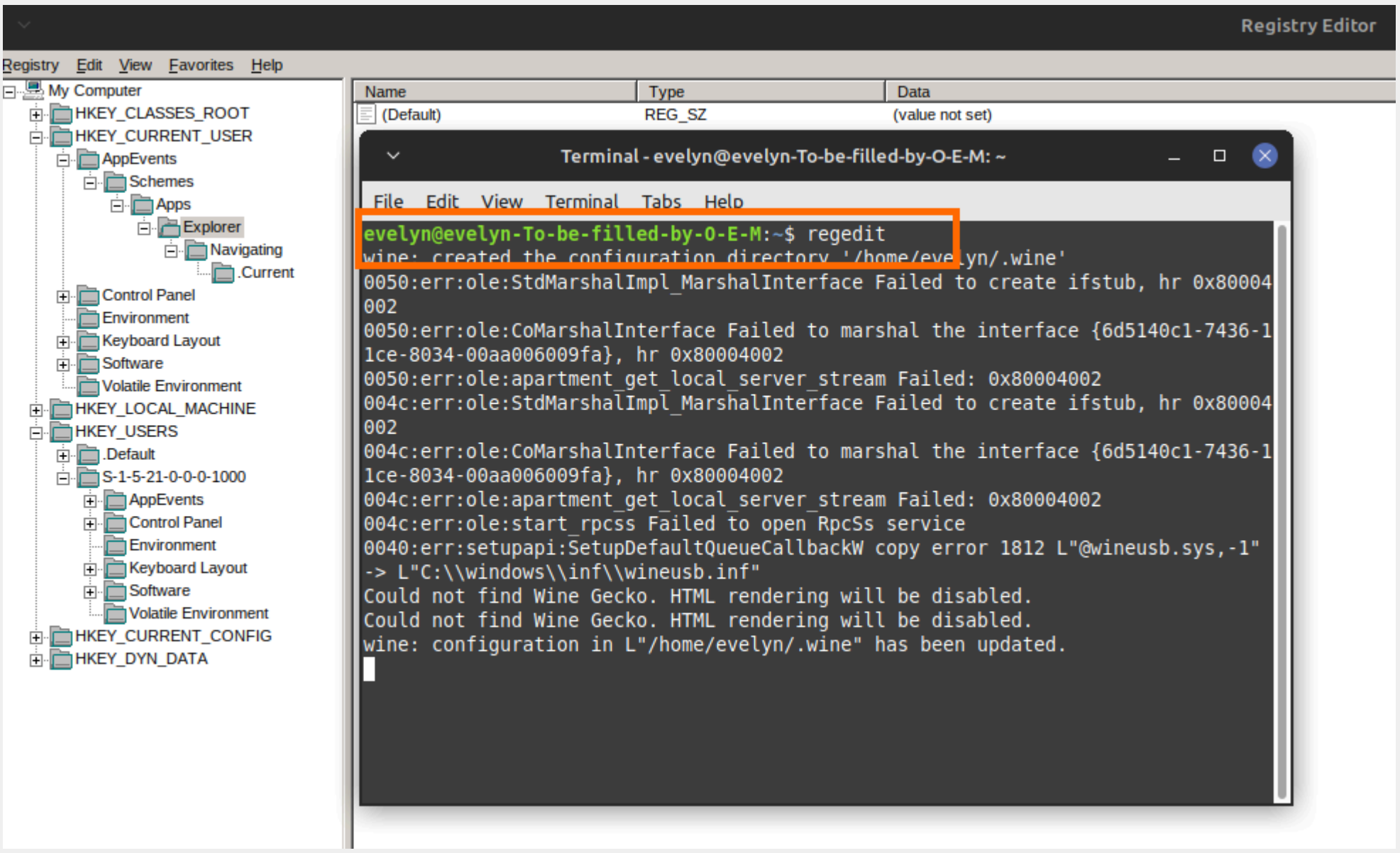
Current user

Knowing the current user's name will help you with Terminal directory commands. To know the logged user into the system, type **who -H** in the Terminal.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ who -H
NAME     LINE         TIME             COMMENT
evelyn   tty7         2022-08-30 11:44 (:0)
```

Regedit

Regedit requires wine to be accessed on Linux. However, it also installs itself when you type **regedit** in the Terminal. More about this in the future references. Note that it may take up to a minute to install so be patient.



Fast access

You can also access and open files using Terminal, this can save some time if you can't find the necessary directory. Open the Terminal and cd to the directory you want to, we'll go to the game directory in Lunar for this instance.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ cd ~/.lunarclient/settings/game
evelyn@evelyn-To-be-filled-by-O-E-M:~/.lunarclient/settings/game$ open accounts.json
```

Simply type **open accounts.json** to run the file. Note that you can also use the CMD to open applications instantly without going to their directories by just typing out the application name. (For example, XClicker if it shows up in -mtime -1)
Use this to start **minecraft-launcher** if they have that and run the latest version they were on, go to singleplayer from there and press right shift, if a cheat GUI shows up, ban for cheating.

Restarting shell before SS

On Linux, explorer is called "shell". To check if they restarted their shell before SS, we'll need to compare their shell time to their system time.
This process may be a little complicated, as we retrieve the PID (Process ID) first, and to get the shell's ProcessID, we first need to know which **environment** they're on. We do that by using an echo command:

echo $XDG_CURRENT_DESKTOP

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ echo $XDG_CURRENT_DESKTOP
XFCE
```

In this case, we're on XFCE environment. Every desktop has its own command to retrieve the PID for. In this case, we type **pidof xfdesktop.**

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ pidof xfdesktop
1442
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

This is different for every desktop. Here are some known commands -

[

Pidof gnome-shell - Linux Gnome

Pidof nemo-desktop - Linux Cinnamon

Pidof dde-desktop - Linux Deepin

Pidof caja - Linux Mate

Pidof kali-desktop - Kali Linux

]

Then, we'll need to retrieve the system's PID, we do that by typing **pidof systemd** this time instead.

Now, to get the runtime of the desktop and the system, we type the following:

ps -p **,**** -o lstart,etime,cmd** - we input the PID of the processes in place of *-s.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ ps -p 1442,1222 -o lstart,etime,cmd
                  STARTED     ELAPSED CMD
Tue Aug 30 11:44:48 2022    07:25:53 /lib/systemd/systemd --user
Tue Aug 30 11:44:49 2022    07:25:52 xfdesktop
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

If time elapsed shows it's before SS, they've restarted the shell before SS.

Modifying files

If players modify files before SS, you can either check the modified file or ban right away. However, if that's not the case, you can keep investigating.

To check what commands they've ran regarding .exe files, type the following:

history | grep .exe

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ history | grep .exe
   52  mono hello.exe
   54  mono hello.exe
   71  history | grep .exe
  129  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' :;: | less |/".Uwu.class
  130  find %{1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
  131  find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
  133  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".Uwu.class
  134  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".reach.class
  135  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".Uwu.class
  136  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".reach.class
  137  find . -name \*.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"Uwu.class"
  140  find . -name \*.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"Uwu.class"
  141  history | grep .exe
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

This will show if they did anything with .exe files. Anything suspicious like **mv AutoClicker.exe AnyDesk.exe** means they've renamed AutoClicker to AnyDesk.

To look for generally recently modified/accessed files (which shows the list of accessed files in the past 24 hours, this is what restarting shell clears/tries to hide) type the following command:

find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe" - you may also type .jar/.class/.o etc.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
2022-08-30 17:50:06.470204334 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000147.log
2022-08-30 17:42:14.583508098 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG
2022-08-30 17:42:14.579507972 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/MANIFEST-000001
2022-08-30 17:42:14.575507849 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000009.ldb
2022-08-30 17:42:14.567507599 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000007.log
2022-08-30 17:37:14.582170689 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_trello.com_0.indexeddb.leveldb/000049.log
2022-08-30 16:49:41.257481332 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG
2022-08-30 16:49:41.253481208 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/MANIFEST-000001
2022-08-30 16:49:41.245480958 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000149.ldb
2022-08-30 16:08:39.373080037 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.blob/b/00/3c
2022-08-30 16:08:00.867886260 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_trello.com_0.indexeddb.leveldb/LOG
2022-08-30 15:19:02.875929447 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG.old
```

Unplugging USB

Unplugging USB before SS results in a ban. To check when they've unplugged the USB, run the following command:

journalctl | grep disconnect | grep usb

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ journalctl | grep disconnect | grep usb
Aug 24 07:51:16 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 5
Aug 24 08:01:50 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 6
Aug 24 08:02:16 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 7
Aug 24 08:08:51 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 4
Aug 24 08:33:51 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 08:52:44 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 5
Aug 24 09:16:02 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 09:26:54 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 6
Aug 24 10:10:24 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 3
Aug 24 10:13:00 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 10:33:31 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 5
```

『Chapter III』

Ban Evasion

As for the final method in this guide, I've made a powershell script that'll parse the whole C:\Users directory with sub-directories, checking all Minecraft folders and other .txt, .json files for the strings "user" & "username", then it'll output a .csv file that you can inspect using Timeline Explorer for your specific usernames.

Download
https://cdn.discordapp.com/attachments/1083164927696588921/1106689460202123304/WatsonAltChecker.ps1

The tool will ask you to input a Username you're looking for.

```
Searching for files
   Processing
   [ooooooooooooooooooooooooooooooooooo                                                    ]

+ ...  "Results are saved in the current directory!" -ForegroundColor Green
+                                                    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The string is missing the terminator: ".
    + CategoryInfo          : ParserError: (:) [], ParseException
    + FullyQualifiedErrorId : UnexpectedToken

PS C:\AutoMation> .\Tool.ps1
  _/          _/        _/_/    _/_/_/_/_/
 _/          _/      _/    _/       _/
_/    _/    _/      _/_/_/_/      _/
 _/  _/  _/         _/    _/      _/

Watson Alt Tracker - By Sherlock Holmes#8388
Results saved in current directory!
PS C:\AutoMation> .\Tool.ps1
  _/          _/        _/_/    _/_/_/_/_/
 _/          _/      _/    _/       _/
_/    _/    _/      _/_/_/_/      _/
 _/  _/  _/         _/    _/      _/

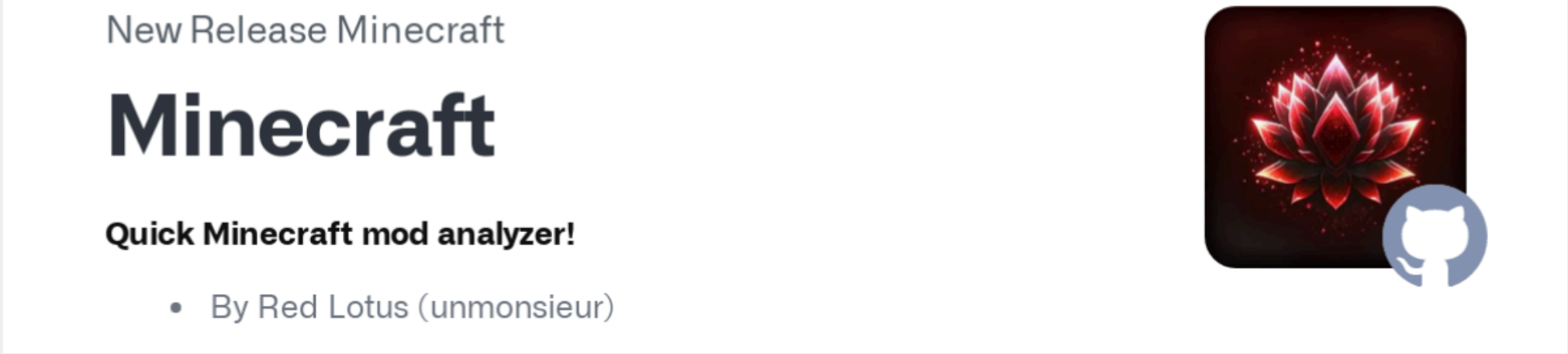
Watson Alt Tracker - By Sherlock Holmes#8388
Enter string to search for: Monsieur
```

There's also a 2nd script that I've created, it will output and wrap all chat logs of the user from known directories. Here's a demonstration of how to use it:

This time, I've also made the script search for deleted .json / .log / .txt files. Here's the downloadable script that you can copy-paste in powershell:

https://cdn.discordapp.com/attachments/1095072783068827769/1106683442105688084/message.txt

Mod Analyzer



Initially meant for 1.9+ tier list communities, apparently, this free Mod Analysis tool is pretty useful for almost every other Mod Client.

- **The Tool calculates hashes and streams (download links) of Mods located in the default Minecraft Mods directories.**

  https://github.com/SherlockHolmesv/RedLotus_ModAnalyzerFREE/releases/tag/Minecraft

  DEMONSTRATION

- **Additionally, you can use the message.txt (powershell script) from the [[Ban Evasion]] section, modify the path to Mods folder and use strings like "autoclicker" or "combat" to scan every single Mod for suspicious classes! Here is a list you could use:**

  AutoClicker|Raven|UwU.class|Reach|FastBridge|KillAura|AntiKnockback|AntiKB|SelfDestruct|Nightmare|Huzuni|gucciclient|Triggerbot|tr1ggerbot|DoxThreat|Hamdulilha|trumpclientftw_bape_ggb6|mergeclienta|sumoclient|shitkid|macros|iReSqtkUVgaWmiReSqtkUVg|otaku

DPS-Sanner

Tool developed by Nay [11599773536619193 63], DPS Duplicate Scanner, proves to be extremely useful in long-run ScreenShares where competitive tournament payout are on the line!

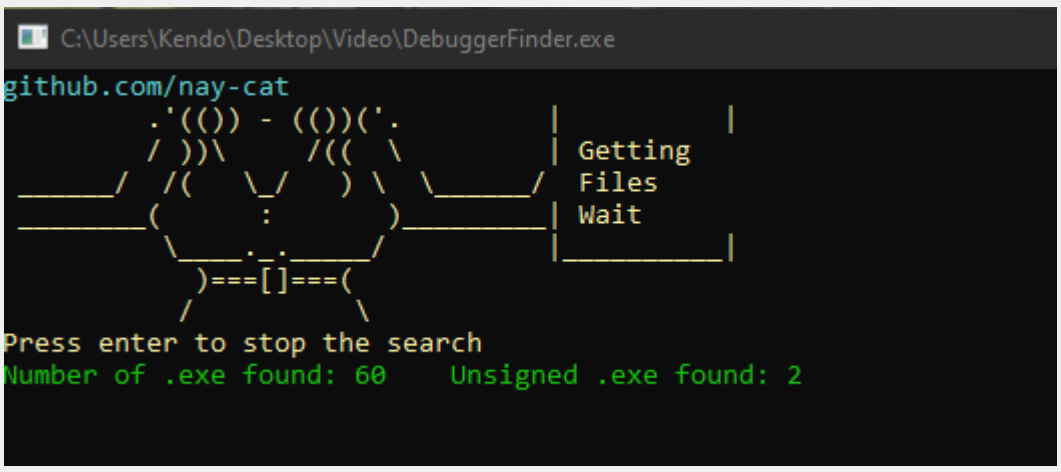
https://github.com/nay-cat/dpsanalyzer/releases/tag/1.3 - DOWNLOAD

DEMONSTRATION

- **The tool analyzes two same name-different compilation date entries from the DPS Service! something that might be added to Echo SS Tool very very soon...**

Anti-Debugger

Pretty much any popular hacked client / cheat has a built-in anti-debugger configured for file's source security, this program was against developed by Nay, detecting all the files that include this functionality on user's device.

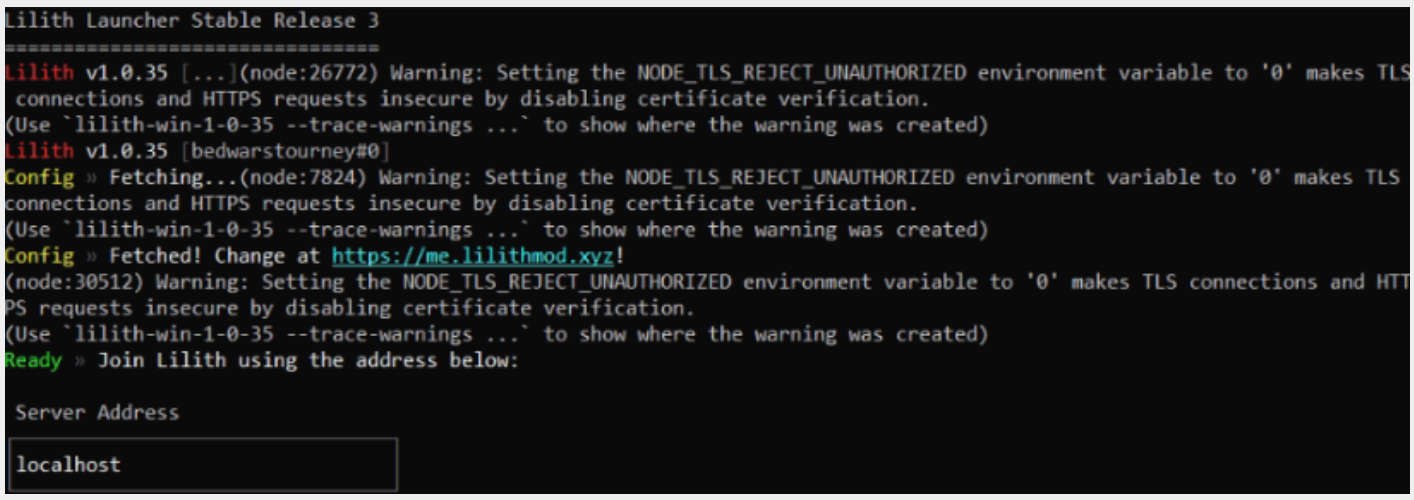


https://github.com/nay-cat/fuckbuggers/releases/download/v1.5/DebuggerFinder.exe

How will you detect proxy cheats

Proxy clients often are meant to work on specific servers (for example, Hypixel), through a host that in some way connects to the target server with a different IP. Most of the proxy clients have died out, latest largest proxy that I am aware of is **Lilith**

Let's review how Lilith works



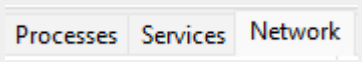
Proxy clients are *in theory* meant to improve the user's PING for smoother connection. Although there are private clients that are meant for cheating. Here's a prime example from one of Headed's old **> Videos <**

Detecting Proxy clients requires the players to remain on the server while they're being ScreenShared. Here are some ways and theories on how unreleased / private proxy clients can potentially be detected. *Let's start with unethical ways. . .*

Requiem has released a video detecting Proxy Clients by IP & Geolocation monitoring. While RedLotus does not approve of this specific detection, your server(s) might, considering you're a trusted ScreenSharer and not an idiot.

https://www.youtube.com/watch?v=nHLFNOtrMTk

An alternative to this would be using System Informer or TCPView by Sysinternals for a more ethical and secure approach, hence we will be analyzing the **remote address** javaw.exe is receiving packets from.

In System Informer, go to **Network** 

TCPView will automatically display Network activity.

System Informer



To accurately determine if the user is on a proxy client or not, you need to know the Remote Address of your server in the first place, so use this to gather it from your device first.

Analyzing chat logs for server IP connections and the IP of what you've found is the next step after using TCPView and SystemInformer. While Wireshark and Geolocation analysis is definite, this is ethical and does not put players in danger of malicious staff members.

Theorising

One theory for catching private proxy clients is that they need a definite host to connect to, which means the user will most definitely be running a script or a terminal / command prompt in the background to transfer the connection towards Minecraft. Looking for suspicious scripts, .exes or terminals may work.

Chat logs and servers list can help with detecting suspicious connections.

Event Logs bypass

First of all, unless the Eventlog service is disabled - these logs can not be bypassed or overwritten.

However, event log tracking can be disabled without the need to disable the service itself entirely. One of the easiest ways is to utilize PsExec to “hijack” a specific event log from the event viewer by disconnecting it. **> DEMO <**

Obviously enough, detecting this is also not an issue considering we will still be able to see the latest entry. If they clear event logs, that can also be traced by event ID 1102. Finding PsExec itself on the current system / current boot is enough to ban for.

Java bytecode execution traces

Detection to this is not mentioned but still possible with the Kernel live dump method covered in the free Red Lotus guide. However, in most scenarios a bytecode execution will leave a .class file behind, using USN Journal by “File Create” > “.class” can guide you thru this. Otherwise using the methods by prefetch mentioned in the Red Lotus server will do most of the trick because it will log what file was executed, what file was dropped and so on.

When it comes to .exes, they may drop a .jar that they use for such execution.

Theorising

While analyzing this behavior, extracting $MFT and $UsnJrnl:J is a good idea. A collection of Prefetch data (WinPrefetch or PECmd) with $MFT and $J, as well as Kernel Live Dump can help with analyzing what was the following action after executing CMD, Powershell, or any Terminal / Scripting environment they have (Vscode, nodejs, python etc)

Use the dates collected from suspicious files / CMD executions from MFT to analyze the USN Journal and Kernel Dump. The rest is on you.

Cheats that don’t load anything with AnyDesk

First things first - we need to even assume what we’ve found is a cheat. Most cheats are unsigned and most cheats require Administrator, thus they have a shield icon on. Most legitimate windows files - don’t. If nothing happens when you run them, try running them from CMD (cd to the directory (cd path) and run the .exe)

Other ways of detecting this is by simply asking the Player to share their screen through Discord whilst having AnyDesk closed or ask them to send you the executable (keep in mind that you can’t ban if they don’t unless you have a strong reason to believe it is a cheat) a strong reason would be:

- If you upload the file on virustotal and it returns as a virus, its behavior is strange and abnormal;
- If the file has no signature + its zone.identifier is from a fishy web (or cdn.discordapp) (in that case you can download the file itself on your device as well)
- If the file has a shield icon and the user has in fact executed it himself in current system boot.
- You analyze the file with DiE (detect it easy) and find relations to AnyDesk

Renamed extension with no traces

System Informer can’t be bypassed unless the file’s extension includes a Chinese character since it can’t read the Unicode of it. However, it leaves a trace with question marks. Simply including a question mark and a dot in the filter of whatever service you need (so ? > . > :\) or using the DPS method for catching renamed extensions (refer to SS Community > Regex) will do.

If the file is an executable, it will leave a trace in prefetch, depending on its execution way. Such files are often executed using the WMIC bypass method.

In the end, using a search everything regex for unicode characters (refer to SS Community > Regex) will do the trick.

For files with no extension, keep in mind that these files can NOT be run even through CMD, except for the WMIC, so filtering search everything for extension-less files (ext:) (space after ext:) and sorting by size (or limiting the size) may work. So:

size:3mb..15mb ext:

And frankly, in theory these types of files should leave a trace in jumplists and shell:recent, so... (or USN Journal > renamed from & renamed to > ?) question marks will do 🙂

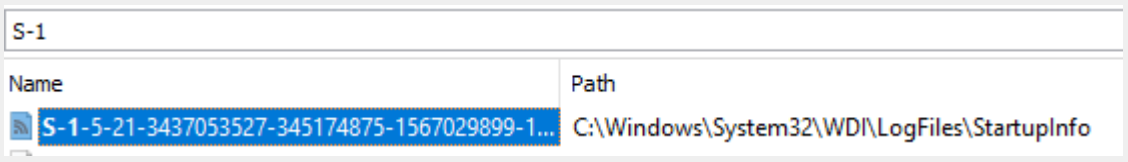
Kernel live dump

For ScreenSharing purposes, Kernel Live dump is a mini-dump of the memory in current system boot, current instance. It can show file executions, CMD commands, powershell commands, file downloads and so on. Its main use would be to detect process hollowing, regsvr / rundll injections and so on. (All covered in the free guide)

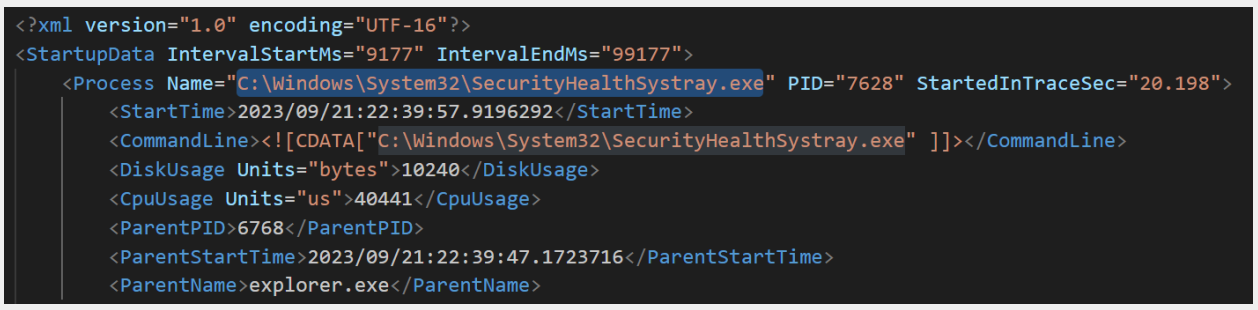
Startup XML

Instead of using task scheduler, some users utilize the startup functionality of Windows to run cheats on system boot. Startup data is logged in **C:\Windows\System32\WDI\LogFiles\StartupInfo**

Every startup .xml starts with a **S-**



The directory is inaccessible by default, but just double clicking on the .xml file opens what we need.



『Chapter IV』

History of Bypasses📑

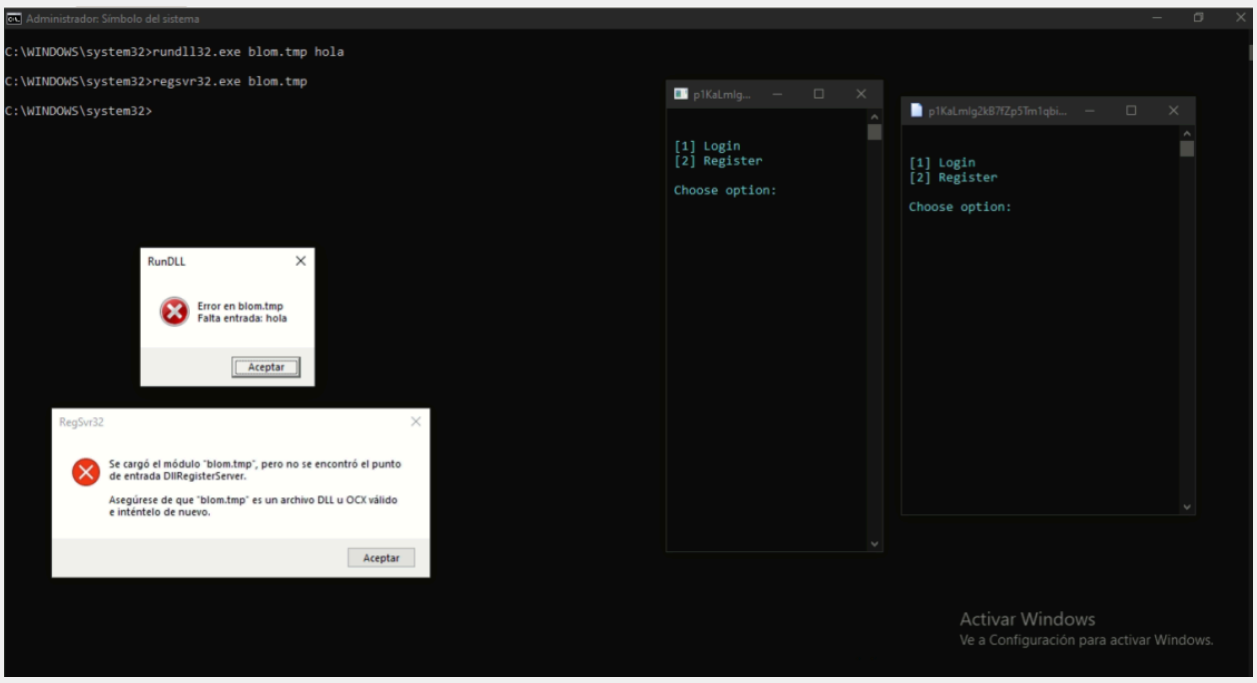
This Tome II release will cover the history of the development of Bypass methods, where they took inspiration from, what were the strongest Bypasses from 2019 to present day (2024).
It is no secret that besides the ScreenShare Community, becoming more popular on June 19th 2021, there were anti-screenshare communities (mostly private) that began developing methods for bypassing; communities like Pragma, Cheating Community, Kraken, Legion, Stringless, EzBypass, EndofScreenShare, Stella, Katana AntiSS were (some still are) dedicated to bypassing ScreenShares - Katana AntiSS being the most known and prioritized. Besides them, many Cheat communities and Discord were created such as, for example, Exelon, Odisea, Akira, Isolation, *obviously for Vape & Skript furthermore* and so on*;*
The ScreenShare Community has had to deal with the rise of Bypass servers and the most skilled Bypassers at the time - Notsnakesilent, Xavi, UUID and others - HogRider, Sookiuk, Moostt_, and many others during ScreenShare Events held in SSC itself.
As of today, the majority of these bypass communities have died out and now ex-bypassers have either quit the community, switched to SSing or maintain a chill “this game sucks, let me cheat!” type of vibes!
Xavi is currently a Delegation member at Red Lotus (ex Headquarters), potentially still the “scariest” Bypasser alongside with Notsnakesilent;
Notsnakesilent now helps the CEO (Owner) of SS Community - Rancio with a SS Tool - Ocean;
In 2024 the ScreenShare Community officially discontinued, after guiding and helping the ScreenSharers, practically ever since 2020, officially since 2021. Being the first major and strong community for ScreenSharers and server staff during the times of no DFIR methodologies being even discovered at all by the community, SS Community was the biggest project that inspired Red Lotus itself. *Side note to remark how old the SS Community actually is - ChatGPT wasn’t even a thing when we started releasing DFIR scripts;*

First wave

The first wave of Bypasses were pathetic in the English speaking communities, as it mainly consisted of shift+deleting files, their prefetch traces, clearing registry entries, replacing files, injecting DLLs through ProcessHacker2 (and other Injectors), editing the date modification of cheats (using external softwares), clearing registry, using .jar cheats with obfuscated classes and so on. As this isn't a primal handbook, it would be uncharacteristic for Red Lotus to be discussing each and every basic bypass individually, thus we'll concentrate on more complex bypasses from 2019~2020 that might even work today.

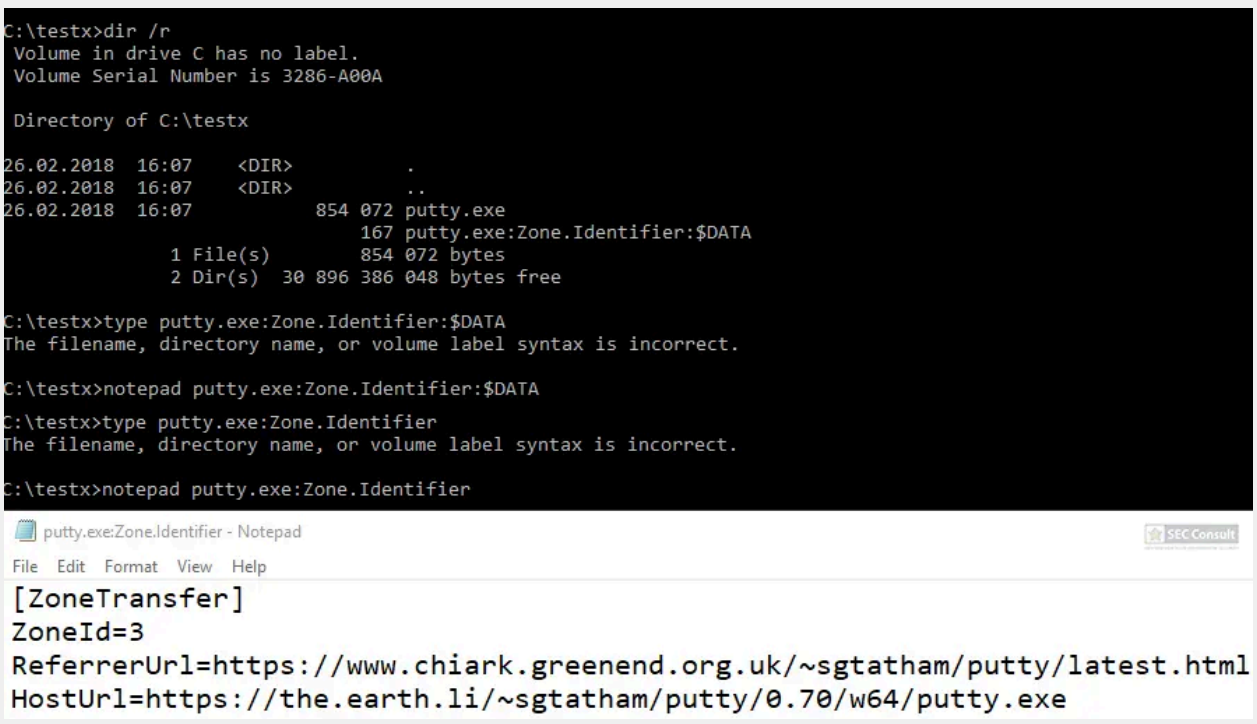
Read-only rundll & regsvr

Attribute "read only" on a .pf trace prevents it from updating and re-logging new actions within the trace. In other words, if data gets *written into* a file / trace / directory every update sequence, setting it to *read-only* blocks that operation.
RunDLL32 and Regsvr32 are used to register / run DLL files, it becomes increasingly challenging to prove an injection of DLLs run this way.



Alternate Data Streams

Although popular in 2019, the Alternate Data Stream method is probably the most undetected bypass even today.



The whole purpose of this method is to hide the file itself on NTFS, if utilized with directory junctions (view the article above), as well as using a Unicode character invisible to System Informer, a smart bypass can be created, making cheats practically impossible to detect if an SSer is unaware of such possibilities or doesn't understand it at all.

Unicode Characters

Universal Character Encoding is often still used today to bypass System Informer checks or to confuse ScreenSharers with primitive or simple methods (that just check .exes, just .dlls and so on), for example, System Informer doesn't support Chinese and Georgian alphabets, thus making any device with these two default languages even tougher to ScreenShare, as every single folder can be put to Chinese/Georgian and USN Journal will not recognize these languages either (displaying them as question marks ?)

(Section redacted)

(Section redacted)

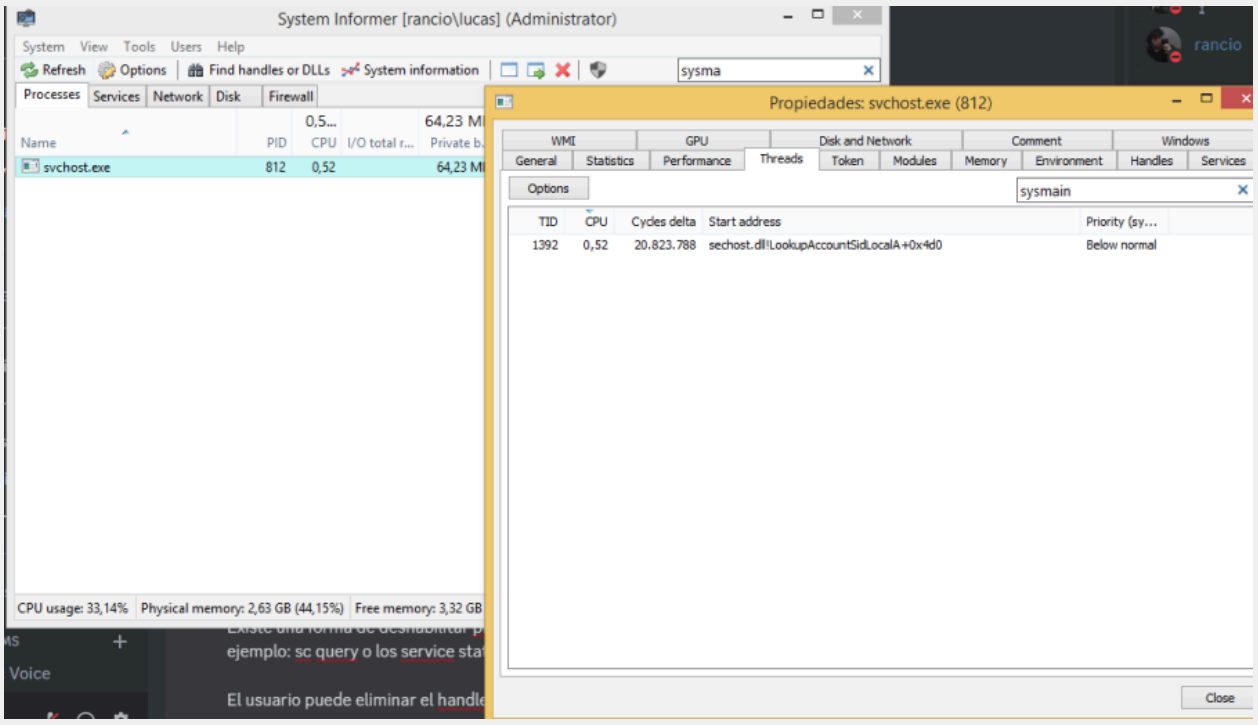
(Section redacted)

RunPE (VBA)

When we hear “RunPE”, we usually mean what’s called a “word bypass”, there’s also “PE Injection” that refers to process hollowing. MS Word allows users to run scripts / cheats via scripts through the RunPE function, making it just harder to prove by date of execution. Most of the players also run codes written in color white, if the SSer doesn’t select a region on the canvas - scripts are invisible.
Bypassers from that period also corrupted their CMD (either making them 100% transparent, for example, or making text match the terminal background and so on)

Suspending Threads & Tokens

A more recent (2021~2022) development of bypasses has led to methods that bypass services and drivers (for example, BAM) without actually disabling them. For example, suspending sechost.dll Thread of Sysmain will prevent it from logging .pf traces!



- List

Sysmain: sechost.dll
pcasvc: pcasvc.dll
dps: dps.dll
eventlog: wevtsvc.dll
dusmsvc: wlanapi.dll
cdpusersvc_[numbers]: ucrtbase.dll

『Chapter V』

📑Anti-Forensic Labs

Bypassers mostly take inspiration from Anti-Forensic Labs, Github (“skidding” source codes to resell them), YouTube or old gen Bypassers (for example, Dream Client’s Task Scheduler Bypass has been adapted by many cheaters and newgen Bypassers, improving its bypassability to sell it as handmade manual bypasses!)
Of course, in this Chapter we will not talk about already known and popularized Bypasses, we’ll cover yet undiscovered, smarter, more effective and complicated bypass methods that will confuse a lot of ScreenSharers.

Anti-Forensics

Extensionless Files

The most known types of a file used in bypasses are either renamed extensions, or no extensions at all! Extensionless files are smartly usable, as Bypassers can fake them to seem like they’re folders (by using hidden characters), launch them using WMIC or because every ScreenSharer filters .exe / .dll or other extensions in the SS - avoid getting caught by any SSer if they forget extensionless files exist.

(Section redacted)

(Section redacted)

(Section redacted)

(Section redacted)

(Section redacted)

(Section redacted)

(Section redacted)

(Section redacted)

『Chapter VI』

Blue Team DFIR🧿

Digital Forensics

Blue Teamers counter the Red Teamers in the field of Digital Forensics, just like ScreenSharers and Bypassers. We, ScreenSharers, are the Blue Team. Red Lotus has developed ScreenShare practices relying on public DFIR utilities, and we have also developed our own approaches to cheats / bypasses spread over in the Community. Thus, before covering all the never-before-seen bypasses and Red Team anti-forensics, let's begin with Universal methods for detecting malicious files regardless of what bypass or execution was used!
For that, we'll be using a lot of DFIR sources, utilities and approaches to general malware detections.

Forensics Tools

Here are the repositories to begin with:
https://github.com/mesquidar/ForensicsTools#live-forensics
https://academy.cyber5w.com/courses/
All Resources - Magnet Forensics
AndrewRathbun/DFIRMindMaps
AndrewRathbun/DFIRArtifactMuseum
https://dfirmadness.com/
windows-forensic-artifacts/execution
https://github.com/mikeroyal/Open-Source-Security-Guide

The following Chapter will cover necessary logic to read every relevant artifact.

Activities Cache

Introduction to Activities Cache

General Understanding

When installing Windows, users are met with security options they choose to enable or have disabled, disabling options that suggest your data tracking will install windows with services like Diagtrack disabled and Activities Cache encrypted.
“Windows activity history keeps track of activity on a device, such as application and services usage, files opened, and websites browsed.” - [Source]
The service and the ‘ActivitiesCache.db’ database existed before the 1803 upgrade (May 2018), but with limited functionality. Another possibly related* activity store location is at:‘Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.taskflow.shell activities\Current’; which means that we are able to access this Artifact starting from Windows 10 Builds before and after 1803.
Win+R & msinfo32 will display what version the user is currently on.

Item	Value
OS Name	Microsoft Windows 11 Home Single Language
Version	10.0.22631 Build 22631

On these versions, let’s presume we want to use Registry data to read user activity. While examining a Twitter post to analyze the given artifact, user @gazambelli has mentioned:
“So far I've seen that \xC6\x32 can be used to separate each artifact. The pattern is always: AbsolutePathToExecutable + Executable + Title. The title can be the title of the web page that was viewed or the folder name that was explored with explorer.exe.”
Further examining the given registry entry, Red Lotus has concluded that following this method in an actual ScreenShare is far from necessary, as there are dangers of being exposed with user emails. Therefore, to analyze Activities Cache on a device with Diagtrack active (sometimes, activities can be encrypted even if diagtrack is turned on), we recommend using WxTCmd.exe by Eric Zimmerman. | Introduction to WxTCmd

- Cite (Windows Timeline 18)

AMCache

Introduction to AMCache

General Understanding

Amcache is one of the most “useless” Artifacts for ScreenSharing if we want to prove date of execution rather than determine whether or not files were installed on the device. In other words, if we for some reason want to determine installation of applications on Windows 10, we may refer to this artifact. If we want to view Recent files on Windows 7, we can also refer to this artifact!
SS Community has already demonstrated its usage back in 2022 for ScreenSharing. I’ll cite their learning material.
1)[https://f001.backblazeb2.com/file/EricZimmermanTools/AmcacheParser.zip]
2)[https://f001.backblazeb2.com/file/EricZimmermanTools/TimelineExplorer.zip]
- Video Demonstration

The default directory to Amcache is `C:\Windows\AppCompat\Programs\Amcache.hve`
An alternative to Amcache for Windows 7 is syscache.hve - a file inaccessible without using FTK Imager and visiting System Volume Information. SS Community has created Ruedas to analyze just that with a Video Demonstration.

Another alternative to both Ruedas and AMCache is RecentFileCache.bcf - present on Windows 7 and Windows 10 at `C:\Windows\AppCompat\Programs\`
https://f001.backblazeb2.com/file/EricZimmermanTools/RecentFileCacheParser.zip
`RecentFileCacheParser.exe -f C:\Windows\AppCompat\Programs\RecentFileCache.bcf --csv .` or, because RecentFileCache may be in use, copy it to a different Directory (for example, downloads) and parse as shown in the clip.

And of course, Amcache is present on Windows 8 as well.
If you’re working on a Python project, feel free to refer to the Amcache parser in PY.

Extended comprehension: anssi-coriin_2019-analysis_amcache

(Section redacted)

(Section redacted)

(Section redacted)

Section Author: UnMonsieur

Windows Tasks

Introduction to Task Scheduler

Redacted

Precautions

To properly conduct the ScreenShare with this approach, you shall manage to do step 1 of this section within the first 15 minutes;

(Step redacted)

(Step redacted)

(Step Redacted)

(Step Redacted)

(Step Redacted)

(Step Redacted)

Step VII

And of course, to cite SS Community (Kendo) for their first ever method for Scheduler detections - Registry Explorer can also be used to detect scheduled task deletions, task owners, even unicode character tasks that can bypass System Informer!

- Imgur Album of the method;

EVTX

(Content redacted)
The SANS Institute has released a sheet defining EVTX Property logs:

EVTX Property logs

Source:	Thesoftwarethatloggedtheevent,whichcanbeeitherap rogramname, suchas"SQLServer",oracomponentofthesystemorofal arge program,suchasadrivername.Forexample,"Elnkii"indic atesan EtherLinkIIdriver.
Event ID	Anumberidentifyingtheparticulareventtype.Thefirstlin eofthe descriptionusuallycontainsthenameoftheeventtype.Fo rexample, 6005istheIDoftheeventthatoccurswhentheEventLogs erviceis started.Thefirstlineofthedescriptionofsuchaneventis" TheEventlog servicewasstarted."TheEventIDandtheSourcecanbeus edby productsupportrepresentativestotroubleshootsystem problems.
Level	Aclassificationoftheeventseverity.Thefollowingevents everitylevels canoccurinthesystemandapplicationlogs: Information.Indicatesthatachangeinanapplicationorco mponenthas occurred,suchasanoperationhassuccessfullycomplete d,aresource hasbeencreated,oraservicestarted. **Warning.**Indicatesthatanissuehasoccurredthatcanimp actserviceor resultinamoreseriousproblemifactionisnottaken. **Error.**Indicatesthataproblemhasoccurred,whichmighti mpact functionalitythatisexternaltotheapplicationorcompon entthattriggered theevent. **Critical.**Indicatesthatafailurehasoccurredfromwhichth eapplication orcomponentthattriggeredtheeventcannotautomatica llyrecover. Thefollowingeventseveritylevelscanoccurinthesecurit ylog: **SuccessAudit.**Indicatesthattheexerciseofauserrightha s succeeded. **FailureAudit.**Indicatesthattheexerciseofauserrighthas failed. IntheEventViewernormallistview,thesearerepresente dbyasymbol.
Log	Thenameofthelogwheretheeventwasrecorded.
Computer	Thenameofthecomputeronwhichtheeventoccurred.Th ecomputer nameistypicallythenameofthelocalcomputer,butitmig htbethename ofacomputerthatforwardedtheeventoritmightbethena meofthelocal computerbeforeitsnamewaschanged.
ProcessorTime	Theelapsedexecutiontimeforuser-modeinstructions,in CPUticks.
UserTime	Theelapsedexecutiontimeforuser-modeinstructions,in CPUtimeunits.
CorrelationId	Identifiestheactivityintheprocessforwhichtheeventisin volved.This identifierisusedtospecifysimplerelationshipsbetweene vents.
ProcessID	The identificationnumberfortheprocessthatgeneratedthee vent.
Thread ID	Theidentificationnumberforthethreadthatgeneratedth

Event (Section		eevent.	IDs Redacted)

(Section Redacted)

(Section Redacted)

ELEX (Event Log Explorer) can be used to monitor logged events in live instances.

Usage

Windows Event Logs can be used to determine when users Logged on, whether or not their Explorer crashed, for example, also to check PCA logs, Journal Deletion, Time change, event log deletions, USB logs, virtual mount logs and so on.

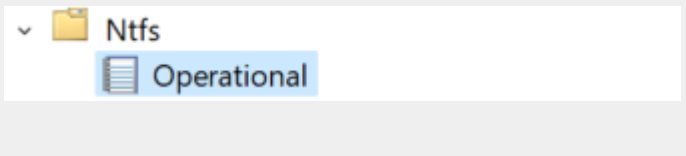
Step I - User Activities

Using event ID 4626 we can determine whether or not the current user configured file, registry or folder permissions onto themselves. ID 4647 is user Logoff time, IDs 4704 and 4705 refer to assigning or revoking user rights; ban evaders may use the same device with different accounts, this can be caught using the event ID 4720 and 4722, to trick SSers they can also disable them - shown in ID 4725 and 4726 shows account deletion. 4738 refers to account change.

This can be useful to target ban evaders or alt accounts who use multiple Windows users to evade. For servers that don't freeze ingame, it could be useful to know user activity event IDs.

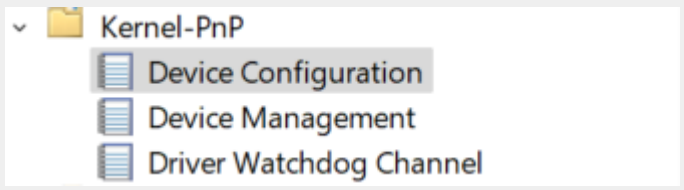
Step II - NTFS Operational

Using event ID 4 on NTFS-Operational event logs, we can detect mounted and dismounted drives. Relevant IDs are also: 9, 10, 158, 147, 149. In general, check every event ID on a specific date. This type of event data can also be used as a replacement for HWID marks, logging disk IDs are more specific and useful than HWIDs.



Step III - Kernel-PnP

Event IDs - 400, 411, 410, 430 in Kernel-PnP sect. Management/Config display generally logged data regarding USB devices, including Virtual Mounts, dismounted or mounted drives and so on. Check relevant date events for further intel. Main channels are Device Configuration and Device Management. Dismounted USBs will be logged as "surprisingly dismounted".



Step IV - Application IDs

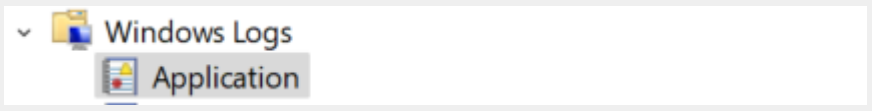
Under Windows Logs - Application it is possible to gather and check a lot of information. For example, if we would like to confirm whether or not a player lies while saying their Explorer crashed, it is possible to check that by analyzing a followup chain of IDs 1000, 1001, 1002, these IDs correspond to an app hang - app crash and app reload instances. They mainly always follow one another if an app crash was actual. However, it is worth noting that even if someone's Explorer crashed but you find they've accessed Process Hacker 2 / System Informer during the day at any given instance, it is safe to ban for it, considering usually app hangs are caused (although by analyzing data in these event Ids you can check so what caused the hang) by nullified bytes.

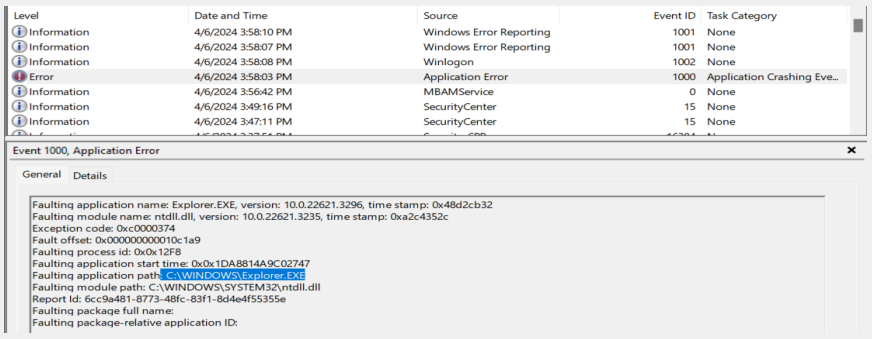
On the other hand, event ID 0 shows which service was stopped.

Event ID 1 shows which service was started.

Event IDs 1035 and 1040 show MSI installations (some cheats are .msi based)

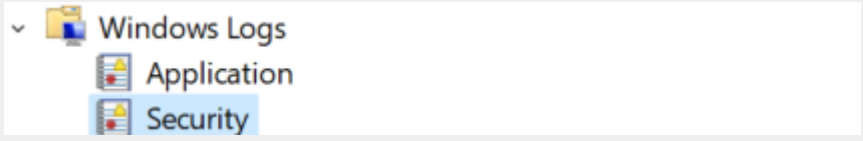
Event ID 1023 shows NET runtime errors.





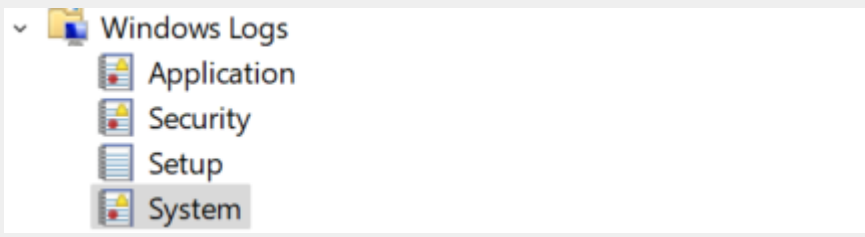
Step V - Security

Security Event logs relate to user activities on the device. It can show User Logins 4624, special logon 4672 (with custom privileges), most importantly - system time change 4616. As well as ID 1102 - Event Log clear, which means that an event log was cleared by the user. This ID task often also shows which event Log was cleared. Bannable offense.



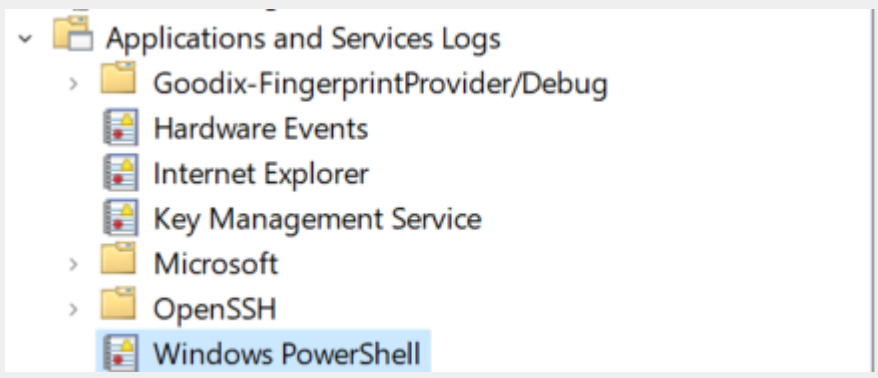
Step VI - System

The System Event log can be used to determine when users closed System Informer or Process Hacker (ID 1) when they loaded it (ID6) or check if users modified service start settings, for example, of Sysmain (ID7040) or HTTPService logs (ID112) or device shutdown (ID1074) so on and so forth! seems to be the most useful channel so far, right?

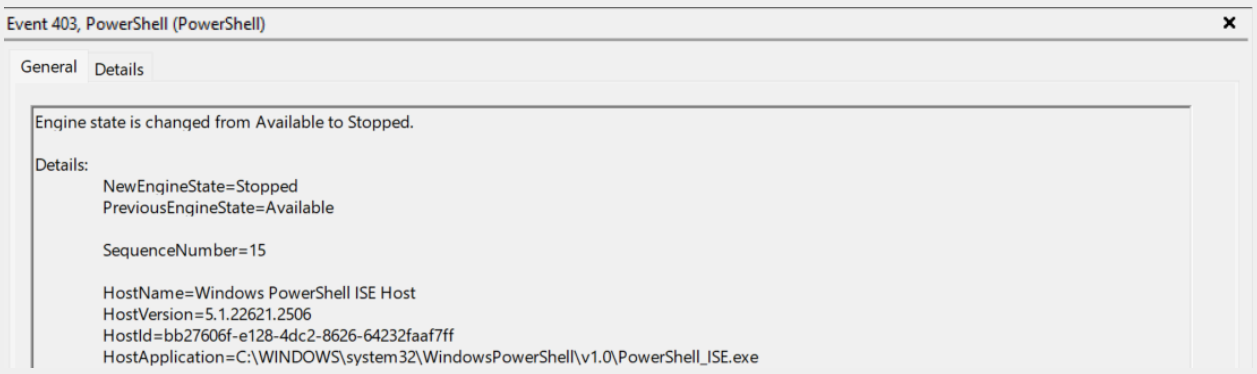


Step VII - Windows PowerShell

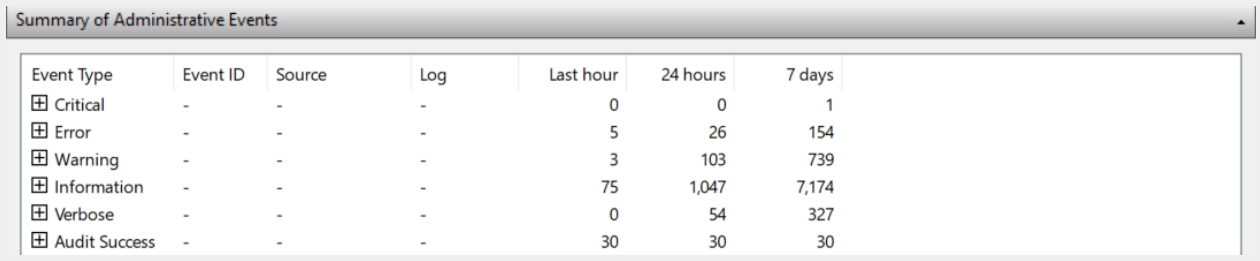
Under Application and Services logs, Windows PowerShell logs powershell commands from both PS and Powershell ISE (which does not generate psreadline logs unlike the regular powershell), however, we would need to learn to differentiate between user initiated commands and automated scripts such as these on AMD for example: *HostApplication=powershell.exe -ExecutionPolicy Restricted -Command $Res = 0; $Infs = Get-Item -Path ($env:WinDir + '\inf*.inf'); foreach ($Inf in $Infs) { $Data = Get-Content $Inf.FullName; if ($Data -match '\[defaultinstall.nt(amd64|arm|arm64|x86)\]') { $Res = 1; break; } } Write-Host 'Final result:', $Res;*



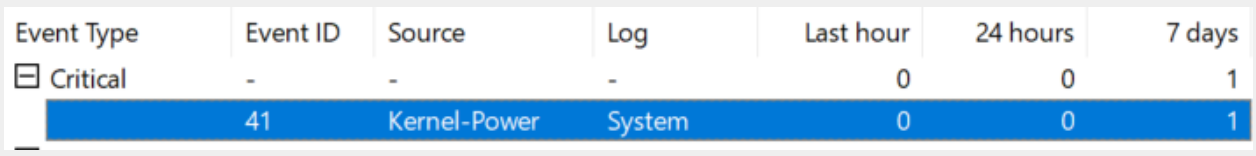
It is an automated AMD script, I've highlighted relevant info that tells us that above. So, how do we get around with this? Well, Event ID 403 in Powershell shows if the terminal was even launched! Thus any command or script after 403 is most likely user executed. After all, you can just copy the command and re-run it to see what it does.



Step VIII - Summary of Administrative Events

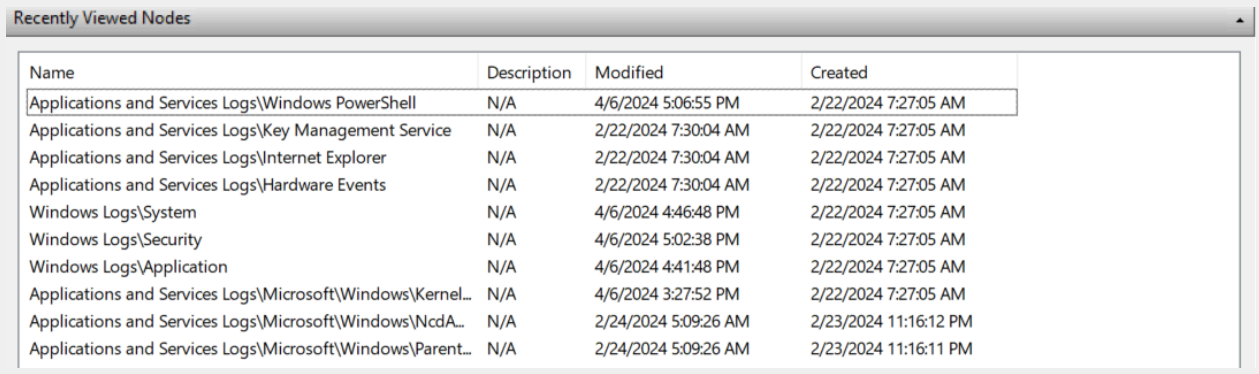


At last to not dive too deep into every single event log, we can generally view every severity of Event IDs under Summaries. Enrolling the Event Types will display which channels have updated, with Event ID, Source, Log and time span displayed next to it.



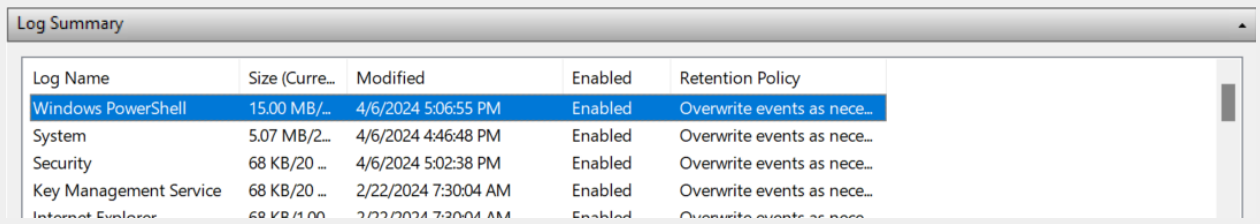
Event Type	Event ID	Source	Log	Last hour	24 hours	7 days
Critical	-	-	-	0	0	1
	41	Kernel-Power	System	0	0	1

Step VIV - Recently Viewed Nodes

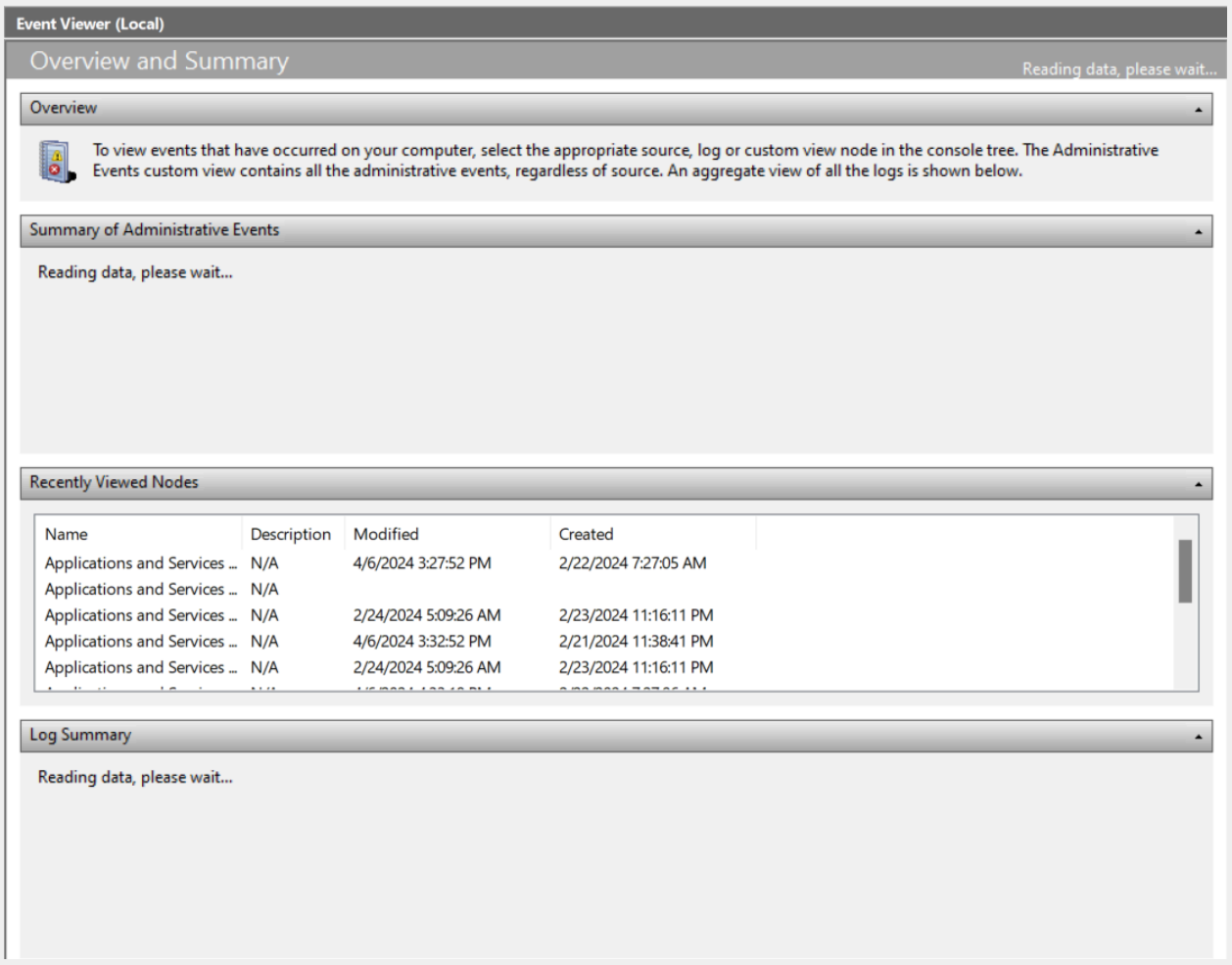


With this, if users use Event Viewer itself to clear logs, we will be able to determine which nodes they accessed recently. It logs user accessed Nodes in session, with dates of modification presenting last access / visit times.

Step IX - Log Summary



By far the most useful section in Event Viewer. It displays Logs with Date Modified (new event IDs being logged) their state (enabled / disabled inheritance) with the possibility to double click on them, which takes us to the selected node to analyze them. For example here, if we double click on Windows PowerShell we will be taken to the PS Event Logs themselves! All of this is loaded when launching the event viewer.



If loading the event viewer returns an error, open CMD and query eventlog service itself by initiating “sc query eventlog”; it is bannable to have this service disabled at all cost.

(Step redacted)

Step XI - Hayabusa

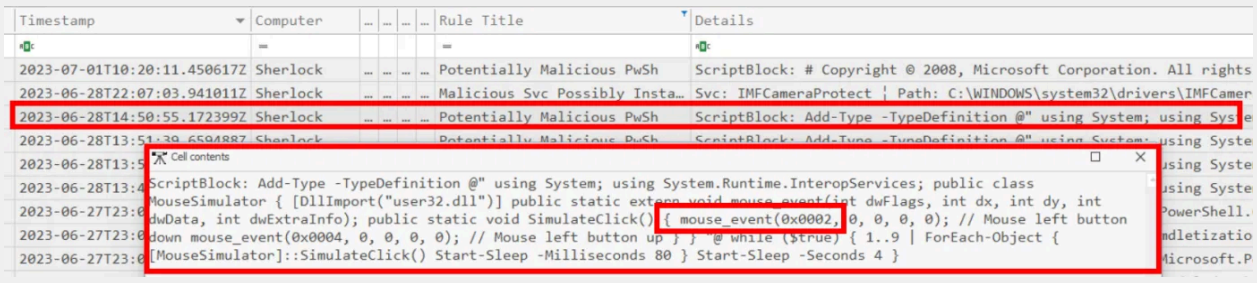


Hayabusa is an event log parser developed by YAMATO Security, Red Lotus has already held webinars in regards to explaining this utility and its use in General, however, we have never mentioned it is capable of working on [Linux with Timesketch] as well. Besides that, here's how it works on Windows:

[Download Here]

Select the Hayabusa build (I'm on `hayabusa-2.6.0-win-x64` in this case) and launch the following command: `hayabusa-2.6.0-win-x64.exe csv-timeline --output CSVOutput.csv -d C:\Windows\System32\winevt\Logs --HTML-report HTMLOutput.html --ISO-8601`

Enable log parsing without restrictions, HTMLOutput.html will show general information regarding detected traces, however, CSVOutput.csv is the data we'll read using Timeline Explorer to look for whatever we need. We'll be able to analyze specific rules by modifying "Rule Title". For example here, we've just found an auto clicker powershell script by analyzing Malicious PWSH.



Here's a timeline explorer filter to automatically load useful rules:

[Rule Title] In ('Alternate PowerShell Hosts - PowerShell Module', 'Change PowerShell Policies to an Insecure Level - PowerShell', 'Defender Alert (Severe)', 'Logon (System) - Bootup', 'Logon (Unlock)', 'Malicious Svc Possibly Installed', 'Possible Hidden Shellcode', 'Potentially Malicious PwSh', 'Powershell Create Scheduled Task', 'Powershell File and Directory Discovery', 'Proc Exec', 'Suspicious Service Installation', 'Usage Of Web Request Commands And Cmdlets - ScriptBlock', 'VHD Mounted', 'VHDX Mounted', 'Windows Defender Malware Detection History Deletion', 'Windows Defender Threat Detection Disabled', 'Windows Service Terminated With Error')

[Here's an extended information regarding Timeline Explorer and how filters work;]

Step XII - Chainsaw

Yet another event log parser, older and more versatile in a sense of modifiability - [Chainsaw] can allow users to create their own Hunt rules, modify the way they're sorted and so on. How does it differ from Hayabusa? What are its advantages? well, there's not much difference besides their purposes. Chainsaw has powerful threat hunting modules and Sigma rules (also modifiable by users) that orient on experience of DFIR specialists while hunting anomalies on the device. It is able to detect the Word bypass, for example (as MS word logs in Event Logs, both Hayabusa and Chainsaw are capable of catching odd behavior) - Here are two Demos of the utility;

[Demo 1]

[Demo 2]

(Step redacted)

Step XIV - Eventlog AutoScans

If all of these steps are too much and you just want to detect specific IDs on the device, you can use Powershell and create a script. For example, let's take this line in a code that checks for Event IDs:

```
$EventIDs = @(3079, 1102, 2003, 7, 7036, 7045, 1001, 1002, 104, 1074, 4663, 7036, 4688, 4660, 4616, 11707)
$EventLogs = @("Application", "System", "Security", "Windows PowerShell", "Windows PowerShell Operational", "Setup", "ForwardedEvents", "Internet Explorer")
```

In $EventIDs you may replace specific numbers yourself (here for example, 3079 is a USN Journal clearance check in $EventLogs, which means it checks every specified Event Log name for the Event ID thus the Powershell will print a lot of errors before finding 3079 in its proper log); So the second part of this script would be something like:

```
$Results = @()

foreach ($EventLog in $EventLogs) {
    $Events = foreach ($EventID in $EventIDs) {
        Get-EventLog -LogName $EventLog -InstanceId $EventID -Newest 10
    }
    $Results += $Events
}

$Results | Export-Csv -Path ".\InstantParse.txt"
```

Which means that for every EventID detected in EventLogs it will scan the Newest 10 entries of that ID and print that in InstantParse.txt file after the scan is finished. The scan is finished when Powershell is done printing Errors (it prints Errors in the terminal to avoid being printed in the .txt results itself); The finished script looks like this - modify as you wish:

$EventIDs = @(3079, 1102, 2003, 7, 7036, 7045, 1001, 1002, 104, 1074, 4663, 7036, 4688, 4660, 4616, 11707)
$EventLogs = @("Application", "System", "Security", "Windows PowerShell", "Windows PowerShell Operational", "Setup", "ForwardedEvents", "Internet Explorer")
$Results = @()

foreach ($EventLog in $EventLogs) {
** $Events = foreach ($EventID in $EventIDs) {**
**   Get-EventLog -LogName $EventLog -InstanceId $EventID -Newest 10**
**   }**
** $Results += $Events**
}

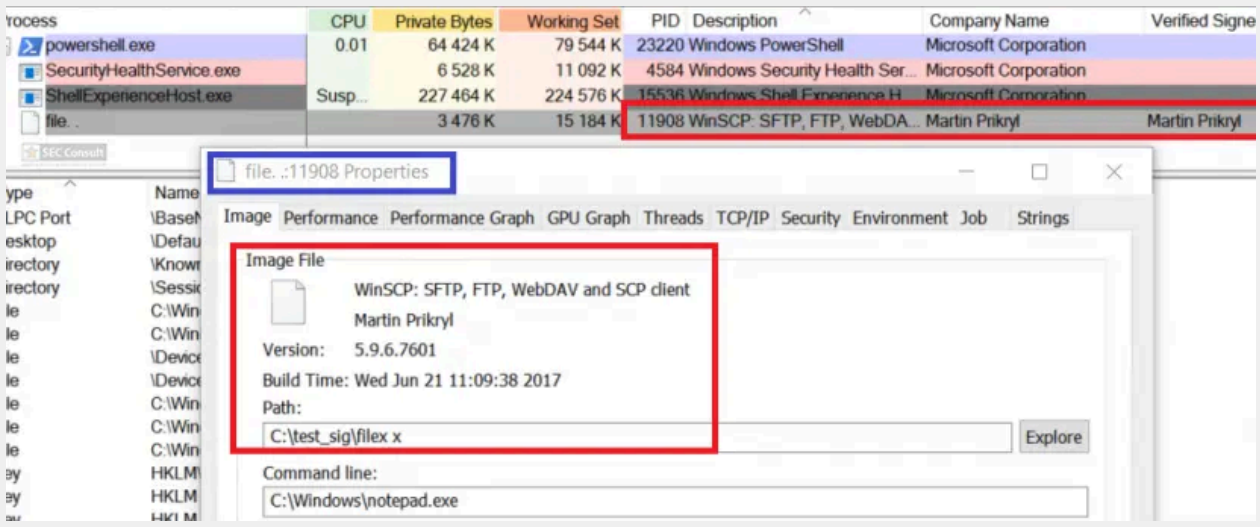
$Results | Export-Csv -Path ".\InstantParse.txt"

(Step redacted)
(Step redacted)

ADS Detections

Introduction to ADS and (Redacted)

Detecting Alternate Data Streams



As shown in the blog mentioned above (Anti-Forensic techniques), a weird file in C:\Test_sig\ directory is executed, it has no extension nor an icon, because it is an Alternate Data Stream, a file that is hidden as a data stream. While creating an ADS or applying it to a file/folder, executing then deleting the Stream itself, USN Journal will obviously create a “Stream Change” entry. Try it for yourself:

type Cheatfile.exe >> C:\Windows\pathtowherever\pathorfile:Zone.Identifier

The type command will create an ADS file. For example, let’s do `type AnyDesk.exe > C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier`

Then execute this ADS using forfiles (as it cannot be executed regularly) by running `forfiles /p C:\windows\system32 /m notepad.exe /c C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier`

Then let’s use AlternateStreamView to find and delete this Stream (specify discordptb folder for example) - we’ll be met with a Stream Change | Close on this file itself, which is more than enough to ban for if caught doing before SS. As you see in the screenshot above as well, file.: has a : mark after it. What does it mean? - it’s an ADS identifier, the file hidden as an ADS will always end with : and Zone.Identifier or $DATA will always be assigned to it. For example, to complicate things we’ll hide Anydesk as: `type AnyDesk.exe > C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier:$DATA`

Thus with forfiles we’d need to execute it alongside with $DATA and usual detection methods might be more challenging to detect these with. If the file is kept on the device, Alternate Stream View will be able to find it, exporting and modifying the extension of ads to .exe is more than enough as it also shows date of modification when launching the cheat. You can select an entire drive to scan too.

If the ADS is executed using a scheduled CMD, it will create a BAM entry but it won’t log in Prefetch. Users also usually delete the ADS logged in BAM. The following steps detect ADS regardless of what bypass was used, even if deleted:

(Step redacted)

(Step redacted)

(Step redacted)

(Section redacted)

『Chapter VII』

Collection of Evidence



Red Lotus Collector was developed by the Headquarters of Red Lotus to aid ScreenSharers during their sessions, aimed to optimize the process and speed of each ScreenShare that otherwise would have lasted for much longer. Practice has shown that by the use of this utility, on average users save 20~25 minutes in total; it has also proven to have enhanced the quality of ScreenSharing - especially for those who have attended RL Webinars during the first release of this utility and for those who are aware of the “secrets” of this tool for better usage.

Usage of the Tool (for visualization)

Copyright (c) 2024 Red Lotus

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The utility can be run both via CMD (as Admin) or by Right Clicking on it (to execute as Admin), such launch guarantees that the collection material it downloads and executes against evidence artifacts runs without errors.



The tool depends on [Eric Zimmerman's tools] to initiate a scanning process, individually it affects or analyzes or collects nothing else. Utilities it downloads and executes include:

file_urls = [
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/bstrings.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/EvtxECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/PECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SBECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SQLECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SrumECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SumECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/WxTCmd.zip"
]

Data it collects is the following:

- **EventLogs [Evtxecmd_Output.csv]**
- **Registry (Fully) [RECMd_Output.csv]**
- **AppResources [SrumECmd]**
- **Prefetch entries and details [PECmd]**
- **Traces of DLLS, EXES, PATHS and existing Scheduled Tasks**
- **Activities Cache & activity operations**
- **Individual Registry entries (BAM, Mounted Devices, etc)**

Using the following data, we can check:

- **Executed .exes**
- **Executed renamed extensions**
- **Injected .dlls**
- **Injected renamed extensions**
- **EventLog manipulations**
- **Registry Manipulations**
- **Traces of executed C# / C++ files or scripts, executed python scripts**
- **Uptime and Shutdown of Processes & Applications**
- **Prefetch traces to troubleshoot broken Winprefetchview**
- **All the existing registry data**
- **USB Related activities**
- **File movement related activities**
- **Scripting environment related activities**
- **Software uninstall related activities**
- **Startup and scheduler related tasks and activities**
- **Dates of executions**
- **Remote and Local USER IDs**
- **Alternative Users (Ban Evasion via alternate microsoft accounts)**
- **And much more. . .**

Evidences of Execution

When it comes to understanding of the term “PE Execution”, many SSers refer (or interpret it) to it as “Process Hollowing”, “doppelganger” and many other terms that sound confusing or scary to many as it seems like there’s no way to detect them. Almost nobody explains and clarifies that **every .exe and .dll is a portable executable format.** Let’s give this sentence a clearer explanation:

Portable - Refers to the ability to execute the given .exe or .dll on any Windows architectures, be it a x32 or x64 or x86 system. Each portable executable has a “header” (unique identifier, also called as “DPS Strings” in communities);

Executable - A compiled physical file that can be executed by the system. In this sense, think of executables as any extension files that can be launched. On the other hand, files that launch before compilation, in the shape of codes, are

called **header files.** Header files are known as “fileless malwares” in the given communities. A form of a header file can be used in what we call a “shellcode injection”;

Now that that’s established, Collector of Red Lotus provides multiple .csv extractions that collect evidence of portable executable executions, as well as potential traces of header file executions. The following .csv files collect PE Execution evidences:

- **Activities Cache (Activity, Activity Operations)**
- **BAM/DAM (BAM for Windows 10+, DAM for Windows 8)**
- **SrumECmd (AppTimeline Provider, PushNotifications)**
- **Registry (RADAR Software, FireWall Rules_SYSTEM, AppCompat...)**
- **PECmd (Both outputs)**
- **Evtx (Event Logs for Mediu-Severe activities)**

We will be discussing each and every detail of the utility and these collections, as well as situational analysis for better understanding below:

-BAM/DAM

Background Activity Moderator still has many uses. What SSers do not know or were not made aware of is that it also logs data of applications on startup, by this, every Scheduled task always logs in BAM - hence why Bypassers need to clear its traces from BAM each and every single time. What’s worse is that sadly BAM only rarely keeps data of cheat files under execution traces upon closing it;

Example A:

SSer (A) ScreenShared a Suspect (B) and found Vape V4 executed right before AnyDesk. The confused SSer banned Suspect B, awaiting their appeal to clarify what could’ve happened.

The SS Manager of the server demoted the SSer and put them under supervision because the suspect argued that an SSer executed their cheat file before SS - after all, why would cheaters run files right before SS.

BAM has the capability to log closed files as execution, thus any cheat logged in BAM before ScreenShare means that the cheater closed their client poorly.

Such cases are very rare, but they do happen. Bam is also full of extensionless and path-less “files”. What really are they?

Program	Batch Key Path
\Device\HarddiskVolume3\Windows\System32\Taskmgr.exe	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001
\Device\HarddiskVolume3\Users\Amade\AppData\Local\Programs…	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001
\Device\HarddiskVolume3\Windows\System32\cmd.exe	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001
Microsoft.WindowsTerminal_8wekyb3d8bbwe	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001
\Device\HarddiskVolume3\Windows\System32\mmc.exe	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001
Microsoft.WindowsNotepad_8wekyb3d8bbwe	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001

In the “Program” column, traces indicate either paths to physical files or alerts of activities. In this instance, it isn’t accidental or unusual that Microsoft.WindowsTerminal activity logged before cmd.exe and the User ID next to it indicates that the action was called by a user. It’s also not accidental that each User ID has different numbers.

e	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1001	2024-
	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1002	2024-
_cw5n1h2txyewy	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1002	2024-
_cw5n1h2txyewy	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1000	2024-
	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1000	2024-
onFlow_cw5n1h2txyewy	ROOT\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-729616088-176994416-1935787870-1000	2024-

In my case, the local user is 1001. Other actions were issued by the system or any other application and so on. **This is a crucial bit of information while reading BAM to avoid false-bans.**

Example A:

SSer (A) ScreenShared a suspect (B) and started analyzing BAM. In BAM results, they saw LockApp_[ID] on User (1001) with Windows.Terminal_ID right before it, also on User (1001). The SSer didn’t mind much of it and continued to SS. In the end the suspect got banned for disabling DiagTrack.

What should’ve been the thought process of SSer A?

The ScreenSharer should’ve realized that Lockapp is the default Windows lockscreen function that only shows up during the user logon operation (the timestamp will indicate when the User launched their device, for example), Windows.Terminal is often seen in Windows 11 and refers to accessing CMD on the device. User ID (1001) means that all of these actions were initiated on the given user (if the SSer confirms AnyDesk was also under 1001), thus it is only logical that the player accessed CMD and set their device on lockscreen (a tactic often used by bypassers while modifying system time to fake the duration of services uptime) - all of that occurring right before ScreenShare can indicate that Conhost or Consent were logged in Prefetch that could show if they had initiated setting commands (Windows settings logging in Conhost) or launched disablers and files using it (files / .bat files log in Consent/Conhost)

BAM is a windows functionality that acts like DPS and Prefetch - they all log .exe files only (not .dlls, not .jars or any other extension), thus **anything other than .exe logging in BAM must be investigated.**

Example A:

SSer (A) ScreenShared a suspect (B), in BAM they skimmed over a file named “message.txt”, the SSer checked shell:recent, checked the message.txt but it included emails and passwords in them.

The SSer got demoted for invading user privacy and not following safety protocols of the server.

What should've the SSer done?

As I mentioned above, BAM only logs .exe files. Therefore - message.txt was a renamed executable that the user modified later on with potentially fake emails and passwords. The ScreenSharer could have banned them right away for launching renamed executables (it would also log in Prefetch, but assuming it did not then the file was a scheduled task)

-Renamed Extensions

Besides BAM, renamed extension files will log in Prefetch, DPS and shell:recent (alongside with csrss, searchindexer and every other common service); RL Collector collects JumpLists and RecentDocs data (both related to recent files) that ScreenSharers can rely on.

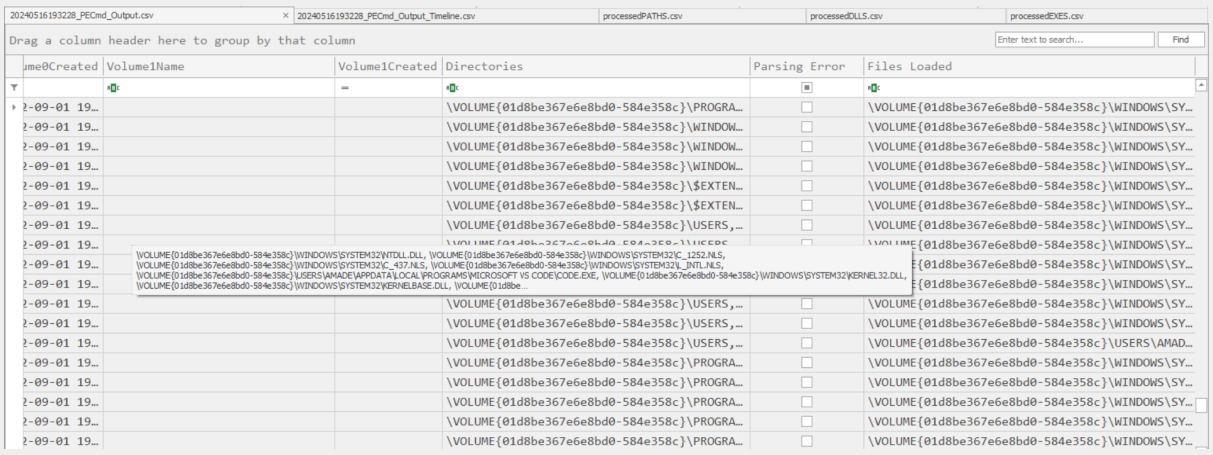
RecentDocs and JumpLists:

In week 3 of our webinars, we discussed important details about shell:recent, its connection to recentdocs and how JumpLists are the main source of Recent access data even if the registry entry of recentdocs is cleared. This artifact is located in shell:recent\AutomaticDestinations and shell:recent\CustomDestinations on devices that have registry logging for recentdocs disabled as well.

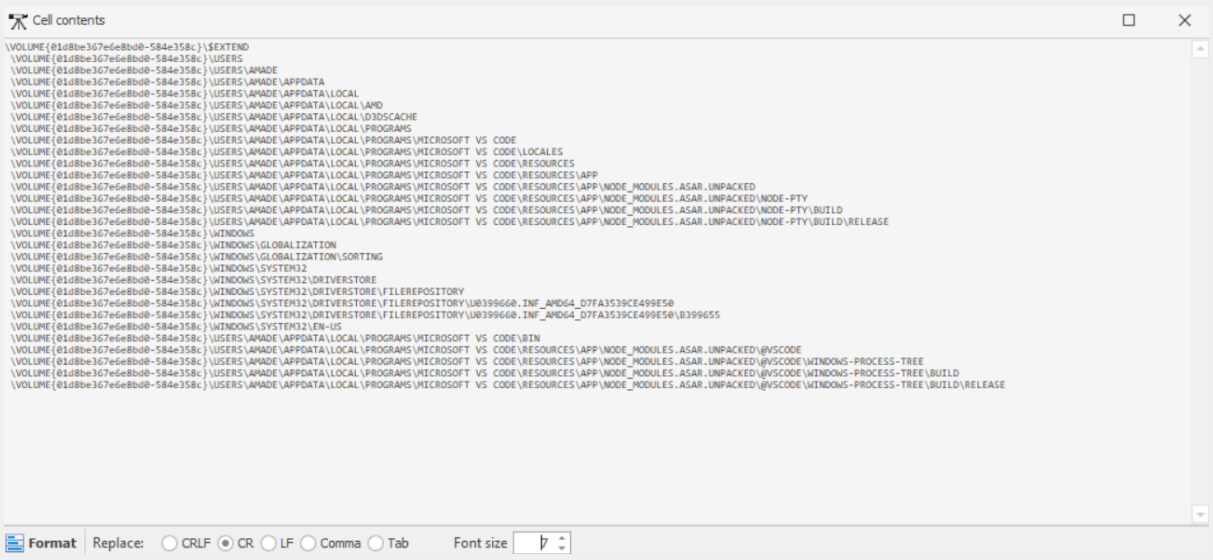
JumpList Explorer / JLECmd can be found at Eric Zimmerman's tools

Here's a demonstration video: JumpLists Explorer - Demo (RL)

The artifact can also detect extension-less files. On the other hand, Prefetch and technically every other service or evidence of execution artifact (Even registry entries) will log all files regardless of their renamed extension. Let's review how you could utilize the free methods of Red Lotus to further detect not only renamed extensions but also other potentially malicious files on the system.



As we already know, PECmd extracts PECmd_Output (which includes loaded file and directory details), by double clicking on them, we'll be met with "Cell Contents" which we can sort with path per line. Click CR then Format in the below menu to do so.



After that, we can copy all of them and create a paths.txt file as shown in BACA's signature checker to analyze all of the given files for signatures! - extensionless files or renamed extensions will keep flagging as InvalidSignature and if they fake the file signature they'll log as HexMisMatch.

powershell Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass && powershell Invoke-Expression (Invoke-RestMethod https://raw.githubusercontent.com/bacanoicua/Screenshare/main/RedLotusSignatures.ps1)

Video Demonstration in General: https://streamable.com/5wk2m3

This is a bit of a tactical approach to pretty much any path I'd like to share with the readers of this guide. This method can be used on any paths.txt with a list of file paths - even extractions from services or other timeline extractions, including Search Everything extractions.

Speaking of Search Everything, by using the following string (and if you want to - modify the file sizes): **!.exe !.dll file: size:5mb..30mb** SSers can sort search everything with non-exe and dll files between the sizes of 5mb and 30mb (again, you can modify this, though it'll be troubling if the user's cheat is 1mb large... you'll have to use other methods as well.) we can basically determine renamed extensions or extensionless unsigned files using this method to be on the device. After finding suspicious flags, SSers can search for them collectively in each artifact or service.

Example A:

SSer (A) ScreenShared a suspect (B) and used the Signature Checker method. They were met with the following results:

Path	Signature	User	SID	Regpath
C:\Users\Amade\AppData\Roaming\Zoom\bin\CptHost.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Users\Amade\AppData\Roaming\Zoom\bin\Zoom.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Users\Amade\AppData\Local\Kingsoft\WPS Office\12.2.0....	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Program Files\SystemInformer\SystemInformer.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Program Files\AccessData\FTK Imager\FTK Imager.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Program Files\Sublime Text\sublime_text.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Program Files\Everything\Everything.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Users\Amade\AppData\Local\Kingsoft\WPS Office\12.2.0....	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Users\Amade\AppData\Local\Kingsoft\WPS Office\12.2.0....	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Program Files\WinRAR\WinRAR.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Windows\System32\	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S
C:\Windows\System32\PickerHost.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-176994416-1935787870-1001	HKLM:\S

The SSer found nothing suspicious and let the suspect go. In two days, their whole SS was published on YouTube, clowning them.

What did the SSer miss?

The ScreenSharer missed a blank file path that has a valid signature. C:\Windows\System32\ below Winrar.exe. File paths will never flag under Valid or Invalid signature. It'll be blank.

Example B:

SSer (A) ScreenShared a suspect (B) and used the Signature Checker method. They compiled a long list of paths extracted from all around the processes (DPS, Pca, SearchIndexer, Csrss) and evidence of execution into one large file and scanned everything with the signature checker. They were challenging the suspect.

After a long wait the tool finally finished scanning. The SSer got bypassed and requested proof of execution of the cheat. The suspect provided them with the proof and showed them the tool that had **scanned the file.** This is the provided screenshot:

```
\VOLUME{01d8be367e6e8bd0-584e358c}\WINDOWS\SYSTEM32\UCRTBASE.DLL, Valid Signature
\VOLUME{01d8be367e6e8bd0-584e358c}\WINDOWS\GLQBALIZATION\SORTING\SORTDEFAULT.NLS
```

What did the SSer miss and misunderstand?

The file path is "GLQBALIZATION", not "GLOBALIZATION" which is the default and actual directory on the device. Sortdefault.NLS not having a signature is **as suspicious as it flagging "unsigned"** - because with the given tool, if files, be it renamed extensions, have BLANK signatures (AKA nothing) - it means that the file was either moved to a different directory, or it was deleted, hence why no results were found, the file was not even scanned.

-Injected & Renamed DLL

Proving the date of injection of DLLs can be very challenging. After all, evidence of execution is not used to prove DLL Injections. What can be used, however, is aforementioned Prefetch (with loaded file paths on DllHost / RegSvr / RunDLL) and the signature checker to analyze .dll files logged in csrss, searchindexer and so on, including .dlls parsed in "processedDLLs.csv" extraction of the Red Lotus Collector.

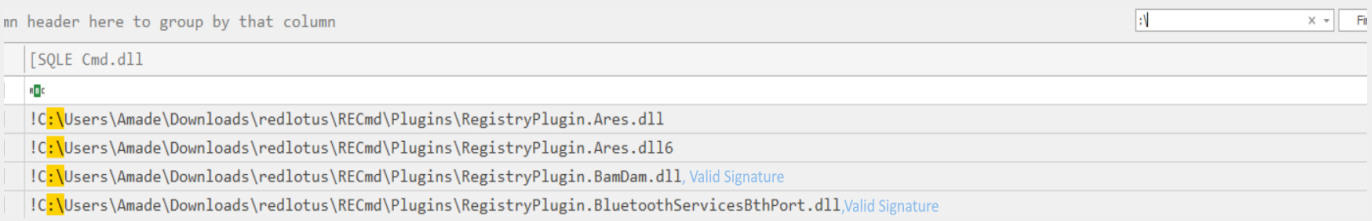
But just like everything else I've mentioned above, the community lacks a general understanding of how dll injections physically work and what can indicate an injection.

Example A:

SSer (A) ScreenShared a FiveM Bypasser who sells their client for $500... and actually gets customers for it. If the SSer wins this challenge, they are promised to be given $100 in return. SSer used a Red Lotus method and analyzed the entire system using the signature checker.

Here's the evidences the SSer collected during ScreenShare:

Filename	Run Count	Volume0Name	Volume0Seri…	Volume0Cr
	=			=
ows\Prefetch\CODE.EXE-14EEF3BB.pf	1	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CODE.EXE-14EEF3B6.pf	2	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CODE.EXE-14EEF3B5.pf	4	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CODE.EXE-14EEF3B4.pf	8	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CODE-TUNNEL.EXE-1812F29C.pf	1	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CMD.EXE-0BD30981.pf	65	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROMELAUNCHER.EXE-B78F8A34.pf	15	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA4A.pf	51	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA49.pf	2	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA48.pf	1	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA47.pf	2	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA46.pf	3	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA44.pf	66	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA40.pf	4	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA3E.pf	33	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA3D.pf	181	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHROME.EXE-AED7BA3C.pf	23	\VOLUME{01d8be367e6e8bd0-584e35…	584E358C	2022-09-0
ows\Prefetch\CHECK_HDD.EXE-0D010B5B.pf	1	\VOLUME{01d8be367e6e8bd0-58d835…	584B558C	2022-09-0

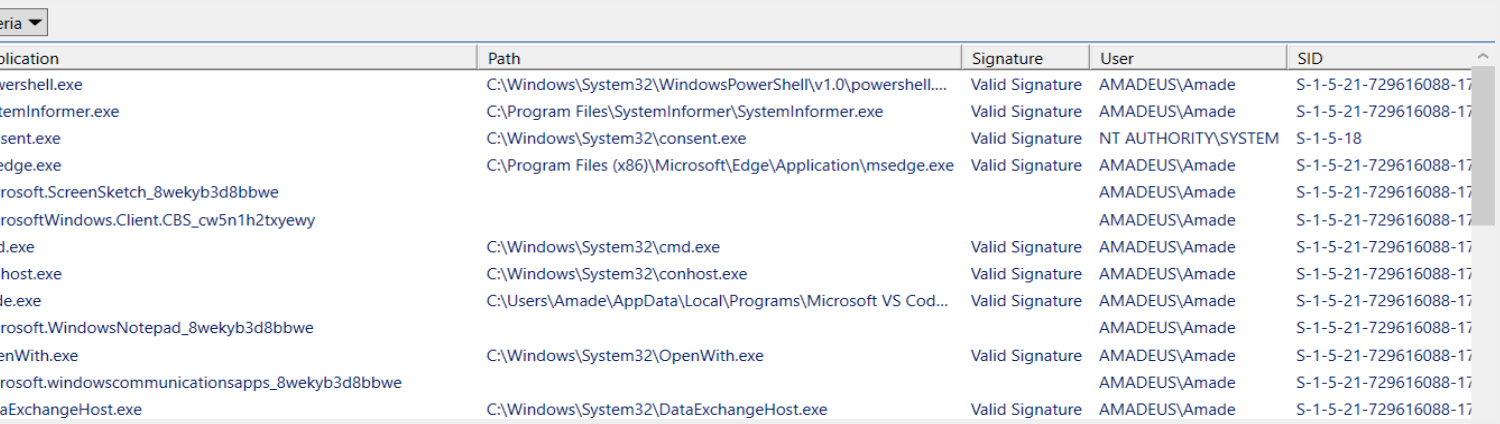


The ScreenSharer got bypassed and the whole ScreenShare was recorded to further advertise their cheat.

What did the SSer do wrong?

After closely reading the information in this guide above, you might've already guessed what happened. Let's review the images:

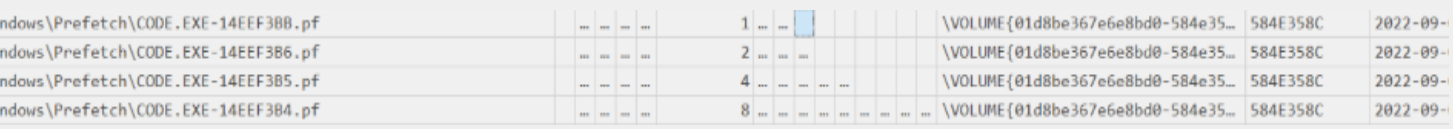
In the Prefetch image, we see a lot of weird behavior of applications, such as Chrome being run multiple times, some weird Code - Tunnel.exe, CODE.exe logging and so on...



plication	Path	Signature	User	SID
wershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell....	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
temInformer.exe	C:\Program Files\SystemInformer\SystemInformer.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
sent.exe	C:\Windows\System32\consent.exe	Valid Signature	NT AUTHORITY\SYSTEM	S-1-5-18
dge.exe	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
rosoft.ScreenSketch_8wekyb3d8bbwe			AMADEUS\Amade	S-1-5-21-729616088-17
rosoftWindows.Client.CBS_cw5n1h2txyewy			AMADEUS\Amade	S-1-5-21-729616088-17
d.exe	C:\Windows\System32\cmd.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
host.exe	C:\Windows\System32\conhost.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
de.exe	C:\Users\Amade\AppData\Local\Programs\Microsoft VS Cod...	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
rosoft.WindowsNotepad_8wekyb3d8bbwe			AMADEUS\Amade	S-1-5-21-729616088-17
enWith.exe	C:\Windows\System32\OpenWith.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
rosoft.windowscommunicationsapps_8wekyb3d8bbwe			AMADEUS\Amade	S-1-5-21-729616088-17
aExchangeHost.exe	C:\Windows\System32\DataExchangeHost.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17

This piece of evidence is more clearly explained to Lotus+ members in Tome II of Red Lotus, but rest assured - it isn't necessary to know anti-forensic techniques (here it's possible the user used an HTA exploit, basically) to detect cheats. So, what are we seeing here?

1) Consistent CODE.EXE executions (this is VSCode)



ndows\Prefetch\CODE.EXE-14EEF3BB.pf	1	\VOLUME{01d8be367e6e8bd0-584e35...	584E358C	2022-09-
ndows\Prefetch\CODE.EXE-14EEF3B6.pf	2	\VOLUME{01d8be367e6e8bd0-584e35...	584E358C	2022-09-
ndows\Prefetch\CODE.EXE-14EEF3B5.pf	4	\VOLUME{01d8be367e6e8bd0-584e35...	584E358C	2022-09-
ndows\Prefetch\CODE.EXE-14EEF3B4.pf	8	\VOLUME{01d8be367e6e8bd0-584e35...	584E358C	2022-09-

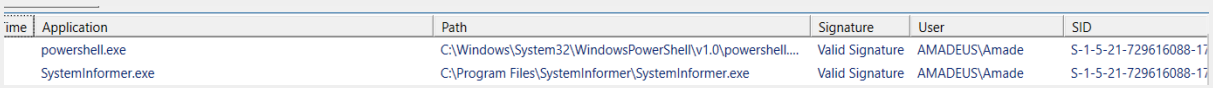
Also look at the Hash and Volume ID of this screenshot as it'll be useful later on. Ignore the dates on this one.

We can already assume that the user initiated a code to hide traces or delete them, or launch a file itself. However, what's weird is the Hash and Volume ID of CHECK_HDD.EXE

It's different from the rest of the logs. We can already assume that this is their cheat or utility they used to inject with, in short, it's a piece of evidence that requires further investigation.

C:\Windows\Prefetch\CHECK_HDD.EXE-0D010B5B.pf	1	\VOLUME{01d8be367e6e8bd0-58d835...	584B558C	

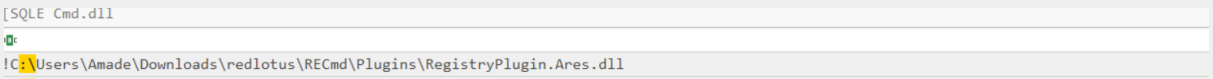
In the next screenshot:



ime	Application	Path	Signature	User	SID
	powershell.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell....	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17
	SystemInformer.exe	C:\Program Files\SystemInformer\SystemInformer.exe	Valid Signature	AMADEUS\Amade	S-1-5-21-729616088-17

We see SystemInformer accessed. Many servers ban access to Process Hacker or System Informer before SS or alongside their games, thus this will also serve as another piece of evidence.

In the final screenshot:



We see a .dll file with blank signature checks. Thus we can already assume that the file was moved out of its original directory or it was deleted. Following all the evidences we have gathered, it would be enough evidence to ban for, because:

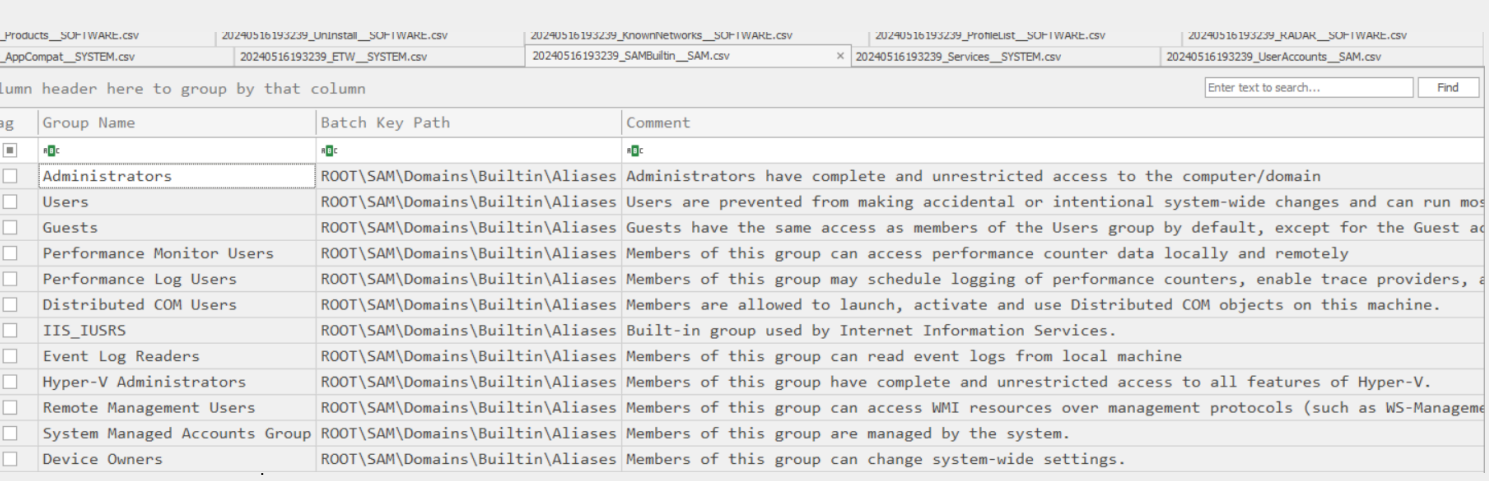
1) CHECK_HDD's hash and Volume differs from the rest (thus it was run from a different drive / volume)
2) System Informer (or Process Hacker or any other DLL Injector) run
3) Unidentifiable .dll file

The ScreenSharer should have investigated these suspicious actions to collect enough evidence for the ban. However - nobody would pay them $100 because Bypassers are just... not honorable.

-EventLog Manipulations

EventLog manipulations are becoming a trend recently, individuals modify eventlog functionalities (by disabling them, hooking, erasing logs and so on) to avoid detections via Event Viewer (however, Hayabusa tends to battle these actions very well. I'll leave it up to you to see how.)

The Red Lotus Collector also collects SAM (Registry) that has a SAMBuiltin_SAM.csv output, it allows us to see if the user has taken full authority over the Event Logs:



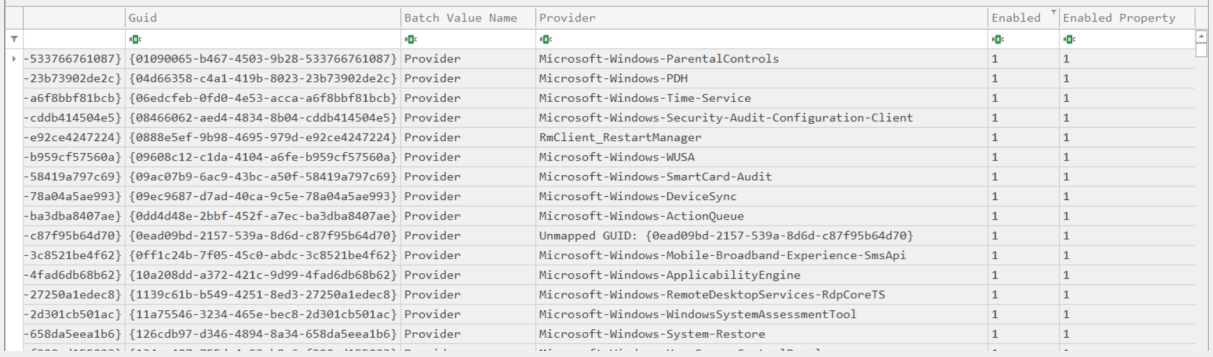
ag	Group Name	Batch Key Path	Comment
	Administrators	ROOT\SAM\Domains\Builtin\Aliases	Administrators have complete and unrestricted access to the computer/domain
	Users	ROOT\SAM\Domains\Builtin\Aliases	Users are prevented from making accidental or intentional system-wide changes and can run mos
	Guests	ROOT\SAM\Domains\Builtin\Aliases	Guests have the same access as members of the Users group by default, except for the Guest ac
	Performance Monitor Users	ROOT\SAM\Domains\Builtin\Aliases	Members of this group can access performance counter data locally and remotely
	Performance Log Users	ROOT\SAM\Domains\Builtin\Aliases	Members of this group may schedule logging of performance counters, enable trace providers, a
	Distributed COM Users	ROOT\SAM\Domains\Builtin\Aliases	Members are allowed to launch, activate and use Distributed COM objects on this machine.
	IIS_IUSRS	ROOT\SAM\Domains\Builtin\Aliases	Built-in group used by Internet Information Services.
	Event Log Readers	ROOT\SAM\Domains\Builtin\Aliases	Members of this group can read event logs from local machine
	Hyper-V Administrators	ROOT\SAM\Domains\Builtin\Aliases	Members of this group have complete and unrestricted access to all features of Hyper-V.
	Remote Management Users	ROOT\SAM\Domains\Builtin\Aliases	Members of this group can access WMI resources over management protocols (such as WS-Manageme
	System Managed Accounts Group	ROOT\SAM\Domains\Builtin\Aliases	Members of this group are managed by the system.
	Device Owners	ROOT\SAM\Domains\Builtin\Aliases	Members of this group can change system-wide settings.

We can also see quite a lot of details about the system (which users are local, Admins, what are the IDs of Guest accounts and so on) like Event Log Reader Group members. Scrolling to the right will reveal user IDs assigned to the groups:

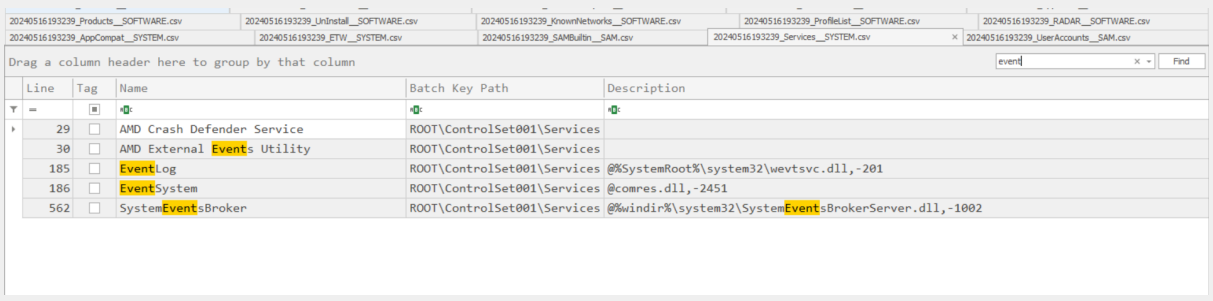
	Batch Value Name	Users
	Multiple	S-1-5-21-729616088-176994416-1935787870-500, S-1-5-21-729616088-176994416-1
ions	Multiple	S-1-5-4, S-1-5-11, S-1-5-21-729616088-176994416-1935787870-1001
h is further…	Multiple	S-1-5-21-729616088-176994416-1935787870-501
	Multiple	
event trace…	Multiple	
	Multiple	
	Multiple	S-1-5-17
	Multiple	
	Multiple	
Windows Rem…	Multiple	
	Multiple	S-1-5-21-729616088-176994416-1935787870-503
	Multiple	

In this case, there are no users controlling the given device (we can also see the local user under ID 1001, as demonstrated in previous sections of this guide!)

The **ETW System displays if Event Logs are disabled or "taken over":**



Usually, none of these providers will be disabled or have their properties disabled. It is still advised to double check their functionalities. In addition, Services_SYSTEM.csv can display the current state of services, including Event Logs, BAM and many more:



Let's discuss a hypothetical scenario that utilizes the given information:

Example A:

An SSer (A) ScreenShared a suspect (B) from FiveM who was asking to be ScreenShared. The SSer used BAM to determine what files they had executed before ScreenShare. For some peculiar reason, BAM showed executions from 2023 only, and around 5 files were executed during the day of the ScreenShare. The SSer didn't mind much of it - he's a very experienced one after all.

In the end, he got bypassed. Turns out, the Bypasser deleted their cheat and cleared USN Journal. The SSer still banned them for clearing BAM and made an exposing video claiming the bypassers are "full of shit", lying to SSers.

What could the SSer do? Was the ban justifiable?

If we follow the standards of Red Lotus while conducting a ScreenShare, this ban would be unjustifiable or lack the evidence - especially because the SSer still continued to SS. With our rules, if an SSer continues to SS for more than 5 minutes after finding a universally bannable offense (Disabled services, deleted files, other anomalies) they lose the right to ban suspects if nothing else is found on the system.

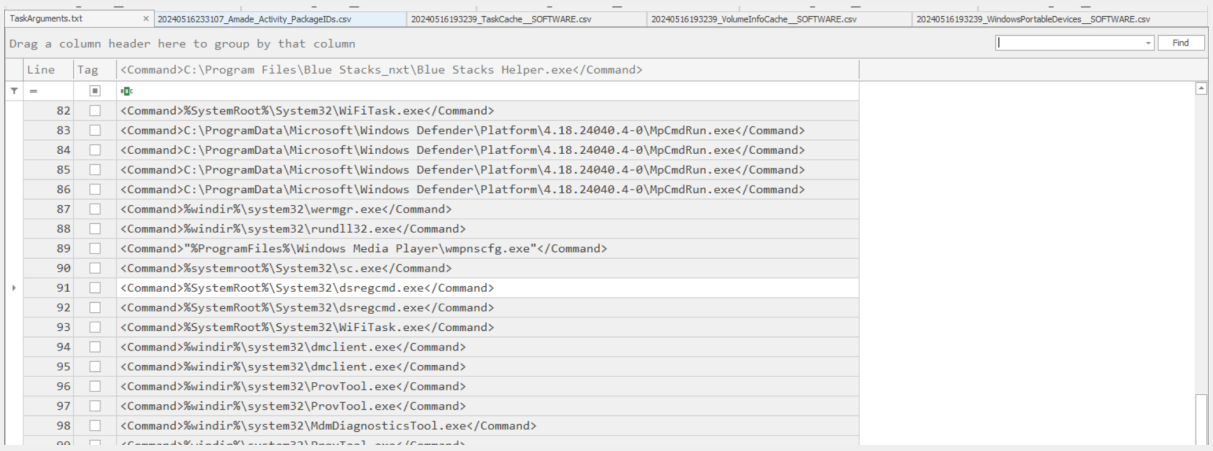
BAM showing all files from 2023 and only around 5 from the current day can indicate multiple things. The SSer should have checked for BAM value deletion, access to regedit or other anomalies - as well as user's permissions over Event Logs, uptime of event logs and aforementioned Event Log deep-dives. They could also refer to methods explained in the free release of Tome II, Book I about Event Logs.

-Temporary Traces

Red Lotus Collector collects 4 types of customized artifacts. These are processedDLLs, processedEXEs, processedPATHS and TaskArguments.txt.

These processed data .csvs export all logs that were ever temporarily saved in the %temp% directory - for example, while running a script for visual studio, loaded file paths that it relied on (or file paths of cheats it was used for) will be logged in %temp% alongside with C# and C++ info caches. Every scripting environment creates a cache data in %temp%, including Python and many, many others.

On one hand, this can also extract powershell file downloads and so on, however, it is not evidence of execution and each detail must be investigated thoroughly. Every method that I have mentioned before can be utilized with this, too.



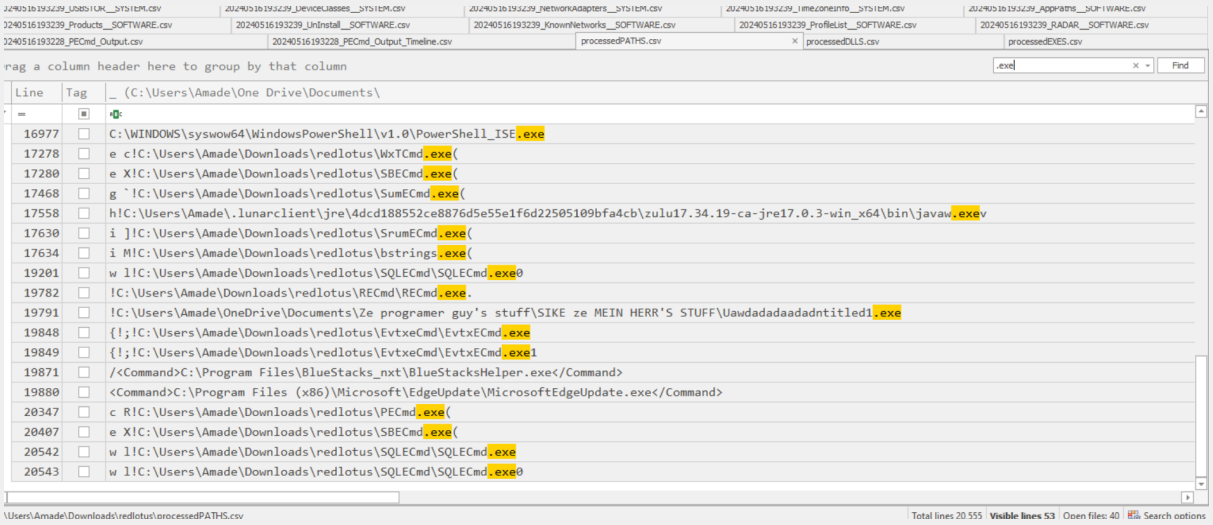
The TaskArguments.txt can also be put in Timeline Explorer (or opened normally), here users can check what files were loaded on system boot (or user logon); any cheat file found here can be instantly banned for. Cmd.exe itself, in system32 - has the potential to be considered bannable as well.

Example A:

SSer (A) ScreenShared a suspect (B) who kept boasting about being undetected. They have bypassed a lot of SSers and nobody can even find a file they used. The ScreenSharer found C:\Windows\System32\CMD.EXE in the scheduler service because they had read Red Lotus public methods and banned the Player for it.

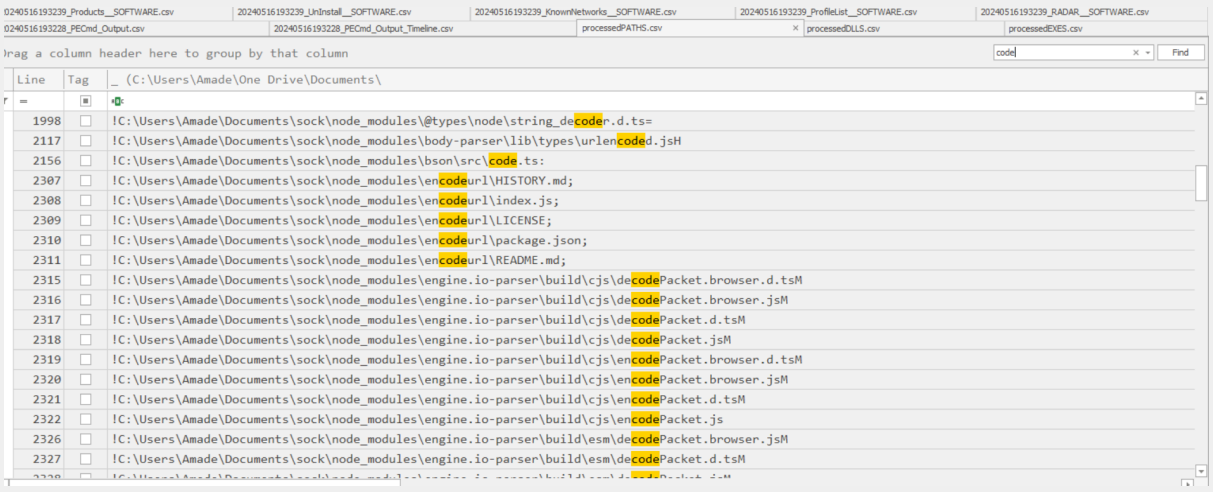
Was the ban justifiable?

The ban is unjustifiable, even though the SSer was dealing with a very annoying bypasser. Reason being, every user's device can have cmd.exe displayed in the scheduler process, but if you copy results or parse scheduled tasks (in c:\windows\system32\tasks) cmd.exe will not be displayed anywhere. Thus, every SSer should rely on extracted physical data and not services data read in system informer. That is the standard of Red Lotus.



ProcessedPATHS.csv logs pretty much every file or file path logged in temp (that's physically extractable), thus it is, alongside processed DLLS and EXES, a very good material to be further dumped (Hide Line and Tag columns before extraction) using Baca's Signature Checker!

They can also log coding environment data files. Sometimes.

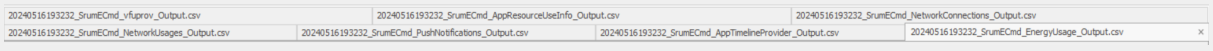


-Uptime of Apps

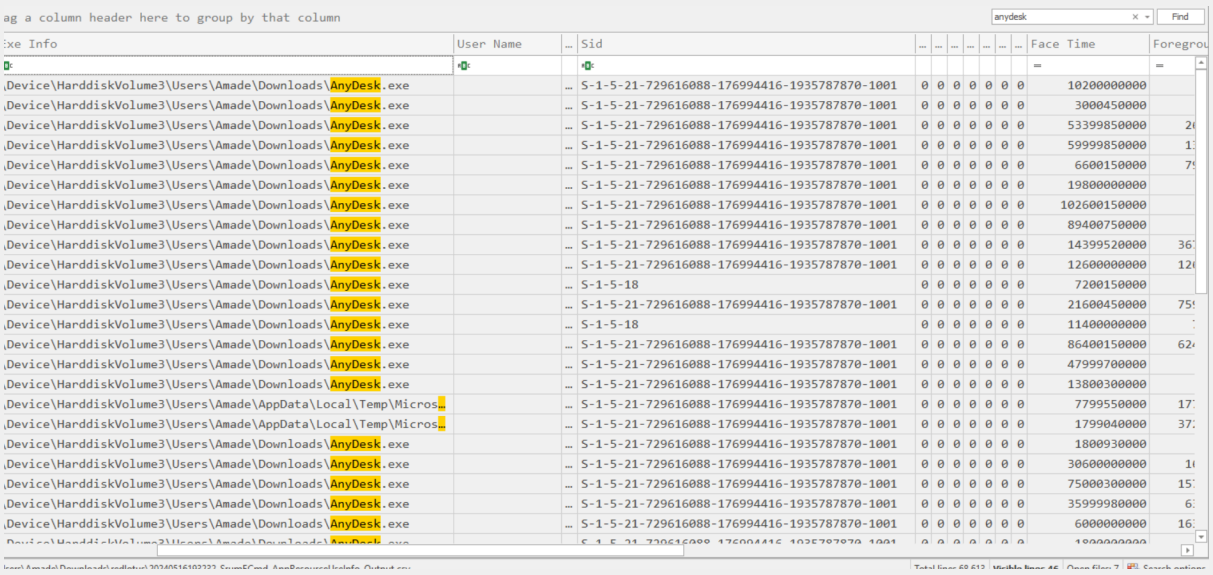
Sometimes bypassers do not know how to bypass certain services or traces. Mainly because they never do their research or buy "bypasses" from bypass providers and servers. SRUM database is one of them. So, what exactly is the SRUM database?

["SRUM is a database introduced in Windows 8 and later versions of the Windows operating system. It stores information about the usage of system resources, such as CPU, memory, and network utilization, and enables the examination of them by the applications running on the system1. SRUM is considered a gold mine of forensic information, as it contains all the activities that occur on a particular Windows system. SRUM tracks and records program executions, power consumption, network activities, and much more information that can be retrieved even if the source has been deleted"]

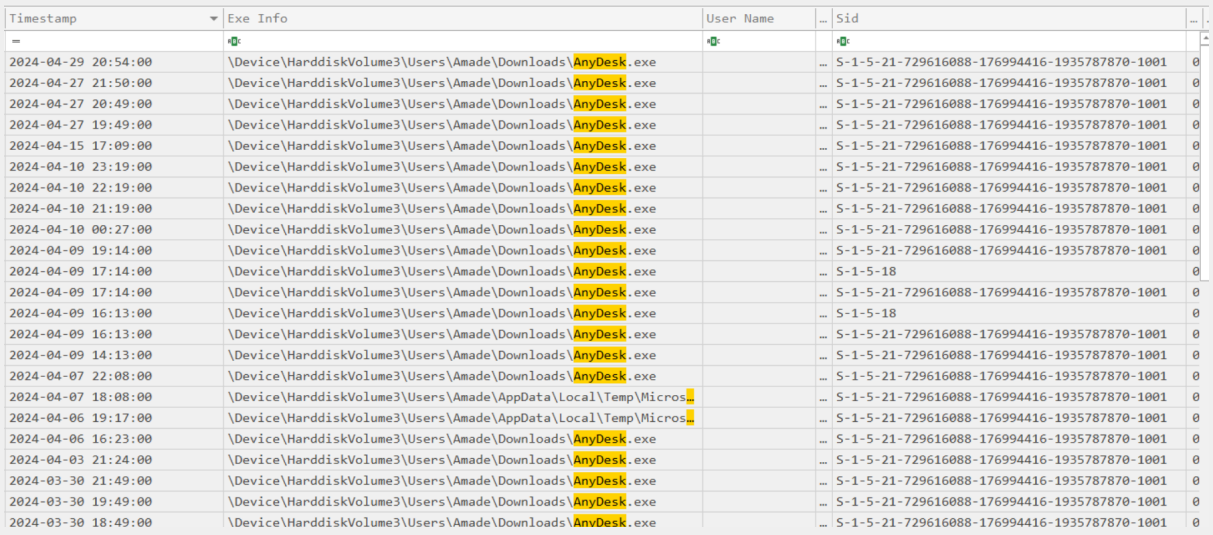
RL Collector, of course, collects this information as well. In fact, it collects a lot of it.



SrumCmd_appresourceinfo.csv shows the uptime of Applications, as well as how much resources it consumed, facetime, background time and so on.

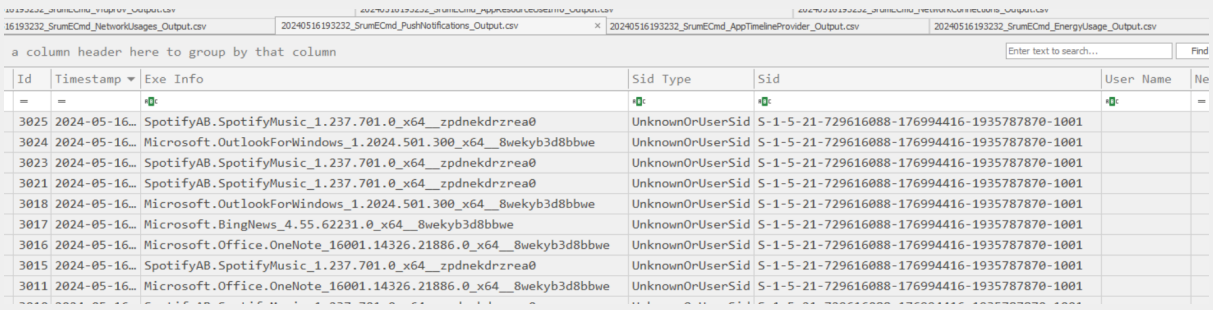


All the given data can be converted into readable format by copying the numbers of, for example, Foreground Time and referring to an epoch time converter. [Epoch Converter - Unix Timestamp Converter]; this data is often (and mainly) used as DPS strings (or alike DPS strings where SSers experiment bypasses and cheats & mark down the most persistent logs of them) - **the most important is a Timestamp.**



Essentially, this is yet another evidence of file execution and file resource usage data.

SrumECmd tends to be the most useful lately with cheats that use push notifications in order to avoid leaving traces elsewhere (or simply for the sake of aesthetics); SrumECmd_Pushnotifications_output.csv logs all of that information as shown down below:



It logs the timestamp of a push notification and its payload sizes as well (displayed all the way to the right) thus, push notification cheats are quite easy to detect with this (alongside with other evidence of execution since they're still physical files...)

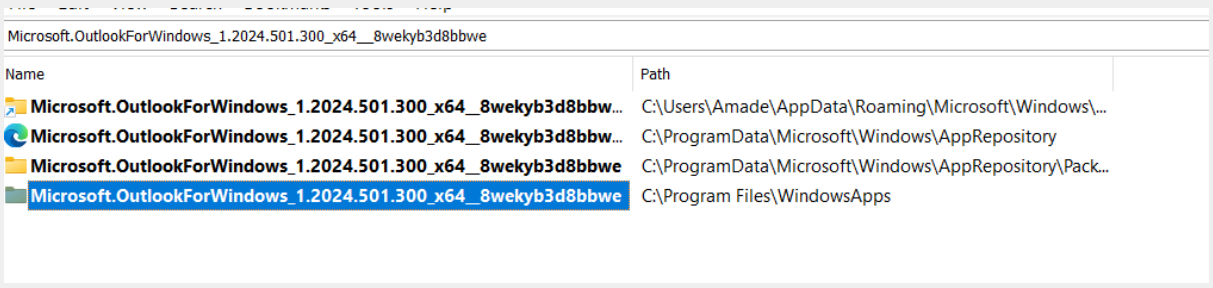
As you may have already noticed, push notifications are in the same format as info given in BAM. Except these ones also have architecture (x64) specifications and some weird numbers with them. Let's review a hypothetical scenario that'll help us better orient with this tiny bit of info:

Example A:

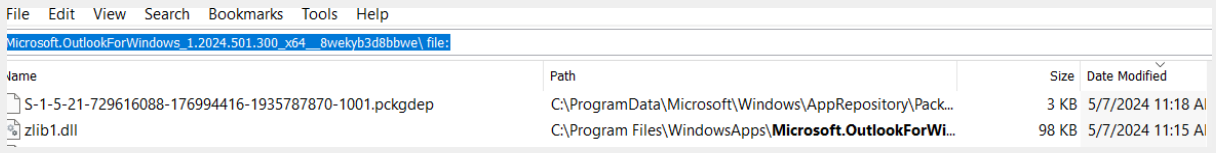
SSer (A) ScreenShared a Bypasser. In push notifications, they found "YourPhone" data with an x32 architecture type while everything else was in x64. The notification also was with a huge payload (over 9000). The SSer didn't mind much of it and continued to SS.

What could the SSer use to their advantage?

The SSer could have checked every other "YourPhone" push notification and compared the results, marking down potential dates the operation took place on. The SSer could also check the path provided. Let's take Outlook for example:



This would reveal a folder that holds all the associated information, such as dependencies, loaded files and so on. The SSer could also use this string in search everything to sort all of the given files, extract results as .txt and scan all of it with the BACA Signature checker: **Microsoft.OutlookForWindows_1.2024.501.300_x64__8wekyb3d8bbwe\ file:**



『Chapter VIII』

Hypothetical Scenarios

This Chapter will help you better theorize and visualize everything during and before ScreenSharing suspects. Let's review hypothetical scenarios with as few screenshots as possible.

Example A

While ScreenSharing a suspect, the SSer checked JumpLists and shell:recent while Echo was scanning, to not waste more time. Shell:recent was full of .mov (video) files (thus the user accessed them) and the default Videos directory accessed prior as well. The SSer didn't mind much of it and waited for Echo to finish scanning. Results were "Suspicious: Vape.gg accessed during system boot"

The ScreenSharer couldn't find anything and banned them for 2 months old vape prefetch having "Close" flag in the Journal before ScreenShare.

What should've the ScreenSharer done?

Any ScreenSharer who has met bypassers with renamed extensions would immediately speculate that the files they're viewing in shell:recent are potentially renamed extensions. This is a sign of good bypass understanding - the most harmless files seem suspicious as well, that's mainly why every advanced ScreenSharer is hyper paranoid when SSing and nobody can understand what they're doing. The ScreenSharer should've checked these .mov files - without opening them, of course. The thing is, when you zoom into the folder full of video files - actual video files have thumbnails whereas suspicious or corrupted video files display the default windows video icon - their size is also between 10~20MB (not always) while high quality videos have higher sizes (varies);

On the other hand, the "close" flag in Journal on a .pf trace is **insufficient evidence** to ban for. Even if JournalTool (of Echo) showed Data Modified and then Close - that **does not mean that the file was closed**; For further understanding, JournalTrace should be used and the .pf trace must be further evaluated to make sure attributes weren't modified.

And when it comes to Echo's vape.gg flag in current system boot, it is universally accepted that **nobody bans for cheat's websites visited** - the flag serves as an indication that something may be suspicious or that the user requires in-depth evaluation. Of course, methods explained in Chapter I of this handbook would be sufficient to catch a Vape user. However, Vape is a signed file - thus signature checks wouldn't really be of much use.

Example B

While ScreenSharing a suspect, the SSer checked Registry Explorer to analyze Comdlg & BAM entries. BAM showed a strange triangle warning Icon, but the suspect claimed they never touched regedit and have no idea what it means. On the other hand, comdlg32 had its .jpg values highlighted in red with purple detail highlights. The suspect also claims they have never touched comdlg32. The ScreenSharer doesn't believe them and bans for clearing comdlg32 and BAM traces as only users are able to delete registry values.

Is the ban justifiable?

Believe it or not, the ban is **unjustifiable** due to **insufficient evidence.** Let's break it down.

Any ScreenSharer who has actually read the Registry Explorer Manual (it comes with the download) will come across a page that talks about icon meanings. It is a screenshot that displays everything necessary to understand highlights and icons in Reg Explorer. In the given scenario, a triangular warning on BAM **does not automatically mean that something was cleared** in the given value - especially if everything else remained in the local user's hive (1001); in the given scenario the SSer should have checked associated and unassociated deleted values for \Harddiskvolume string that would reveal which exact values were deleted within BAM. On the other hand, the manual also shows a second icon - circled red warning icon. This warning icon would be enough to conclude that BAM was cleared in the current system boot (in instance) and not previously, days or months ago.

When it comes to ComDLG32 ban, it has no justification because if something got highlighted in red using Registry Explorer then detailed values still remain and display the deleted data, not to mention how they could've simply checked associated and unassociated deleted values as well. Besides, JumpLists exist. There are many alternatives to this detection and **knowing the extension is enough information to look for evidence of execution OF that specific extension everywhere.**

Although it is likely that the suspect actually cheated, by Red Lotus standards, they would be unbanned due to lack of evidence. However, if BAM displayed a red circled warning icon in this example - no matter what the user “claimed”, they would remain banned for clearing BAM entries during current system boot.

Example C

While ScreenSharing a suspect, the SSer skimmed through WinPrefetchView in a search of some newly released undetectable AutoClicker trace. The Bypasser tends to be in many bypass servers and there's a high chance of them actually owning a cheat. The ScreenSharer doesn't know that the AutoClicker is a .bat extension, the file also simply doesn't log in services other than Pcasvc and Explorer, but the ScreenSharer was told by other SSers that Explorer and Pcasvc are useless methods.

In the end, nothing was detected and the bypasser was let go free. Turns out they really owned a .bat cheat that they launched using a Powershell script to avoid leaving traces even in Pcasvc.

What should've the SSer done?

When it comes to new cheats that nobody knows the extensions to, Red Lotus has already developed numerous universal (general) methods for them. RL Collector would collect a lot of data that could've been used to detect the given bat file. After all, the SSer had a chance to detect it by analyzing Conhost / Consent further in WinPrefetch because .bat files keep logging in these two .pf traces. And not only - there are a few .bat files by default on Windows, thus using search everything to filter out **.bat content:"@echo off"** then analyze each and every .bat (or if the SSer had seen the GUI of the file they could type any word from that bat) - due to not a single default windows .bat having "echo off" in their contents, the detected .bats would most likely be this autoclicker. When it comes to proving its execution - always check pcasvc, explorer, searchindexer and what not. Never listen to ScreenSharers who don't utilize every possible approach to their advantage.

The ScreenSharer could've also used the Powershell script detection methods mentioned in RL Handbook Tome II (PUBLIC) version.

Example D

While ScreenSharing a suspect, the SSer knew of a WinWord bypass being used, thus they decided to analyze WinPrefetch for every MSWord / Word entry. Upon inspection, they found nothing suspicious that was loaded into the .pf trace so they decided to scan with Echo and Ocean. Both tools flagged Karma on the winword.exe file in the current instance. The ScreenSharer banned them based on SS Tool flags.

Turns out the suspect was on a fresh Windows installation and SS Tools false flagged.

What should've the SSer done?

WinWord (VBA) bypass relies on developer mode Word (or Excel) to actually launch applications or codes as a vba. Usually, it doesn't actually leave traces of the execution within its prefetch itself (unless it runs the file);

This bypass also relies on the user actually accessing a word file with the developer mode toggled on. Accessing Macro settings and modifying them by hooking a file / different word script to the given word file. Sometimes, vbas are indeed .vba files and they log in Prefetch. It's basically quite a complicated thing to prove properly.

Despite all of that, the ScreenSharer **false banned due to lack of evidence and care.** They should've checked loaded word files in the prefetch trace (or even shell:recent) to determine whether or not it has the developer mode toggled on. If it's toggled off, the SSer should then realize that **SS Tools flagging winword.exe as Karma doesn't warrant a bannable offense - or any other cheat. It must be double checked and investigated further.** In this case, if the SS Tools flagged .pf traces of WinWord as Karma, then it'd be more likely that Karma was actually launched using WinWord, because it's the structure of its .pf that was modified / changed / abnormal! Thus, in this case, again, the ScreenSharer should check a word file, go to options, macros and see if it corresponds or is linked to another word file (the developer mode must be turned on), in which case **purely finding this modification** would be enough to ban for accessing scripting environments!

Checking WinPrefetchView can be done using methods explained in Chapter 1 or by a naked eye to check what files were loaded and to double check them later on.

If the user forgot to delete their script, here's what SSers should look for:

```
Sub AutoOpen()

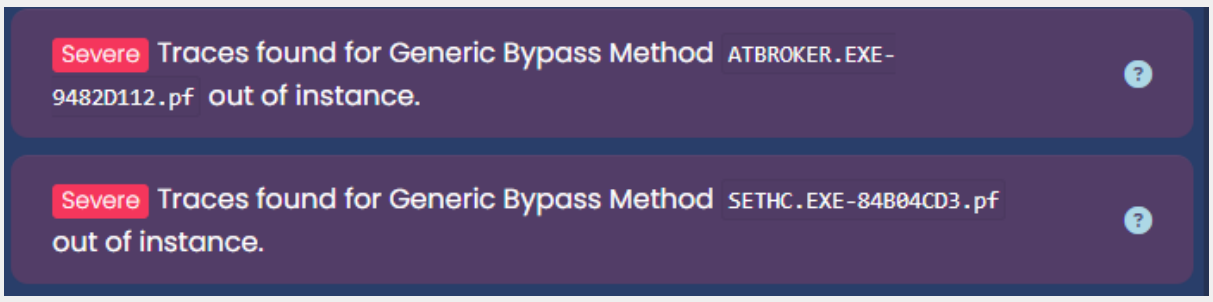
Set objShell = CreateObject("Shell.Application")
objShell.ShellExecute "calc.exe", "", "", "open", 1

End Sub
```

That launches calc.exe. ScreenSharers could also look for .vba files on the device individually.

Example E

After scanning the Player with Echo, the ScreenSharer was met with unfamiliar flags, which was:



Community members told him that the flag represents a .pf deletion flag or a "type" bypass (replacing one .pf's data with the other) that the Dream Client often uses (although the SSer is from FiveM and doesn't understand it.) Thus, the ScreenSharer banned a suspect under the reason "Bypass Attempt - In instance"

What should've the SSer done?

In the given scenario, the ScreenSharer issued a false ban due to a false flag in Echo. With current Red Lotus "Quality Control", every SS Tool (Echo, Ocean, Avenge, Storm and so on) must be double checked manually. In practice, these echo flags represent numerous things, let's break it down:

1) Deleted Prefetch

It is unlikely that this flag stands for Deleted Prefetch. Why? because cleared or deleted prefetch logs under **Generic Bypass Method** without specifying the file itself. As we read in Echo's flag announcement a while ago.

2) Overwriting Prefetch (type bypass)

It is more likely that the given flag corresponds to type overwrite of the .pf trace. Using the command type pf1.pf >> pf2.pf where bypassers specify a legit then cheat file to be overwritten, will cause this flag to be displayed and both .pf traces will also be shown. But what happens when there aren't just two but one or more than two .pf flags?

3) Bulk data overwriting Prefetch

It is rare but likely that the bypasser used some sort of a software to modify the contents of each .pf flagged.

4) Corrupted Prefetch

In practice, this flag is linked to corrupted prefetch traces that have nothing in their contents. So, data of that PF trace is nullified. SSers (including ones from Red Lotus) argued whether or not it's a bannable offense regardless of the potential of it being a non-user initiated action. For now, I'd say it's not bannable since we have the name of the file and we can check Journal on that file name. Other RL methods will still catch files executed but hidden this way. However,

reading what happened to the .pf using JournalTrace would draw a clearer image and hint on what exactly to do or what really happened.

Thus, what to do? look further. Use the win prefetch view to check the trace.

Example F

When ScreenSharing the user, an SSer used EVTX logs collected using the Zimmerman tool (or used RL Collector to automate the collection); they begun by analyzing time change data and extracted the following:

Map Description **User Name** **Remote Host** **Payload Data1** **Payload Data2** **Payload Data3**

The system time was changed *OldTime:* ***2024-04-15 20:15:51.6857370*** *NewTime:* ***2024-04-15 20:15:51.6857470*** *Reason: An application or system component changed the time.*

They kept it as a part of evidence but continued to ScreenShare. Later on the SSer also found a Microsoft-Windows-Powershell provider with a weird to him script (here's a bit of it:)

Time Created

2024-02-27 05:38:39

ScriptBlockText: $paths = Get-Content -Path "paths.txt", $strings = @("This program cannot be run in DOS mode","OpenProcess","88i8s8.8B8a8c8k8g8r8o8u8n8d8l6m-$Te7","Mhttp://crl4.digicert.com/DigiCertTrustedG4CodeSigningRSA4096SHA3842021CA1.crl0=","eb:DARA","MODE CON COLS=100 LINES=30","h.UPX1","H_RDATA","H.UPX0","\gIq6","me/dewgs/clicker/ClickerFrame$1.class","fF66H6n","IsDebuggerPresent","themida","@.themida","Inject","Injecting","[+]","memory address","D3D10CreateDeviceAndSwapChain","UPX",".UPX2","Click".

Then ScreenSharer decided to ban the user for Changing System Time and Bypass Attempt for running suspicious scripts.

Is the ban justifiable?

System time changes are very inconsistent windows behavior. As we see in text, it updated by 0.0000100 seconds and the event log was generated! and in the given scenario, the Powershell script would warn as a "Warning" (not malicious-pwsh) in event logs because **the script reads file data.** It is a YARA-like string analyzer. The ScreenSharer should've paid more attention to the time change data and **re-run** the powershell script to ensure it was a malicious script, **but even if it was a malicious script, notice how the date is 2 months before ScreenShare,** thus this is a poor level ScreenSharing and the ban is undoubtedly false.

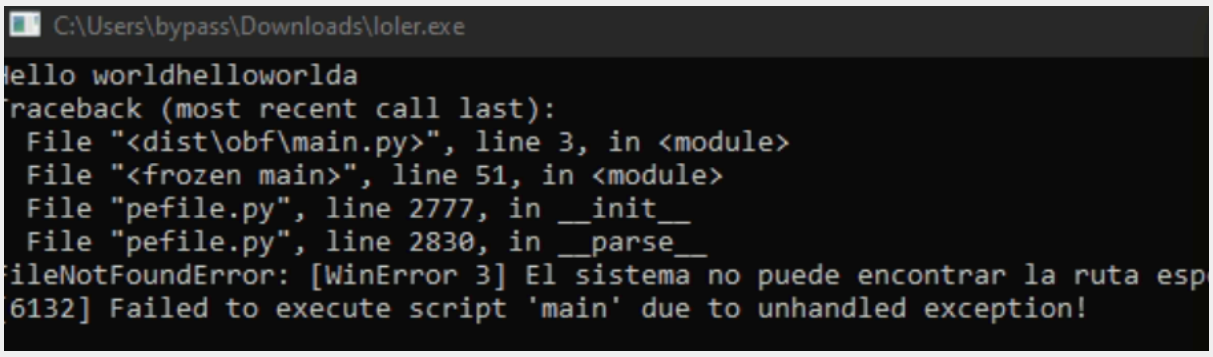
Example G

When ScreenSharing the user, an SSer decided to use Registry Explorer and check bookmarks for pre-made destinations. He wanted to prove the date of execution of Karma that he found by extracting all .exes from Search Everything and scanning them using the Baca Signature Checker. Near the end of ScreenShare, he found a weird CLI based file that auto-closed every time he launched with an explorer icon, displaying momentarily upon launch. The ScreenSharer didn't think much of it and let the user go.

What should've the SSer done?

In the given scenario, there is no doubt that the ScreenSharer should have used CMD to launch the application - because that way he'd be able to see errors and the reason for it closing. This is a behavior often seen in CMD based applications or process hollow utilities. Basically, every CLI based application (CLI means Command-Line) for example, bstrings as it doesn't have a GUI.

Next, the ScreenSharer could've also uploaded the file to virustotal to analyze its behavior, registry keys and values it modified / interacted with and so on. So, what'd happen if it returned a weird result like this?



PE File requirement means that the run tool **requested to provide a portable executable to execute.** Thus, it is a process hollowing tool. Process Hollowing utilities are still files that leave traces on the device. **Finding a process hollowing tool is enough reason to ban the user for, of course, in instance (**or whatever your server rules are)

Using every method in Chapter I and publications of Red Lotus is sufficient. Same thing goes for files that don't run because AnyDesk is on - uploading them to Virustotal will return enough data that can conclude the ScreenShare. Or, simply asking the suspect to run it after closing AnyDesk when the SSer joins via Discord call should be enough.

Additional Material

Video Guides

https://www.youtube.com/watch?v=B4SmfKkazXo
https://www.youtube.com/watch?v=xAE7fSGFcXM
https://youtube.com/playlist?list=PLb5pE5KTRzmaX_ZRjNF9nZUcbwmUYqj0R&si=qDOGpljbns7KzInZ
https://youtu.be/fTsTszeXB28
https://youtu.be/OQx-uLih9-o
https://youtu.be/Z9Oe_P7-Wm8
https://github.com/lattiahirvio/SSTool/tree/main
https://youtu.be/1FnZPq1OYzM

Readable Material

https://github.com/mesquidar/ForensicsTools#live-forensics
https://academy.cyber5w.com/courses/timeline-analysis
https://learn.microsoft.com/en-us/windows/win32/fileio/file-attribute-constants
https://dfirmadness.com/case-001-autoruns-analysis/ https://dfirmadness.com/getting-into-infosec/linux-macos/
https://learn.microsoft.com/en-us/sysinternals/downloads/autoruns
https://dfirmadness.com/investigating-shellcode-alert/

Tools

http://launcher.nirsoft.net/
https://shadowexplorer.com/downloads.html#google_vignette
Find and recover deleted files and previous versions of existing files inside Windows shadow copies

Other Red Lotus publications

Red Lotus Library

『Red Lotus』

Red Lotus Unity

Foundational Regulations

This shall establish legal standards for

The Red Lotus Headquarter

Privileged individuals

Chapter 1 - Granting Rights

1. <u>Granting rights</u>

1) Right to sell **Lifetime (until the end of Red Lotus itself)** services up to the Course of Red Lotus coaching is granted to Headquarters from the previous Owner of Red Lotus.
2) Headquarters still does not maintain the right to sell Extended course (Lotus+) to third parties without notifying the previous Owner first, taking permissions, or depositing income into the previous Owner's account.
3) Red Lotus services fall under Contractual and Intellectual Property Laws. Intentionally or negligently, grantees shall not leave room for Criminal errors.
4) Only four Owners are set by the previous Owner. Newly set Ownership shall not be removed.
5) The HQ is granted the right to accept new Delegates.

2. <u>Delegating rights</u>

1) Tasks, concerning proportional to regulations responsibilities are entrusted (delegated) to the Delegation team of Red Lotus.
2) Delegated tasks are tasks that a staff team is entrusted with. It is a custom[10].
3) Exclusive delegated tasks (tasks, that exclusively spread in Red Lotus; Red Lotus only) include:
    a) Handling tickets
    b) Notifying the Headquarters of purchase inquiries
    c) Background checking clients who want to make a purchase
    d) Scheduling and Hosting webinars
    e) Assisting partner trainees
    f) Opening branch markets

3. <u>Branch markets</u>

1) Branch markets are services proposed by the Delegates to the community below or equal the price of RL Academy.
2) Branch markets shall be relevant to ScreenSharing and guaranteed by the Headquarters in form of Onboarding, Forums, Channels, Unity Plaza announcements.

Chapter 2 - Branch Markets

4. Depositing branch market's income

1) Creditors (sellers) shall only deposit income in their personal PayPal, Tebex or other internationally verified stores. Debtors (buyers) may only be required pay in US Dollars.
2) Deposited total income shall not exceed 500 US Dollars. Every exceeding income shall be deposited to active Owners.
3) Regional fees must be ignored, debtors shall not be required to pay the exact established price that disregard fees.

5. Branch market creditors

1) Branch market sellers must be members of the Delegation team or partner server staff teams.
2) Branch market sellers must be genuine with their offers and services.
3) Branch market creditors are not Course members of Red Lotus. Therefore, they are disallowed to open branch markets.
4) Partner server staff members (partner server creditors) who decide to open a branch market in Red Lotus may do so after paying a one-time fee of 5$ to one member of the Ownership.
5) Partner server creditors may not advertise external services unrelated to ScreenSharing.

6. Branch market sectors

1) Branch market sectors that include programs and tools must also include their source codes linked under the sector.
2) Branch market sectors that include free content must not be turned into paid content in the future.

7. Disbanding branch markets

1) Branch markets and their sectors will be disbanded by the Ownership, if:
    a) It/they sell(s) bypass(es);
    b) It/they advertise(s) external services;
    c) It/they sell(s) free utilities or methods (governed by Intellectual property Laws);
    d) It/they sell(s) public content (governed by Copyright Laws);
    e) It/they sell(s) digital forensic articles (governed by Copyright Laws);
    f) It/they sell(s) malicious content with malicious intent (governed by Criminal Laws);
    g) It/they provide(s) no source code(s) of softwares that may also not be malicious;
    h) It/they is/are not subject to genuinely fulfilled obligation(s)
    i) It/they is/are unrelated to ScreenSharing
2) Disbanding branch markets that violate sec.1 of this article is an obligation of the Headquarters. Not subject to refunds.

8. General deposit limits

1) The Ownership or Branch market creditors shall not exceed $500 in income, exceeded funds shall be used to support the server by:
    a) Hosting giveaways
    b) Purchasing partner ScreenShare Tool enterprises
    c) Supporting partners
    d) Funding motivated ScreenShare teams of partner servers.

Chapter 3 - RedLotus Merchanting

9. Individual Merchanting of the Headquarters

1) Current Headquarters, those assigned by the previous Owner and not newly assigned by the active HQ, reserve the right to deposit individual trade(s) (by 100% income) of Red Lotus Course into their accounts after ensuring the Customers are completely aware of the services and agree to the contract.
2) Every HQ member is subject to rights in section 1 of this article.
3) Purchase transcript must be sent to the Customer.
4) Fees that the HQ received through Branch Markets must be spent on giveaways.
5) The HQ can set Monthly fees or charge $5 each time the Branch Market makes a purchase.

10. Solidary Merchanting of the Headquarters

1) The main 4 HQ members are Solidary Merchants of Red Lotus services defined in Chapter 1 of these regulations.
2) Lead HQ members are obligated to deposit 60% income by trading the Course if:
    a) The total income of Red Lotus has reached $500 (the sales have reached 13 Individual buyers); every next (14th, 15th, 16th, etc) income is a must to be deposited to the previous Owner by 60% of received quantity (this includes invoices received with fees), 40% of which shall be used to support the server via hosting giveaways (in any form) or depositing it to Partner server's Stores or Ownership to pay their ScreenSharers, or guaranteed by the HQ (only #1 will receive the exceeding funds from the HQ without depositing it to the partner Owners)
    b) There are more than one new purchases within the same period. Same Period is the time span of active purchase inquiries, where one buyer has opened a ticket but has still not paid, during which a 2nd inquiry is made of a Customer who also wants to pay. In this case, either one of Lead HQ members shall be called to both tickets, they must receive the payment of both buyers and only then deposit 60% from Total;
    c) Exclusively, if a server or a community decides to purchase a Course Packet (for over 5 members at once), they receive a 10% discount on their purchase. This income will not count into the $500 limit of the server and the previous Owner shall receive 60% invoice of the Total. Remaining 40% stays with whomever from the HQ has managed the packet inquiry.

11. <u>Deposits under the exclusive contract of P.A (Partnership Agreement)</u>

1) If the Buyer is unable to pay via PayPal to individual Headquarters, they may purchase relevant price worth ($40 for Course, $20 for Academy) product specifically and only from Ranked Bedwars Store at Ranked Bedwars Webstore | UNBANS (tebex.io) - For Course, they shall buy **Unban [Permanent]** and the Ownership of RBW must be given payment credentials and confirmation of the Purchase. The Buyer must join the RBW Discord server (discord.gg/rankedbedwars) to validify purchases.
2) Following section 6 of this Article, the Ownership shall also be asked to deposit $20 from that specific Course purchase into SS Top balance.
3) Following section 6 of this Article, if the Academy purchase is made, the Ownership shall also be asked to deposit $10 from the Academy purchase into SS Top balance.
4) The remaining part of that specific Course or Academy purchase shall be used for the Partner, they are given freedom of choice.
5) This contract is exclusive between Red Lotus and Ranked Bedwars.
6) The Ownership must be notified before the Purchase is made with all the details regarding the inquiry.

Chapter 4 - RedLotus Packets

12. <u>Lotus Course $40 *(Standard - Lifetime)*</u>

1) Red Lotus Course (Standard) is a service of Red Lotus that each Headquarters member can officially sell following the regulations of this Document. **The Cost is $40**
2) Lotus Course (Standard) Packet includes:
    a) Live-SS Support; Customers can have the HQ assist them during their ScreenShares, answering questions, providing Tools or continuing the ScreenShare for them. The quality of the Manual ScreenShare will be based on whichever HQ Member conducts it. It may not be a guaranteed determiner of the Player cheating, or not cheating.
    b) Premium Manual methods that orient on newly released bypasses and their detections - if the Bypass is sold for below $40 on the market. This combines every $200 worth bypass on the Market. Currently, there are detections of clients and bypass methods sold for over $200 in total.
    c) Manual ScreenShare methodologies and approaches to Bypass methods or questions asked by the Customer after the Purchase.
    d) Individualized Coaching and a systematic list of what the Customer's weaknesses are, suggesting approaches for improvement.
    e) Invitation to Red Lotus Academy. The Academy includes hosted online Webinars where the HQ or Delegation or invited guests will hold live coaching tutorials, answer questions from the Course members and demonstrate detections Live.
    f) Guidance of Linux and MacOS ScreenSharing.
    g) As a Bonus, Customers receive premium Red Lotus scanners and Tools. This is not included in the price as the Course and Lotus+ are both for Manual (by hand) ScreenSharing and understanding the procedure.
    h) Periodic one-on-one Coaching either through recorded Videos, written Guides and Tests, or live calls.
    i) Individualized ScreenShare Tests through Documents, Text tests or by ScreenSharing the Coach's device, where they set-up known bypasses and tests.
    j) Course members will receive a "Certificate" after the Headquarters issues a finalizing Test of their skillset. This Certificate is aimed to give the Course member opportunities in Partner servers of Red Lotus, instant acceptance in their ScreenShare Teams.
    k) This purchase will last for a Lifetime, meaning until discontinuation of Red Lotus.

13. <u>Lotus Academy $20 *(Lifetime)*</u>

1) Red Lotus Academy is a purchase that allows buyers to attend Webinars hosted by the HQ or Delegation specifically dedicated to the Academy. This does not include themes for the Course.
2) The Academy includes abilities to:
    a) Attend extended hour Webinars on specific leaked Bypass detections.
    b) Receive an Intermediate Certificate after finalizing Coach's guidance that can also be used to apply in Partner servers.
    c) Academy will receive Boosters methods and permissions for a Lifetime.

14. <u>Lotus+ $70 (*Extended - Lifetime*)</u>

1) Only the Founder Owner of Red Lotus is allowed to sell this service individually; The Founder Owner is *UnMonsieur* ***this is a separate purchase and can not be bought with discounts after purchasing Course or the Academy;***
2) This is the highest Manual coaching service RedLotus Offers. Also known as "White Lotus", This extended Course includes:
    a) Everything from the Course and Academy;
    b) Written Guides for Beginners on Windows, MacOS, Linux;
    c) Written Guides for the Advanced and above - Windows;
    d) More concentration from the HQ and Delegation - this means more occasional SS Tests, Tutorials, sending recorded mock ScreenShare videos, extended one-on-one coaching and more.
    e) The Extended Course teaches how to gather information from Forensic artifacts, understand their usage specifically for ScreenSharing. Offers custom Forensic utilities that automate collection procedures.
    f) Improved imitations of YARA, custom hunt rules, faster and more effective Manual SS Methods, approaches, utilizations of Tools.
    g) Private recorded ScreenShare demonstrations of more complex ScreenShares.
    h) Detections of bypasses exclusively released recently, are up to date, harder to detect. This includes bypasses and clients that bypass SS Tools.
    i) In Depth evaluation of ScreenShares done by the Customer.
    j) Allowance to remain or apply for Bypass servers without receiving mass server bans - only if the HQ is made aware of the reasoning.
    k) Detections of Anti-Forensic methods yet not used within communities to develop better understanding of systems.
    l) Teaching bannable and non-bannable situations, how the system reacts to certain actions and more.
    m) Extended oriented coaching to develop "ScreenShare sense"
    n) Any support related to ScreenSharing.
    o) Ability to request a specific Downloader to be coded for the Customer.

p) Ability to request a specific scanner, (that the HQ is able / knowledgeable to code) that does not infringe user Privacy or does not violate Red Lotus SS Unity Charter, to be made.

3) Utilities developed by the HQ mentioned in Section 2 of this Article will be as good as the HQ is capable of making them.
4) This purchase is a Lifetime purchase. Servers are not allowed to buy this service collectively
5) In absence of the Founder Owner, nobody is allowed to sell this service. Customers are made aware of the absence of the Founder Owner, only Lotus+ customers themselves and the Lead HQ Owners assigned by the Founder Owner replace these services without losing quality.
6) 117 pages worth of Handbook - Red Lotus SS Handbook Tome II (Introductory, Book I and Book II)

14) Enterprise

1) Servers who decide to purchase an Enterprise are limited to buying the Standard Course only.
2) Enterprise Owners will receive a 10% discount from their Total payment.

April 21st, y2024

RedLotus Document - Approved by UnMonsieur[1]

The new dual Ownership is valid until January first, 2025.





- UnMonsieur | RL Headquarters

RL HeadQuarter

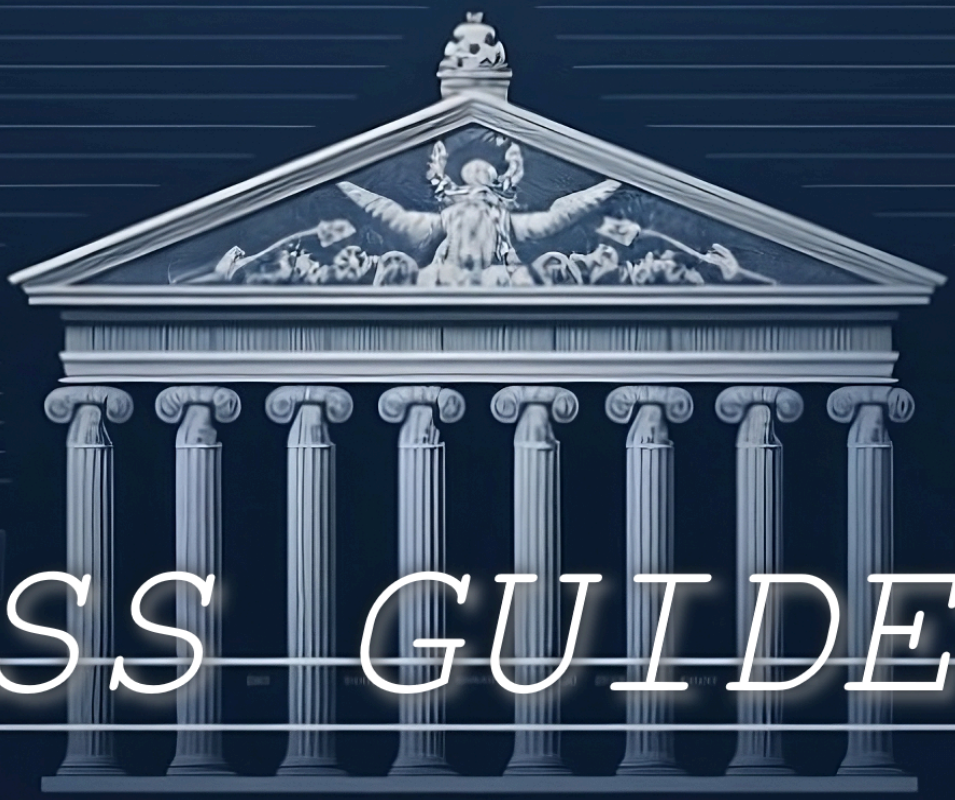
03

Screenshare Handbook - Tome II

118 original pages / reproduced in full

[FULL] [RL] SS Handbook Tome II (5).pdf

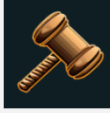
Original source content begins on the next page. Author credits and illustrations are preserved.



SS GUIDE

Tome II

Author: UnMonsieur | [RL HQ]





Governing Principles

This Guide is established upon "norms and regulations for the selection and proper implementation of ScreenShare teams within servers, communities, individual teams";
suggested for no imperative reason for implementation by Red Lotus Unity.
It is deemed by this Unity that the security and safety of players is more important than the accuracy and speed of ScreenShares, whether advantages are detected or not.
It is deemed by this Unity that the importance of data security is imperative over anything else that may favor Servers, Communities, Individual SS Teams.
It is condemned by this Unity to disestablish institutions regulating fair-play and privacy measures ScreenShare Teams are guided by, within communities.
It is deemed by this Unity that the experience and specialization of so-called "ScreenSharers" (other: PC Checker, Account Reviewer. . .) shall be tested and approved by positioned individuals responsible for functioning of the Team (SS Management, Administration, Leadership, e.t.c).
It is deemed by this Unity that every server implementing or copying this document, regulations, official Red Lotus regulative publications and suggestions, is responsible for building a system in complete accordance with the Unity of Red Lotus, governed by the Red Lotus Charter and Red Lotus Charter of Establishment.

~ UnMonsieur | Headquarters of Red Lotus

Anti-Corruption Measures

Measures and Regulations in this Handbook shall be read, understood and implemented by SS Managers, Administration or Ownership of recipient Servers, Communities, Individual SS Teams. Learners not responsible for regulating their SS Teams may begin reading from Chapter I.

Article 1 - Types of Ownership

Ownership is divided in two categories by the Unity of Red Lotus:

- Server Ownership
- Community Ownership

Article 2 - Types of Leadership

Leadership is divided in two categories by the Unity of Red Lotus:

- Individual Leadership of a Team
- Unified Leadership of a Team

Article 3 - Ownership and Leadership

(1) Server Owners are founders of an in-game Server with a specific purpose (Server of "Roleplays", Server of "Minigames", Server of "Factions" e.t.c)
(2) Community Owners are founders of Community Servers with no in-game purposes.
(3) Individual Leaders are the Head members of Unified SS Teams or Groups.
(4) Unified Leaders are SS Groups with no individual Leadership, every member is considered a "Leader", or, Unified Leaders are SS Groups consisting of Individual Leaders Uniting across communities.

Article 4 - Owner's Pact

(5) Server Owners must ensure that the new regulations will be implemented and controlled.
(6) Server Owners should take it only upon themselves to watch after SS Management (Owner's Pact), monitor whether general staff members, even second in charge, disturb their work or influence Management on appeals, applications, ongoing ScreenShares, ScreenShare Tickets or related.
(7) Server Owners may regulate this Article constituted by the possibility of its enaction, establish periodical lustration systems or related.
(8) Community Owners

Article 5 - Owner's Seclusion

(9) Server Owners must be secluded from the decisions on bans, regulations and projects initiated by SS Management (Owner's Seclusion)
(10) Secluded Owners still maintain the responsibility under the Owner's Pact.
(11) Owners are not required to know ScreenSharing to have a say, but must not influence ScreenSharers or ScreenShare Management.

Article 6 - Leader's Pact

(12) Leaders select highly skilled individuals, whilst being the strongest one of them, upholding secure and mature standards and moral stance, understand intruding in other people's devices is not a foolhardy responsibility and maintain traits respecting the practice of ScreenSharing while showing the community there are safety measures installed within advanced communities.

(13) Individual Leaders have the privilege to cooperate with Red Lotus, for their showcase in the Unity of Red Lotus, implement their ScreenSharers in partner servers of Red Lotus.

Article 7 - Unified Leaders

(14) Red Lotus is an example of Unified Leadership, the Headquarters and Delegation representing members and cross-server leaders / owners of ScreenShare related projects & communities.

(15) A Unity of Leaders, if any established in the future, will be perceived by Red Lotus as their ally, it is a responsibility of the Unified Leadership to work on the improvement of ScreenSharing, honesty and security within communities and ScreenSharers.

Article 8 - Anti-Corruption

(16) Anti-Corruption measures defined in this charter refine pre-installed regulations governing any server, community, team implementing these regulations to prevent false punishments, bribing, planting evidence and every other possible act that can be done against the genuinity and accuracy of ScreenShare results and procedures.

(17) Any amount of bribery must be disallowed. Any benefit received from non honorary beneficials must be split amongst teams. A guarantee's role is put upon Leaders, Managers, Owners.

(18) If evidence is insufficient or raises questions, Management and Leadership must not believe the ScreenSharer's explanation.

(19) Evidence gathered against suspects must be shared with them upon reasonable requests. Reasonability of a request must be judged accordingly to their behavior and purpose of sharing.

Introduction

If you are new to ScreenSharing (after: SS) or have just returned from so-called “retirement” as an ex “ScreenSharer” (after: SSer), it is important to be familiar with recent news and updates within ScreenShare Communities and “modernized” practices of the hobby.
The Red Lotus SS Unity is a project first initiated in 2023, March 3rd. Ever since that day it has invited servers, communities, hobbyist SSers and enthusiasts to establish the strongest and most high-quality unity that SS communities have ever seen before. The official ScreenShare Community was discontinued on December 12, 2023 and Red Lotus SS Unity was announced as today’s best SS Community individuals can learn from.
Currently, Unity is associated with around 13 other communities.



Instant Links

RedLotusBam.ps1
ExecutedProgramsList
LastActivityView
Eric Zimmerman's tools
USB Drive Log
AccessData_FTK_Imager_4.7.1.exe
Easy Journal Viewer
System Informer
Search Everything

Video Guides

How to use Prefetch in a ScreenShare | Red Lotus
Expanded Tutorial: https://www.youtube.com/watch?v=B4SmfKkazXo
Beginner's Video Guide (Old)
ExecutedProgramList
B.A.M
System Informer Filters
Prefetch, BAM, PCA
https://youtu.be/fw0fpqYRhiU

Chapter I

Basics of ScreenSharing

Test this yourself on your device: Download an executable file (.exe, for example, AnyDesk itself) and execute it, then close it. Right click on the file and go to its properties - you'll notice how "Date Modified" shows the date of the file's download time and "Date Accessed" shows when you double clicked on it.
At first glance, you have just proven both the download and execution dates of the software, had it been a cheat - this would be your first ban of a cheater. It is logical to ask yourself: if ScreenSharing is this simple, why have I heard of people getting bypassed?
Short answer: Because it isn't that easy.
Long answer: To perfectly showcase the significance of a ScreenSharer's understanding, let's repeat the same aforementioned process but change it a little bit;
If you have Windows or a Custom Anti-Virus, initiate it and wait for it to finish scanning the entire device. After it's done scanning, right click AnyDesk and go to properties again. Or even better, scoop through your Documents / .txt or other executable files, right click and check their properties. Chances are, all or some of these files will have "Date Accessed" showing the current time, even when you haven't executed the file at all! Why does that happen? Why is this relevant to know? Can you start explaining how to find vape already please?!
Basically, there is a difference between "Accessing" a file and "Executing" the file. While "Date Accessed" on individual files updates any time they are moved, selected, scanned by the antivirus, downloaded, edited or modified (that also updates "Date Modified") renamed, interacted with, sometimes even when the directory is Accessed (especially, with unsigned files or suspicious files that instantly get scanned by the AntiVirus every time you enter the directory / folder they're in - that's why sometimes you get alerted of a virus only when you see them in a folder as the antivirus tends to "forget" threats);
Therefore, if we want to prove execution of a file "in instance" (current system boot, current date / with Minecraft) or even "out of instance" (before Minecraft, hours, days early, months ago, you'll meet these two flags in Echo), we will refer to "Evidence of Execution Artifacts" (in short, execution artifacts or just artifacts) to prove an execution!
Windows is full of artifacts that log executions, evidently - file properties are not evidence of execution, besides "Date Modified" as it only updates with user interactions or by using external (separate, outer) applications to run them. Running a

file and executing the file - same thing, though you will often see SSers using these terms differently because of their experience with Bypasses and, I guess, psychology behind them. SSers often only use “Executed” when they refer to manually (by hand) executed files by double clicking on them, that is why they mention things like “execution logged in PCASvc” - that is because PCASVC is a service that logs executions and can be bypassed if the file is ran differently. “Running” the file is often associated with launching the file through CMD, for example (running the file through CMD (Command Prompt) as Admin bypasses PCA and there is no “execution” trace!) but in the end, both terminologies are the same.
Let’s begin explaining evidence of execution.

Evidences of Execution

Prefetch

Hold WIN Button + R at the same time on your keyboard, type “prefetch” (without quotes) in the RUN window. Press Continue on any warnings. Notice how every executable file you’ve executed logs there with exact execution dates (“Date Modified”);
Your AnyDesk should be one of them, this way we have solid and undeniable proof that the file was actually executed. Since Red Lotus Unity has produced two complete tutorials about Prefetch, we won’t expand it here, instead, watch them!



How to use Prefetch in a ScreenShare | Red Lotus
Expanded Tutorial: https://www.youtube.com/watch?v=B4SmfKkazXo

B.A.M

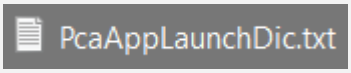
Background Activity Moderator is a registry component that logs executions. Very often, BAM shows executions that have bypassed Prefetch or PCA.
powershell Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass && powershell Invoke-Expression (Invoke-RestMethod https://raw.githubusercontent.com/PureIntent/ScreenShare/main/RedLotusBam.ps1)
To use the script above, open CMD as Admin and paste it there.



Video Demonstration

AppCompatPCA

This component is exclusive to Windows 11 ONLY (although, we’ll talk about how to check the same thing on Win10 as well), one of the strongest SSers (now retired) Rancio has discovered this Windows Evidence of Execution Artifact back when Windows 11 first got released.
This artifact is located in *C:\Windows\appcompat\pca*

Check 

ExecutedProgramList

Even though WinPrefetchView is practically better, other Nirsoft tools like ExecutedProgramList and LastActivityViewer also “parse” (modern SSers adapted Digital Forensics terminologies thus Parse is a synonym to “analyzes”) registry values that prove a file was executed.
ExecutedProgramsList
Video Guide

LastActivityViewer

L.A.V is one of the oldest tools in the book. The application parses both Prefetch and more Registry evidence of Execution that can be used to prove a file was executed or a directory was accessed.
LastActivityView
You can view a full explanation regarding it (Minutes: start from 6:15, end whenever the explanation is finished.)
How to ScreenShare [Beginner's guide] - YouTube

JumpLists

Windows keeps track of accessed folders, files and disks in the recents folder (accessible by shell:recent with WIN+R); bypassers tend to clear recent files by deleting RecentDocs with Regedit, however, JumpListsExplorer reads data through Automatic and Custom destinations on the device that can not be bypassed unless disabled.
Eric Zimmerman's tools

Chapter II

Common Methods

Even today, bypassers use things like USB Drives, mouse macros, file replacement and so on to attempt and bypass SSers. Majority of these methods are caught using System Informer regexes;
System Informer allows us to read traces of the whole system by analyzing services, every action on the device or execution of a file leaves residues in services and USN Journal.
For example, renaming the extension of .exe to .png then using the CMD to run the renamed executable as .png will leave a trace in Prefetch (refer to Prefetch section of this handbook), but what if they delete both the file itself and its prefetch trace, as well as BAM values and delete only the .png trace's recent file, then somehow manage to bypass JumpLists too?
To detect such actions, we will need to use multiple tools at once. These tools are:

System Informer | Search Everything | JournalTool | Richard

System Informer

System Informer is an application that lets you read memory of the services; services, on the other hand, log information of all processes on the device (executed files, accessed directories, pop ups and so on), I will quote the ScreenShare Community with their guide on how to dump System Informer:

Because most people don't know how to filter in Process Hacker due to misinformation problems, since the same format of: *Length 4 - Image & mapped* was always used, I will explain the utilities of these.

Height
It is the weight in bytes that has the value to show, Example:

Let's pretend that this text: `1892js` has 4 bytes of space occupied in the Process/Service to be filtered, this will be shown because it matches the recently added filter, now, It will not have any kind of utility in File Trackings, since normally, these exceed 10 or even more.

Because of this, it is recommended to filter by `5` - `6` or `10` definitely in case of file tracks.

Image
Can be given to an uncompleted trace, i.e.: example I selected a file named `Hello.txt`, this same `Hello.txt` may not have been saved several times in **Example:** Explorer.exe, hence, it will be an uncompleted, **"**Lost**"** and random value of the same Process/Service to be filtered.

[⚠️] - **It is recommended not to filter image in any filter.**

Mapped
May be due to a direct correlation with the same process to be filtered, where such a byte was connected to such a process and it is not clogged, they will be verified as something that really happened and will serve for data changes between processes.

[⚠️] - **Remove this option may cause loss of information when performing a Screenshare, it is recommended to enable it at all times**.

Extended Unicode
According to a Process Hacker administrator, it is used to show accents in Process Hacker filters, in many cases, it usually shows a small part of the Unicode table.

[⚠️] - **It is recommended to enable this.**

Since System Informer is new, SSC Hasn't stated this in the guide:

Make sure you enable Kernel mode Driver from settings, otherwise reading process memory will be impossible to do.

Demonstration

Explorer

By dumping the Explorer service, you'll be able to view data of either executed, accessed, loaded on screen applications - this isn't proof of execution but proof of presence on the device, so to say...

Following the Demonstration video in the dumping part of this section, after file:/// (or :\ first) you can dump contains (case-insensitive) .dlls, .exes, .jars, .bats and so on.

You will have to combine the use of Explorer and Evidence of Execution (Prefetch, BAM, PCA, JumpLists and so on) to ban Players in instance.

Csrss

Dumping process remains the same in System Informer regardless of what service you choose. Dumping by :\ first then .exe or .dll in Csrss is the best way to go around proving executions of cheats. Obviously, if you want to prove the date of execution, use Prefetch, BAM, PCA and others.

Demonstration

Renamed Extensions

By dumping explorer, csrss, DPS or searchindexer, using Regex (Case-Insensitive), we will be able to exclude .exes and .dlls from search, thus only leaving other extensions in the results. If Prefetch and BAM don't show other extensions, this might help you find them.

In this case, I'll be using DPS only for Demonstration purposes, and you should start with it, too.

If you filter a file name in DPS then Filter > Contains (Case-Insensitive) and type !! (as a string, just !!) - two different dates means a renamed file was executed. Can hint on File replacement if there are no two same name files on the device. You can double check if they're the same files logged in DPS by executing them and re-opening the system informer.

Demonstration

Search Everything

We can use search everything to sort all .exes and .dlls on the device, even renamed extensions, by specific sizes. For example, most Karma and Vape file sizes range between 17~20MB, with Search Everything, if you type "size:5mb..20mb ext:exe", you will sort all .exe files between the sizes of 5 and 20 MB.

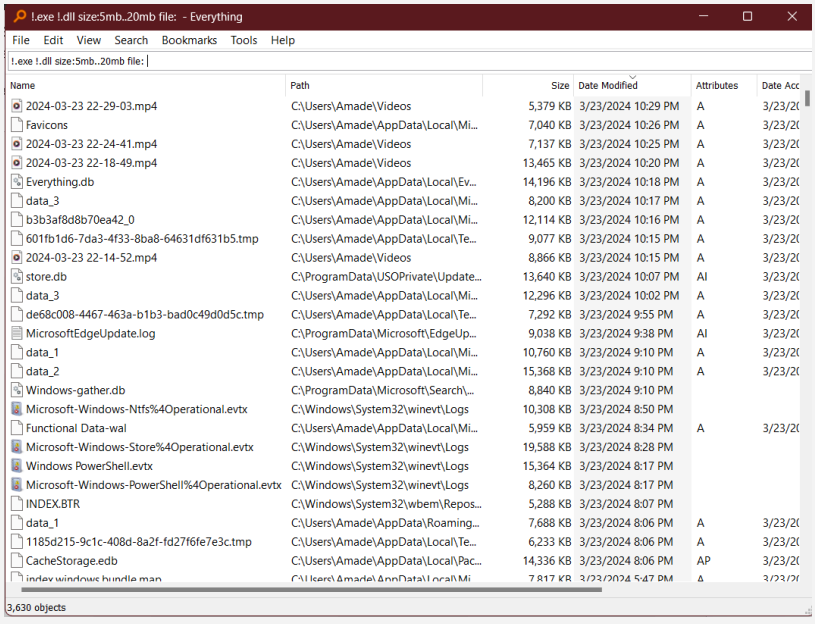
size:5mb..20mb ext:exe - Everything

File Edit View Search Bookmarks Tools Help

size:5mb..20mb ext:exe

Name	Path	Size	Date Modified	Attributes	Date Acc
SpotifyWidgetProvider.exe	C:\Program Files\WindowsApps\Spo...	8,892 KB	3/22/2024 7:02 PM	A	3/22/2(
msteams.exe	C:\Program Files\WindowsApps\Mic...	17,986 KB	3/21/2024 11:50 AM	A	3/21/2(
XboxPcAppFT.exe	C:\Program Files\WindowsApps\Mic...	6,762 KB	3/21/2024 10:03 AM	A	3/23/2(
XboxPcApp.exe	C:\Program Files\WindowsApps\Mic...	13,815 KB	3/21/2024 10:03 AM	A	3/23/2(
XboxGameBarWidgets.exe	C:\Program Files\WindowsApps\Mic...	14,514 KB	3/21/2024 10:03 AM	A	3/23/2(
$R84VJ24.exe	C:\$Recycle.Bin\S-1-5-21-729616088...	16,801 KB	3/19/2024 11:13 PM	A	3/19/2(
BingChatInstaller.EXE	C:\Windows\Temp\MUBSTemp	17,454 KB	3/19/2024 12:34 PM		
BGAUpdate.exe	C:\Program Files (x86)\Microsoft\Ed...	17,627 KB	3/19/2024 12:34 PM	AI	3/23/2(
MicrosoftEdge_X64_122.0.2365.92_122.0.2365.80.e...	C:\Program Files (x86)\Microsoft\Ed...	11,374 KB	3/17/2024 10:05 PM	AI	3/23/2(
systeminformer-setup.exe	C:\Program Files\SystemInformer	15,304 KB	3/16/2024 11:31 PM	A	3/23/2(
setup.exe	C:\Program Files (x86)\Microsoft\Ed...	6,964 KB	3/16/2024 7:24 PM	A	3/20/2(
setup.exe	C:\Program Files (x86)\Microsoft\Ed...	6,964 KB	3/16/2024 7:24 PM	A	3/23/2(
setup.exe	C:\Program Files (x86)\Microsoft\Ed...	6,964 KB	3/16/2024 7:24 PM	A	3/23/2(
MicrosoftEdge_X64_122.0.2365.92_122.0.2365.80.e...	C:\Program Files (x86)\Microsoft\Ed...	11,374 KB	3/16/2024 7:24 PM	AI	3/23/2(
ntkrla57.exe	C:\Windows\WinSxS\amd64_micros...	10,950 KB	3/12/2024 9:46 PM	A	3/18/2(
ntkrla57.exe	C:\Windows\System32	10,950 KB	3/12/2024 9:46 PM		
vmms.exe	C:\Windows\WinSxS\amd64_micros...	14,482 KB	3/12/2024 9:46 PM	A	3/18/2(
SenseNdr.exe	C:\Windows\WinSxS\amd64_windo...	9,127 KB	3/12/2024 9:46 PM	A	3/18/2(
SenseNdr.exe	C:\Windows\WinSxS\amd64_windo...	13,605 KB	3/12/2024 9:46 PM	A	3/18/2(
LockApp.exe	C:\Windows\WinSxS\amd64_micros...	10,900 KB	3/12/2024 9:45 PM	A	3/17/2(
LockApp.exe	C:\Windows\SystemApps\Microsoft....	10,900 KB	3/12/2024 9:45 PM	A	3/23/2(
Taskmgr.exe	C:\Windows\WinSxS\amd64_micros...	5,268 KB	3/12/2024 9:45 PM	A	3/17/2(
Taskmgr.exe	C:\Windows\System32	5,268 KB	3/12/2024 9:45 PM	A	3/23/2(
ntoskrnl.exe	C:\Windows\WinSxS\amd64_micros...	11,798 KB	3/12/2024 9:44 PM	A	3/17/2(
ntoskrnl.exe	C:\Windows\System32	11,798 KB	3/12/2024 9:44 PM		
explorer.exe	C:\Windows\WinSxS\amd64_micros...	5,244 KB	3/12/2024 9:44 PM	A	3/17/2(

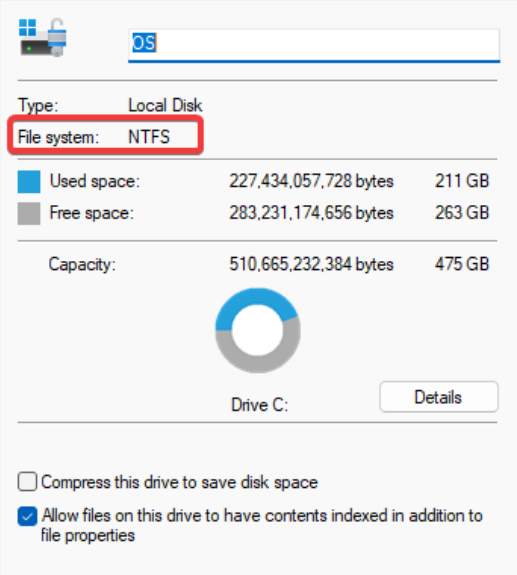
If you want to check for renamed extensions, you can use this regex instead:
"!.exe !.dll size:5mb..20mb file: "
Sort by Date Modified, this way you'll only have to check exec--utions of files in current date. Date modified on renamed extensions updates upon launch.



JournalTool can be used to prove deletion and renaming of files.

Quoting Red Lotus SS Guide:

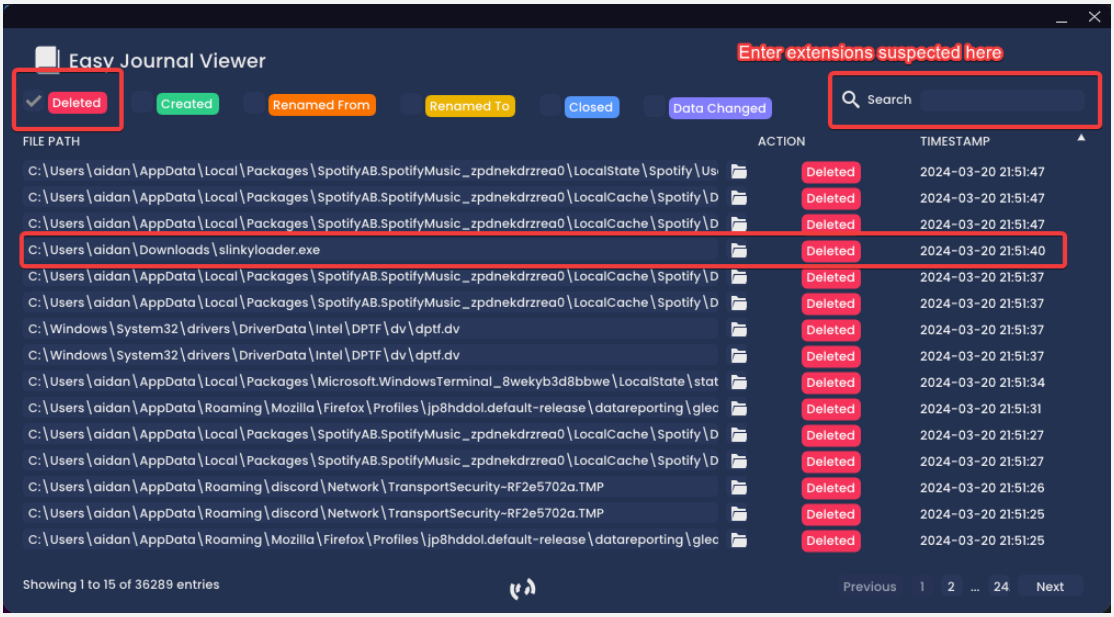
First, check if the drive is anything but FAT32. If it is FAT32, other methods will have to be used since it does not contain a USN log of data.

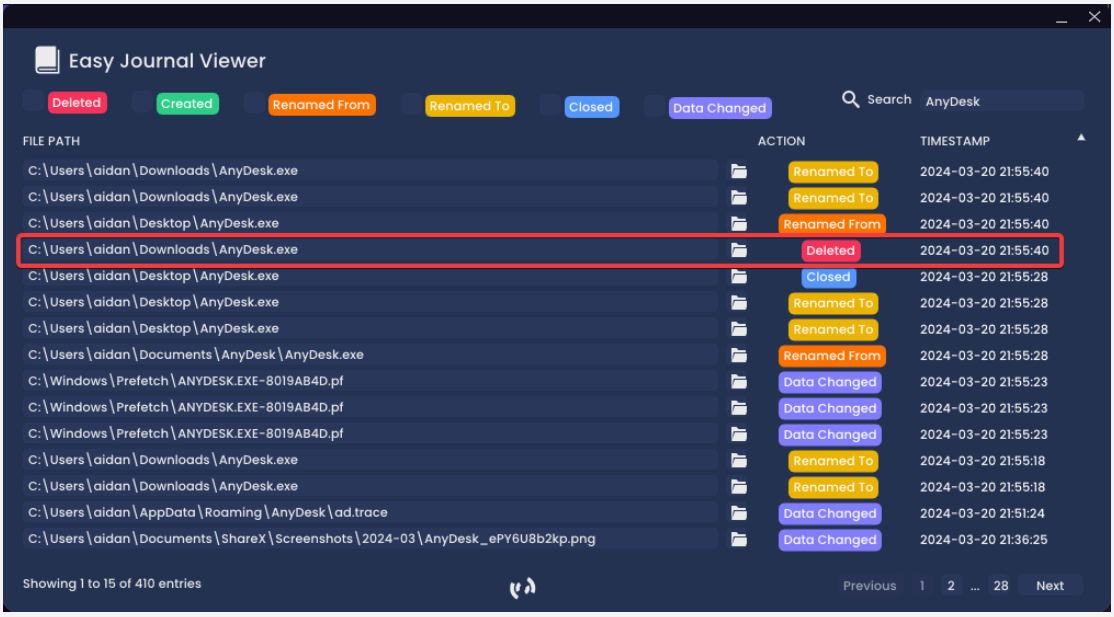


We can detect shift deleted files by using Echo's Journal Tool.

[Echo Journal Tool]

Replaced files will also show up as deleted:





Task Scheduler

Nowadays most Bypassers use Task Scheduler to bypass the SS, as launching files this way bypasses Prefetch. You don't need to know what exactly it is at this stage, just use System Informer to dump Scheduler service and filter <Command.

[Demonstration]

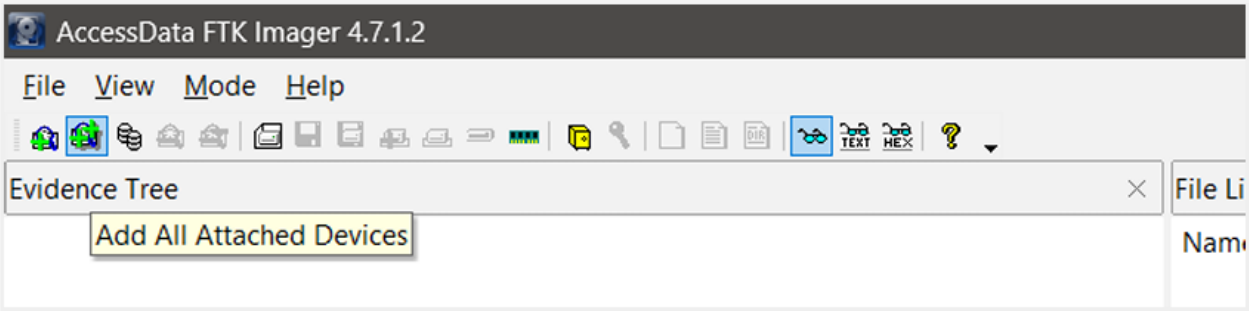
Chapter III

USB Methods

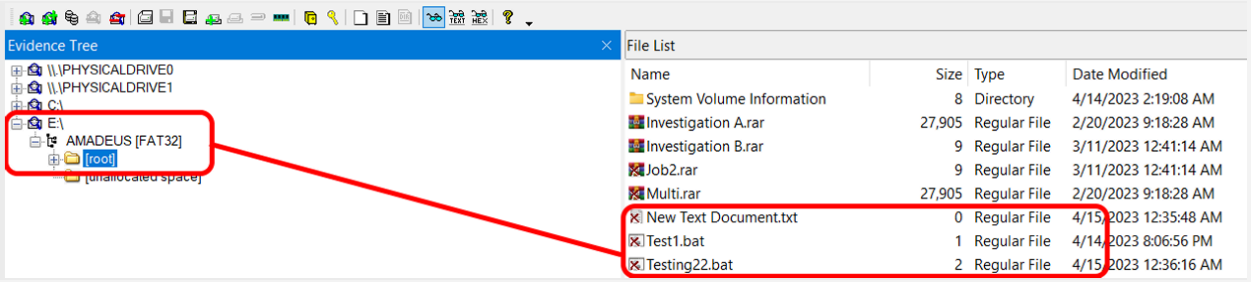
FAT32 Deletions

https://d1kpmuwb7gvu1i.cloudfront.net/AccessData_FTK_Imager_4.7.1.exe

Download FTK Imager and add all attached drives:

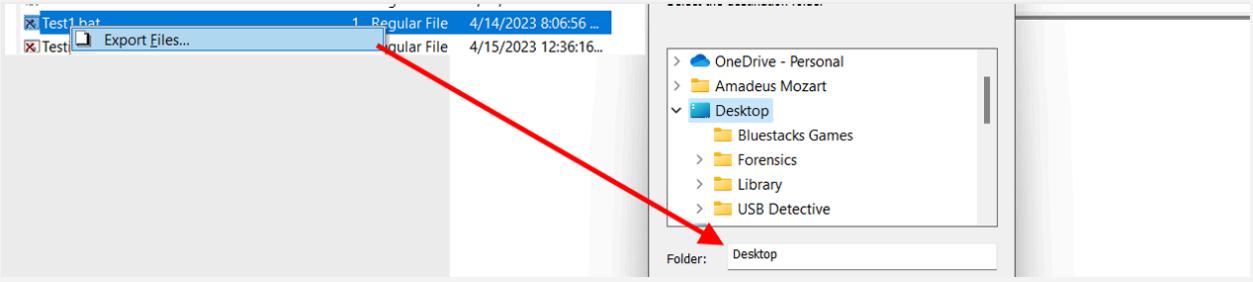


Select a specific driver letter, navigate thru Root of it and check deletions.



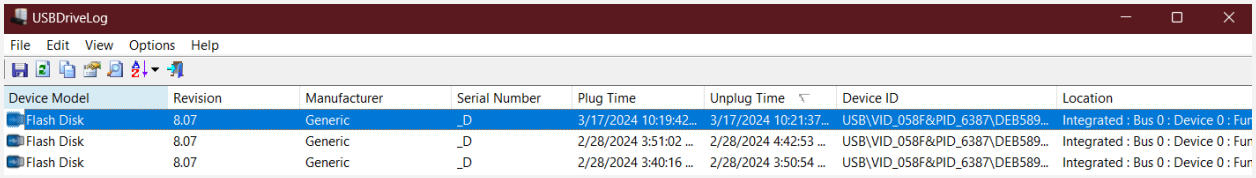
Before banning for a file that's marked with Red X, prove execution of it. In these cases, just dumping processes like Csrss, DPS, SearchIndexer, Pcasvc is enough with the specified file name.

You can also try recovering the deleted file, if Size doesn't show 0.



USB Removals

USBDriveLog is a Nirsoft application that can display a list of unplugged USB Drives!
[USB Drive Log For Windows 10 and Windows 11 (nirsoft.net)]



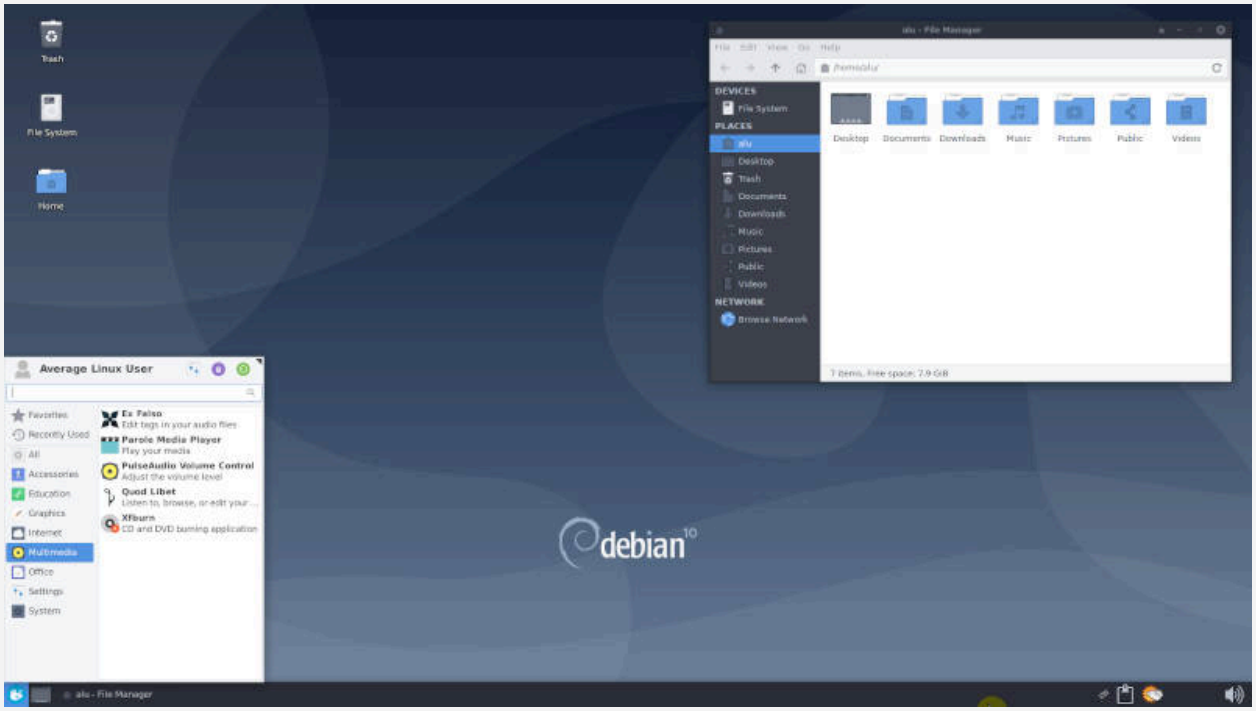
Chapter IV

Linux SSing

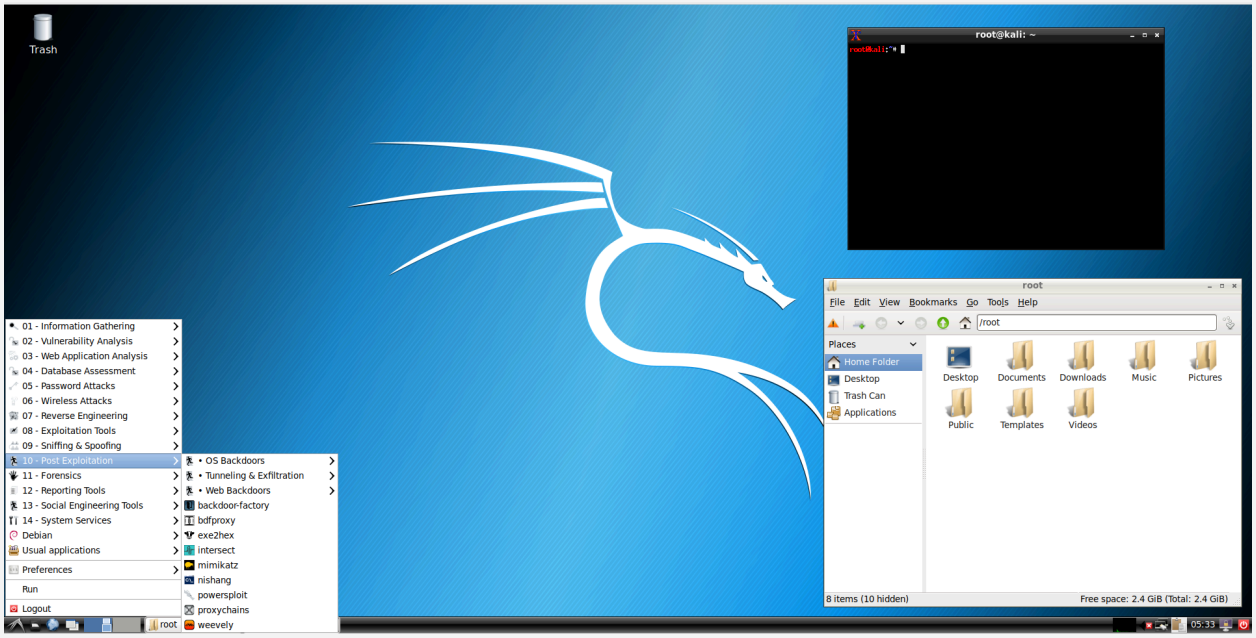
Introduction

Linux ScreenSharing is by far the least investigated due to the numerous operating systems. (Debian, Ubuntu, Kali, Mint, XFCE, Fedora, etc)... Here are some example desktops from chrome:

XCFE Debian



Kali Linux



The guide concentrates on the most **commonly** used systems, as well as the most working methods.

Terminal

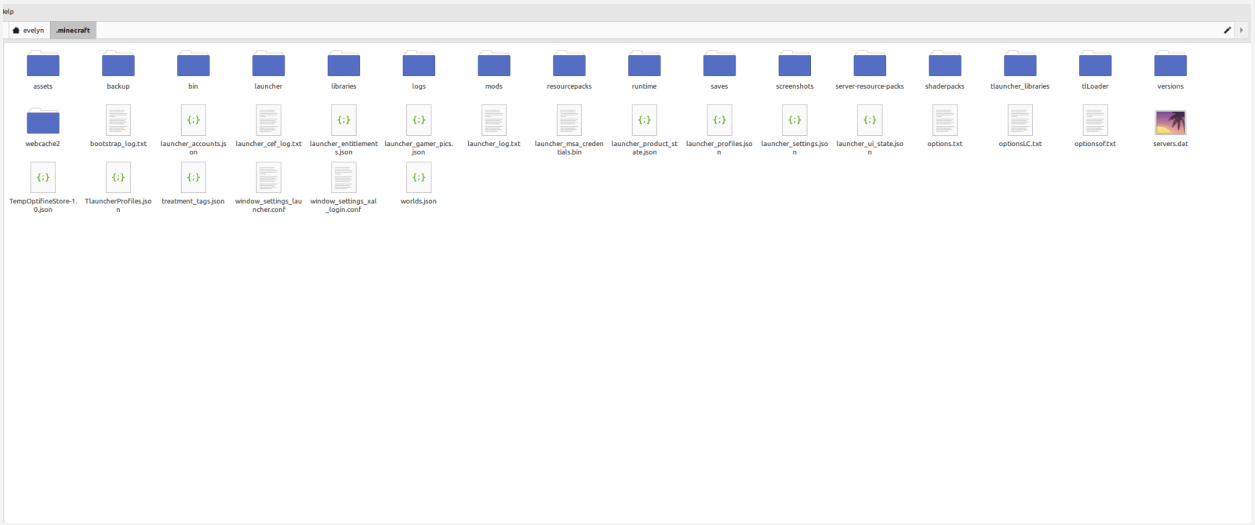
CMD on Linux is called the “Terminal”, opens by pressing **Ctrl+Alt+T** and is mainly what we’ll be using for ScreenSharing. Alternatively, simply press the terminal button in their taskbar/wherever you see it. It looks like CMD or something like this >



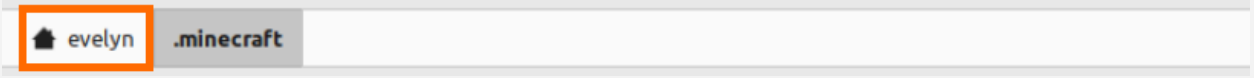


Directories

Everything on the Linux is stored in users folders, for example, if we want to go to the .minecraft folder, we'll go to the users menu and type the directory there, in this case we type **/home/user/.minecraft** - note that instead of "user" we put their system's username. We'll check everything in .minecraft starting from user caches to Mods, logs, etc.



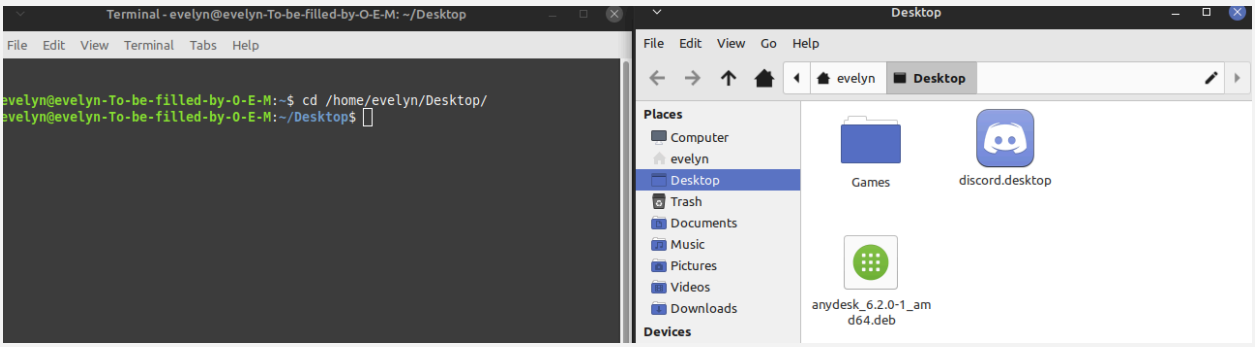
We'll know the Username by paying attention to the search bar. The search can be edited by double clicking it.



You can also simply do this by using the Terminal (way easier than this if you don't understand), we open Terminal and type the following

cd /home/user/.minecraft

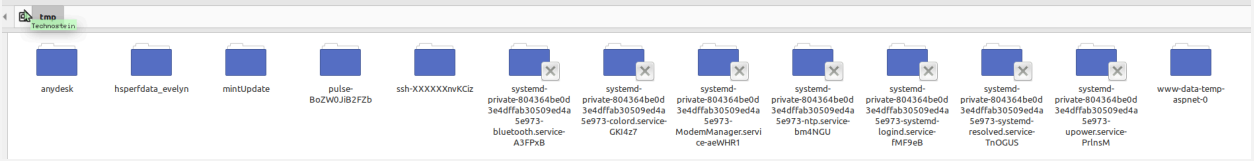
It works with any directory, for this instance we'll use Desktop for it.



This obviously works with any directory in .minecraft, as well as for Lunar client, etc. Directory for Lunar is

cd /home/username/.lunarclient/

For accounts/ban evasion you check the **game folder** while for chat logs/related you check the **offline folder.**
Go to **/tmp/** in the search bar to check temp files. JnativeHook can be checked here as well.



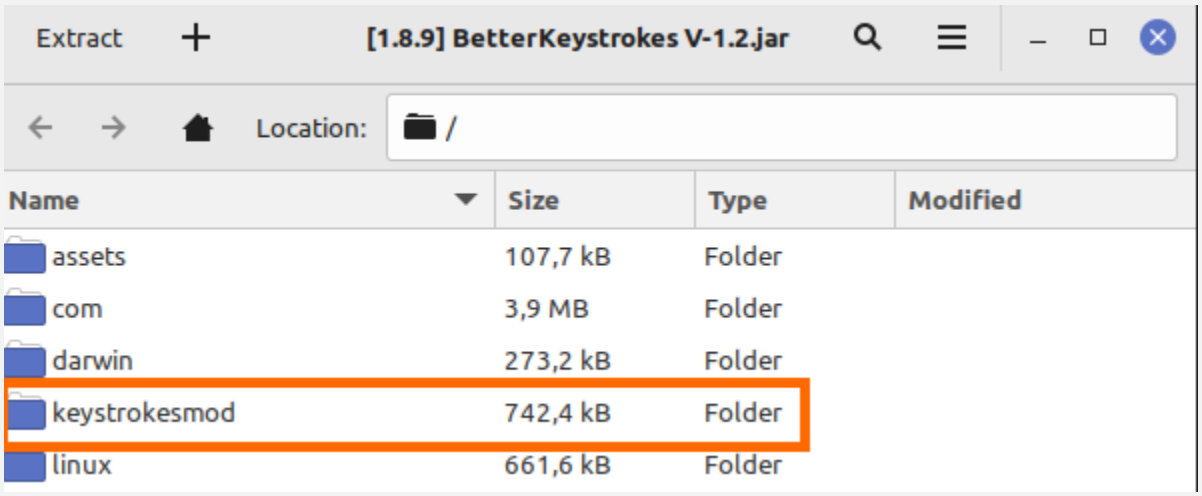
Opening Minecraft/.Jar files

While some SS tools indeed do work on Linux, downloading them might be a bit complicated in some scenarios as we use the Terminal to download files as well. However, there's a bit simpler way to do that which also works on Windows.

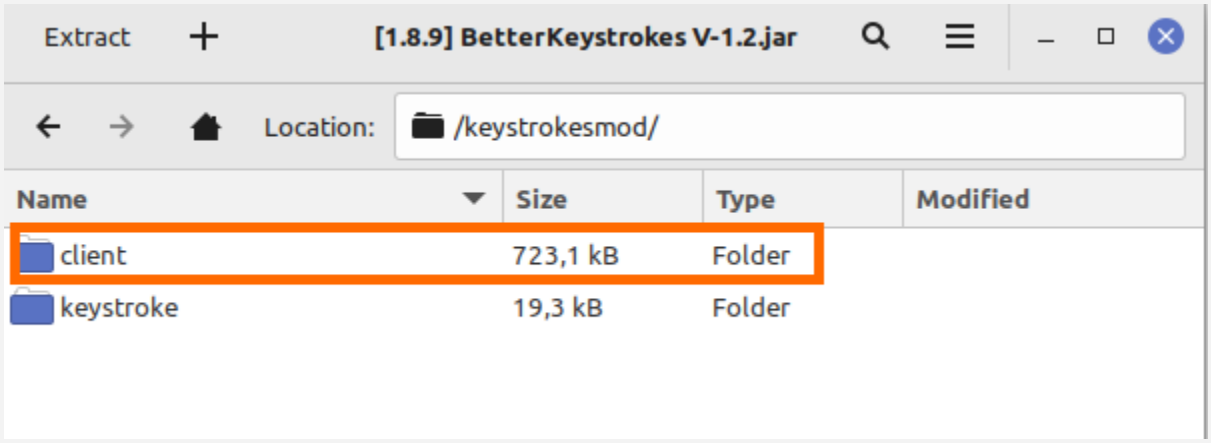
We'll begin with opening Minecraft, for Lunar and Tlauncher, you'll have to right click them and open them with Java.



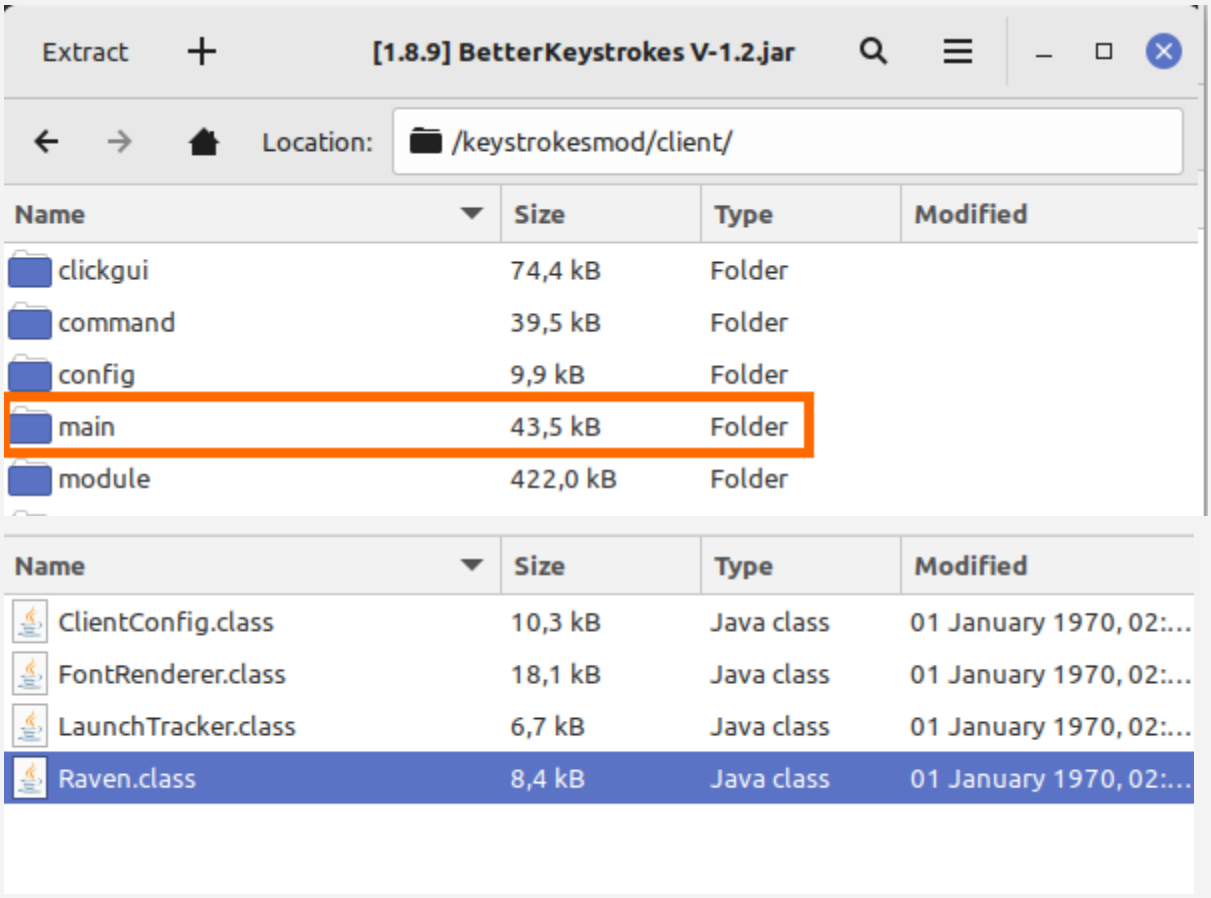
For opening .jar mods, you'll hover over "Open With" and press anything regarding open with Archive/Rar/Zip. Analyze everything shown there, imagine they're simply classes of the mod.



You're mainly looking for the mod name and a folder called **client** in these cases.



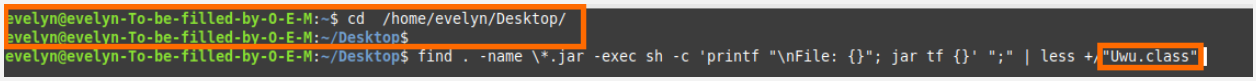
Usually this will leave you with a bunch of folders that represent mod classes, for example here you're looking for the **main** folder as we're investigating Raven.



This process is a bit more generalized if you encounter different .jar files and you're unsure of what to look for, use Terminal. Cd the directory of the mods folder in .minecraft (if they're on forge) and type the following command in:

find . -name *.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"name.class"

Note that instead of the “name” in .class we’ll filter already known to us classes. Check the Intermediate Windows guide to know what to look for in multiple jar mods. We’ll use Desktop as an example for now.



You can also go to the mods directory manually and right click > Open in Terminal to open the terminal in that specific directory.

You may filter classes such as **AutoClicker, AimAssist, Reach, Destruct** etc as these are commonly in the jar cheat mods. If anything pops up in the results that indicates this class is from that specific Jar file (for example it shows the directory of the mods folder with the mod name and UwU.class etc) you can ban.

Modified recycle bin

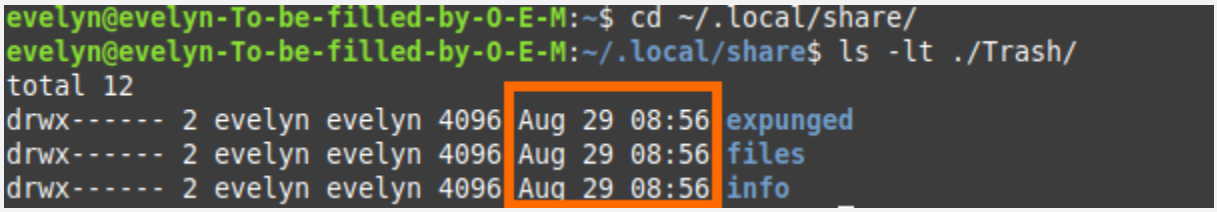
To check the last modified date of the recycle bin, easiest way is by using the Terminal and typing the following commands:

To get to the recycle bin directory, we type the following first:

cd ~/.local/share/

Then we’ll check the last modified time of the bin by typing:

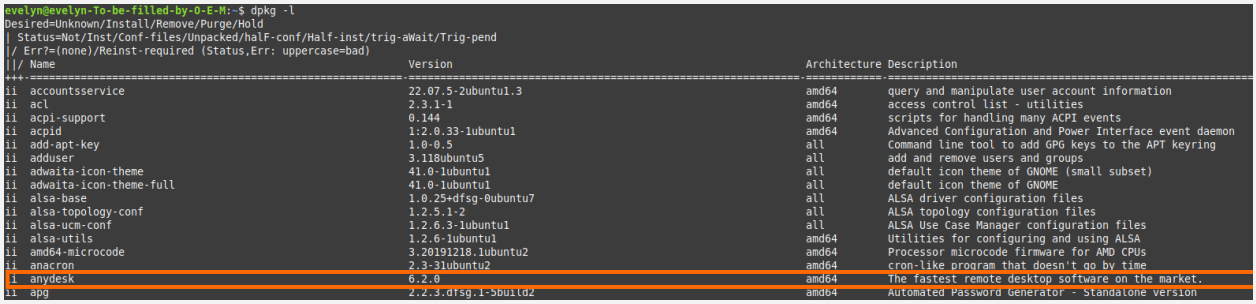
ls -lt ./Trash/



Installed applications

Checking installed might help us catch users with a VPN or anything related, possibly even clickers as well. To access the list of currently running programs, we type the following command in the Terminal:

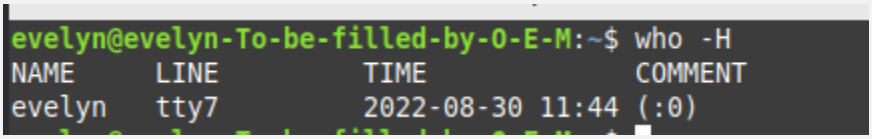
dpkg -l



This may give us an idea of what to look for. To check when they last ran an application, type **-mtime +1** in the Terminal, compare the dates to minecraft logs.

Current user

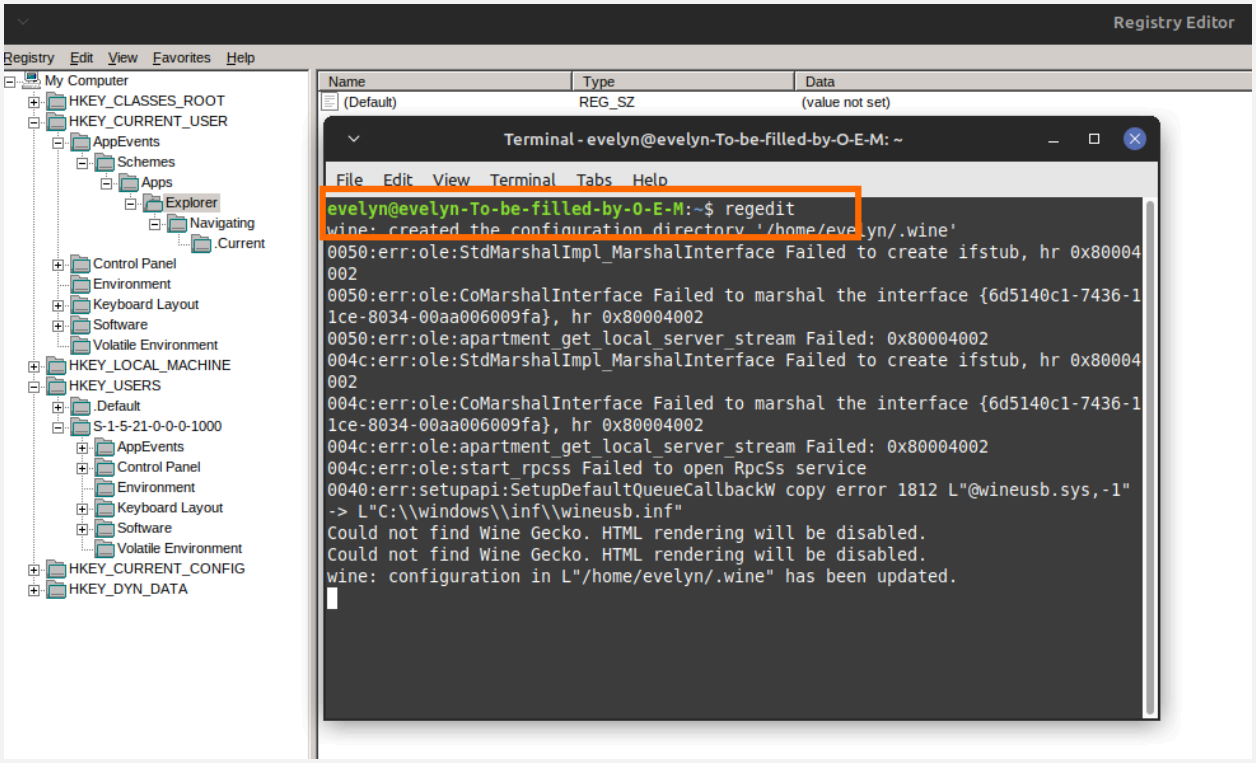
Knowing the current user's name will help you with Terminal directory commands. To know the logged user into the system, type **who -H** in the Terminal.



Regedit

Regedit requires wine to be accessed on Linux. However, it also installs itself when you type **regedit** in the Terminal. More about this in the future references. Note that

it may take up to a minute to install so be patient.



Fast access

You can also access and open files using Terminal, this can save some time if you can't find the necessary directory.

Open the Terminal and cd to the directory you want to, we'll go to the game directory in Lunar for this instance.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ cd ~/.lunarclient/settings/game
evelyn@evelyn-To-be-filled-by-O-E-M:~/.lunarclient/settings/game$ open accounts.json
```

Simply type **open accounts.json** to run the file. Note that you can also use the CMD to open applications instantly without going to their directories by just typing out the application name. (For example, XClicker if it shows up in -mtime -1)

Use this to start **minecraft-launcher** if they have that and run the latest version they were on, go to singleplayer from there and press right shift, if a cheat GUI shows up, ban for cheating.

Restarting shell before SS

On Linux, explorer is called “shell”. To check if they restarted their shell before SS, we’ll need to compare their shell time to their system time.
This process may be a little complicated, as we retrieve the PID (Process ID) first, and to get the shell’s ProcessID, we first need to know which **environment** they’re on. We do that by using an echo command:

echo **$XDG_CURRENT_DESKTOP**

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ echo $XDG_CURRENT_DESKTOP
XFCE
```

In this case, we’re on XFCE environment. Every desktop has its own command to retrieve the PID for. In this case, we type **pidof xfdesktop.**

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ pidof xfdesktop
1442
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

This is different for every desktop. Here are some known commands -

[
Pidof gnome-shell - Linux Gnome
Pidof nemo-desktop - Linux Cinnamon
Pidof dde-desktop - Linux Deepin
Pidof caja - Linux Mate
Pidof kali-desktop - Kali Linux
]

Then, we’ll need to retrieve the system’s PID, we do that by typing **pidof systemd** this time instead.
Now, to get the runtime of the desktop and the system, we type the following:
ps -p **,**** -o lstart,etime,cmd** - we input the PID of the processes in place of *-s.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ ps -p 1442,1222 -o lstart,etime,cmd
                  STARTED     ELAPSED CMD
Tue Aug 30 11:44:48 2022    07:25:53 /lib/systemd/systemd --user
Tue Aug 30 11:44:49 2022    07:25:52 xfdesktop
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

If time elapsed shows it’s before SS, they’ve restarted the shell before SS.

Modifying files

If players modify files before SS, you can either check the modified file or ban right away. However, if that's not the case, you can keep investigating.
To check what commands they've ran regarding .exe files, type the following:
history | grep .exe

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ history | grep .exe
   52  mono hello.exe
   54  mono hello.exe
   71  history | grep .exe
  129  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' :;: | less |/".Uwu.class
  130  find %{1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
  131  find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
  133  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".Uwu.class
  134  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".reach.class
  135  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".Uwu.class
  136  find . -name \*.jar -exec sh -c 'printf "/nFile: {}"; jar tf {}' ";" | less +/".reach.class
  137  find . -name \*.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"Uwu.class"
  140  find . -name \*.jar -exec sh -c 'printf "\nFile: {}"; jar tf {}' ";" | less +/"Uwu.class"
  141  history | grep .exe
evelyn@evelyn-To-be-filled-by-O-E-M:~$
```

This will show if they did anything with .exe files. Anything suspicious like **mv AutoClicker.exe AnyDesk.exe** means they've renamed AutoClicker to AnyDesk.
To look for generally recently modified/accessed files (which shows the list of accessed files in the past 24 hours, this is what restarting shell clears/tries to hide) type the following command:
find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe" - you may also type .jar/.class/.o etc.

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ find ${1} -type f | xargs stat --format '%Y :%y %n' 2>/dev/null | sort -nr | cut -d: -f2- | grep ".exe"
2022-08-30 17:50:06.470204334 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000147.log
2022-08-30 17:42:14.583508098 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG
2022-08-30 17:42:14.579507972 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/MANIFEST-000001
2022-08-30 17:42:14.575507849 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000009.ldb
2022-08-30 17:42:14.567507599 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000007.log
2022-08-30 17:37:14.582170689 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_trello.com_0.indexeddb.leveldb/000049.log
2022-08-30 16:49:41.257481332 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG
2022-08-30 16:49:41.253481208 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/MANIFEST-000001
2022-08-30 16:49:41.245480958 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/000149.ldb
2022-08-30 16:08:39.373080037 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_www.youtube.com_0.indexeddb.blob/b/00/3c
2022-08-30 16:08:00.867886260 +0300 ./.config/BraveSoftware/Brave-Browser/Default/IndexedDB/https_trello.com_0.indexeddb.leveldb/LOG
2022-08-30 15:19:02.875929447 +0300 ./.config/discord/IndexedDB/https_www.youtube.com_0.indexeddb.leveldb/LOG.old
```

Unplugging USB

Unplugging USB before SS results in a ban. To check when they've unplugged the USB, run the following command:
journalctl | grep disconnect | grep usb

```
evelyn@evelyn-To-be-filled-by-O-E-M:~$ journalctl | grep disconnect | grep usb
Aug 24 07:51:16 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 5
Aug 24 08:01:50 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 6
Aug 24 08:02:16 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 7
Aug 24 08:08:51 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 4
Aug 24 08:33:51 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 08:52:44 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 5
Aug 24 09:16:02 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 09:26:54 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 6
Aug 24 10:10:24 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.3: USB disconnect, device number 3
Aug 24 10:13:00 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 3
Aug 24 10:33:31 evelyn-To-be-filled-by-O-E-M kernel: usb 2-1.4: USB disconnect, device number 5
```

Chapter V

MacOS SSing

Introduction

MacOS is yet another least studied OS. However, it is also quite simple to SS. The most beginner-friendly SS Guide has already been made by an old friend - Ovq. Here's the guide:

MacOS | SS Guide

Advanced Section

Chapter VI

History of Bypasses📑

This Tome release will cover the history of the development of Bypass methods, where they took inspiration from, what were the strongest Bypasses from 2019 to present day (2024).

It is no secret that besides the ScreenShare Community, becoming more popular on June 19th 2021, there were anti-screenshare communities (mostly private) that began developing methods for bypassing; communities like Pragma, Cheating Community, Kraken, Legion, Stringless, EzBypass, EndofScreenShare, Stella, Katana AntiSS were (some still are) dedicated to bypassing ScreenShares - Katana AntiSS being the most known and prioritized. Besides them, many Cheat communities and Discord were created such as, for example, Exelon, Odisea, Akira, Isolation, *obviously for Vape & Skript furthermore* and so on*;*

The ScreenShare Community has had to deal with the rise of Bypass servers and the most skilled Bypassers at the time - Notsnakesilent, Xavi, UUID and others - HogRider, Sookiuk, Moostt_, and many others during ScreenShare Events held in SSC itself.

As of today, the majority of these bypass communities have died out and now ex-bypassers have either quit the community, switched to SSing or maintain a chill “this game sucks, let me cheat!” type of vibes!

Xavi is currently a Delegation member at Red Lotus (ex Headquarters), potentially still the “scariest” Bypasser alongside with Notsnakesilent;

Notsnakesilent now helps the CEO (Owner) of SS Community - Rancio with a SS Tool - Ocean;

In 2024 the ScreenShare Community officially discontinued, after guiding and helping the ScreenSharers, practically ever since 2020, officially since 2021. Being the first major and strong community for ScreenSharers and server staff during the times of no DFIR methodologies being even discovered at all by the community, SS Community was the

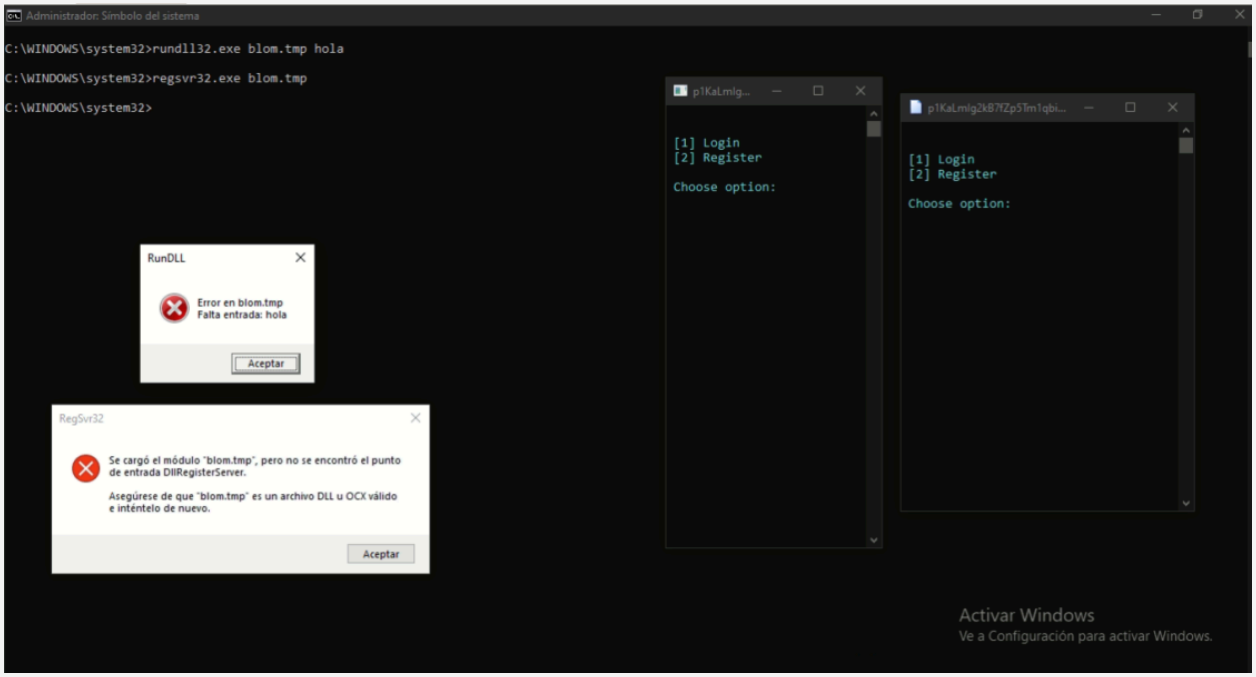
biggest project that inspired Red Lotus itself. *Side note to remark how old the SS Community actually is - ChatGPT wasn't even a thing when we started releasing DFIR scripts;*

First wave

The first wave of Bypasses were pathetic in the English speaking communities, as it mainly consisted of shift+deleting files, their prefetch traces, clearing registry entries, replacing files, injecting DLLs through ProcessHacker2 (and other Injectors), editing the date modification of cheats (using external softwares), clearing registry, using .jar cheats with obfuscated classes and so on. As this isn't a primal handbook, it would be uncharacteristic for Red Lotus to be discussing each and every basic bypass individually, thus we'll concentrate on more complex bypasses from 2019~2020 that might even work today.

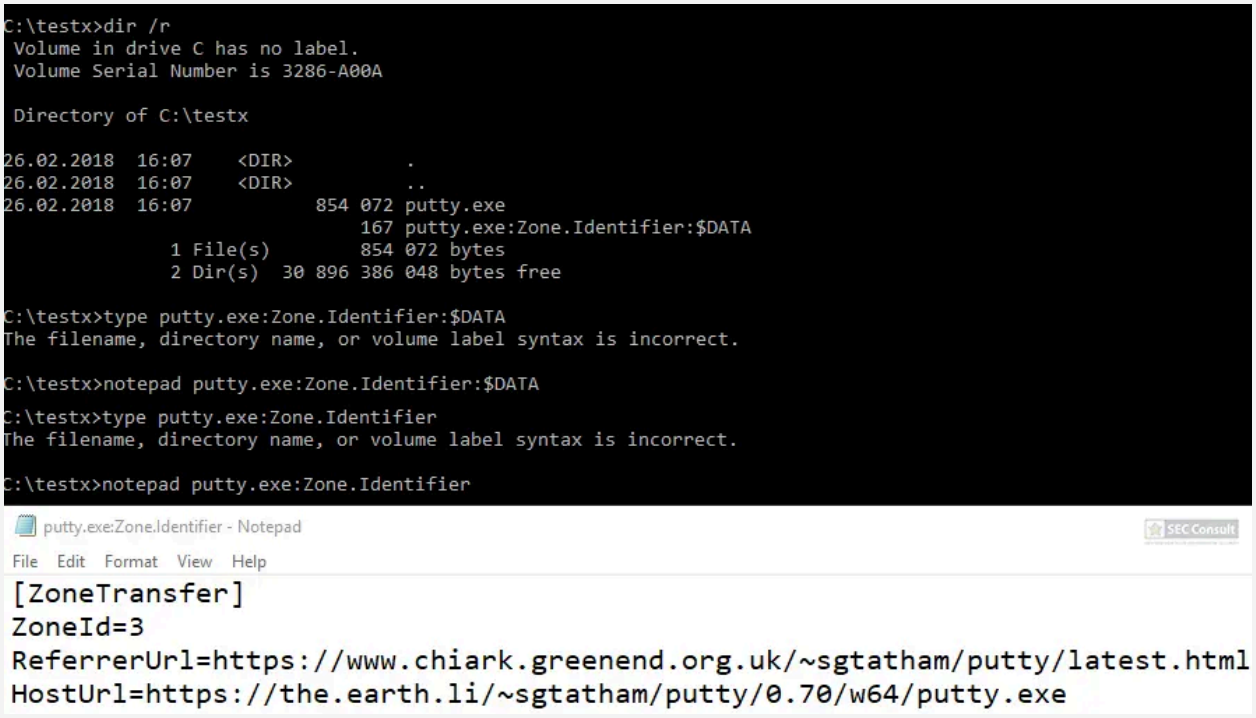
Read-only rundll & regsvr

Attribute "read only" on a .pf trace prevents it from updating and re-logging new actions within the trace. In other words, if data gets *written into* a file / trace / directory every update sequence, setting it to *read-only* blocks that operation.
RunDLL32 and Regsvr32 are used to register / run DLL files, it becomes increasingly challenging to prove an injection of DLLs run this way.



Alternate Data Streams

Although popular in 2019, the [Alternate Data Stream]() method is probably the most undetected bypass even today.



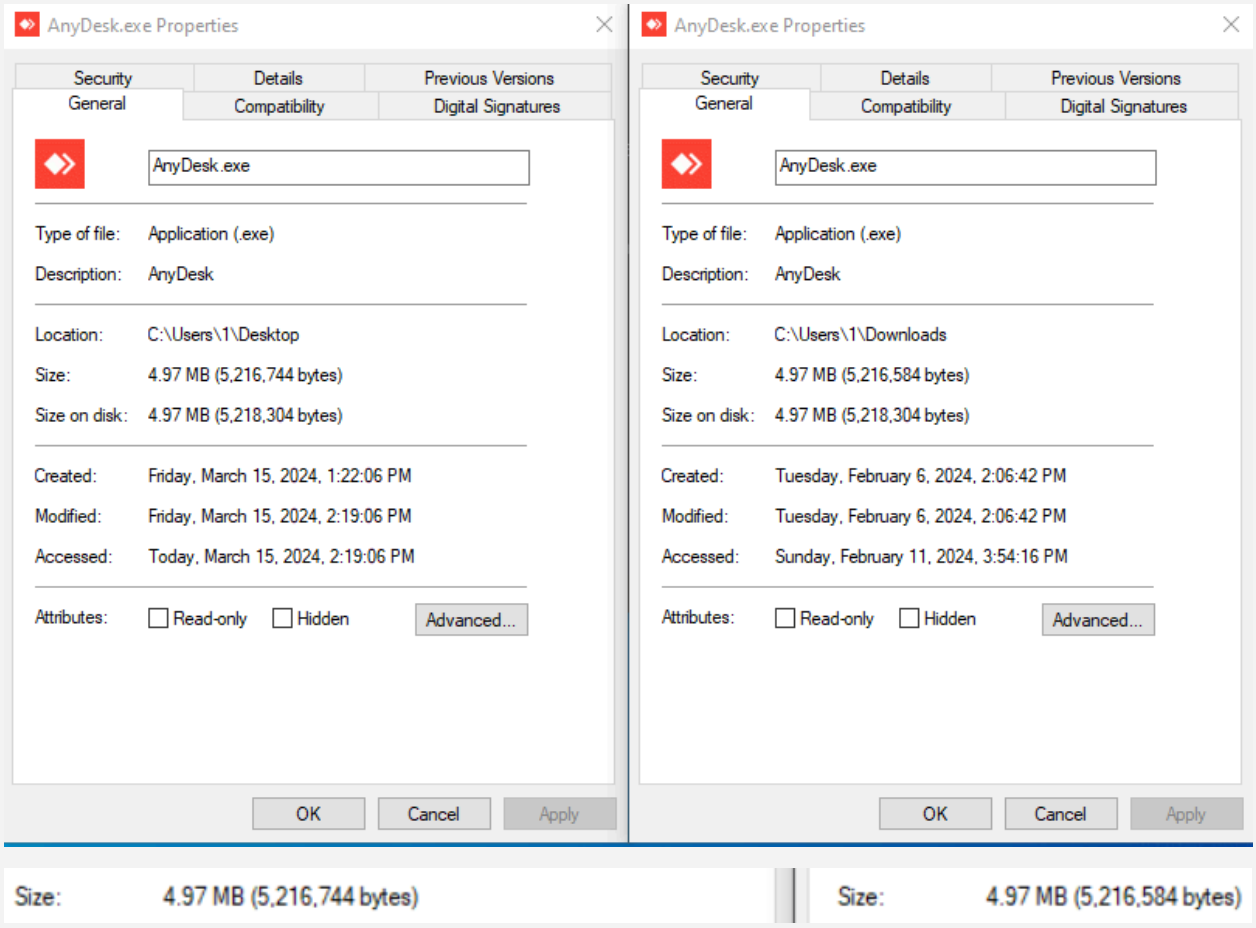
The whole purpose of this method is to hide the file itself on NTFS, if utilized with directory junctions (view the article above), as well as using a Unicode character invisible to System Informer, a smart bypass can be created, making cheats practically impossible to detect if an SSer is unaware of such possibilities or doesn't understand it at all.

Unicode Characters

Universal Character Encoding is often still used today to bypass System Informer checks or to confuse ScreenSharers with primitive or simple methods (that just check .exes, just .dlls and so on), for example, System Informer doesn't support Chinese and Georgian alphabets, thus making any device with these two default languages even tougher to ScreenShare, as every single folder can be put to Chinese/Georgian and USN Journal will not recognize these languages either (displaying them as question marks ?)

Camouflaged Cheats

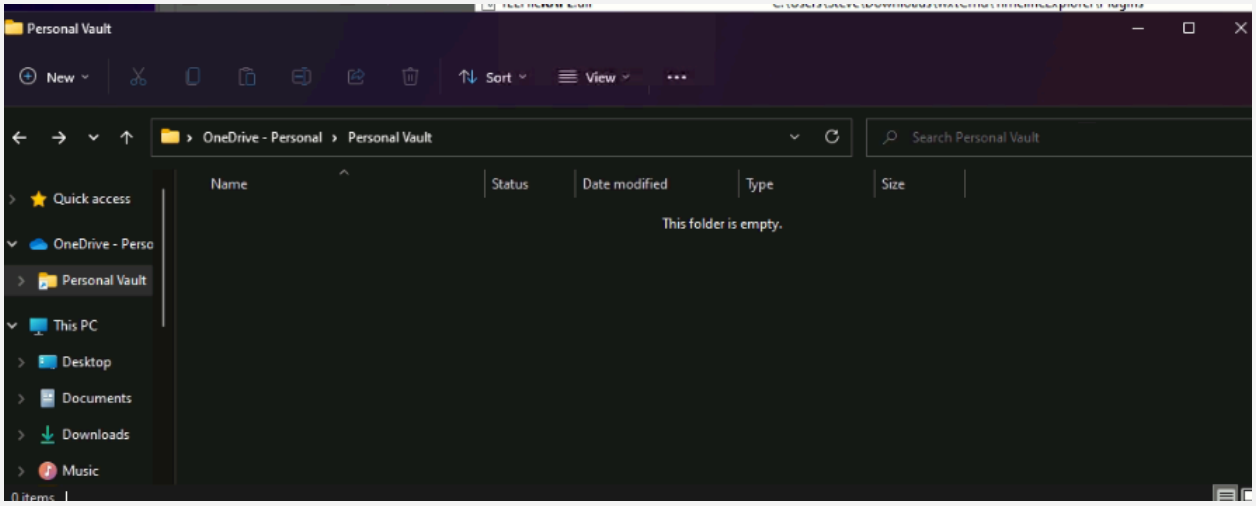
The whole purpose of camouflaging a file is to make it look like legitimate files like Lunar Client, Process Hacker 2, Spotify, AnyDesk, Zoom, Notepad and so on. Yes, all of the apps I've just listed have been turned into cheats during that period. Here's an example (press CAPS+TAB after launching notepad.exe to open the cheat, I'd not recommend running it on your own device though.) download notepad clicker; even Katana anti SS Tool, for example, camouflages as AnyDesk! (the screenshot below isn't Katana but an AnyDesk Clicker, notice how the file sizes are different although everything else looks legit:)



Most of these camouflaged cheats are just AutoClickers, thus they're pretty simple to find but harder to prove, easy to miss these tiny details if one hasn't developed great ScreenShare logic & sense.

Cloud-stored Cheats

Although it sounds complicated, Cloud stored cheats refer to clients executed from cloud Directories, such as WPS Cloud, OneDrive and so on. These files can be moved and replaced on OneDrive remotely, from a phone or a second device. Thus there isn't much to do in a ScreenShare if one is unaware of it being a possibility.



Steganography

Steganography is basically hiding malware within legitimate files such as .pdfs or .jpgs / .pngs; unlike Alternate Data Streams, this bypass is easily detectable since the malware is still physically viewable in NTFS. Applying this knowledge for bypassing using complicated cheat executions or cheat files (such as .bats, .py, code files, .hh files or renamed extensions and so on. . .) will pose a challenge for newgen ScreenSharers.

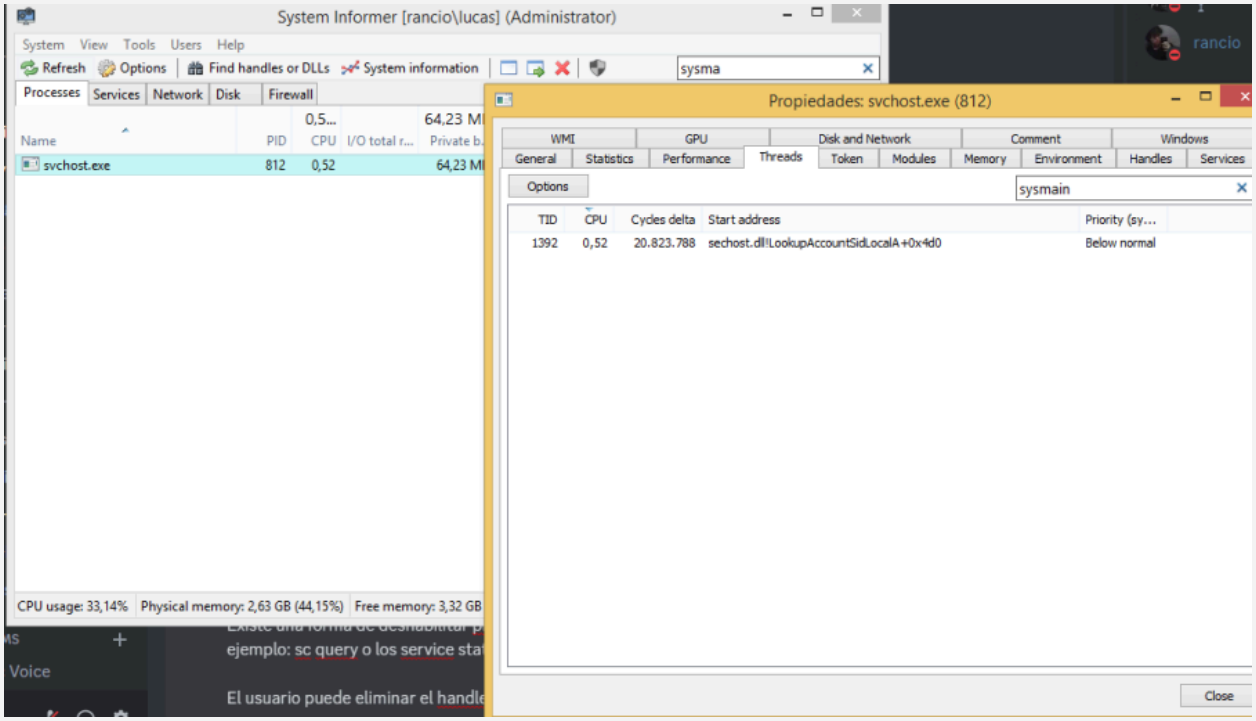
RunPE (VBA)

When we hear "RunPE", we usually mean what's called a "word bypass", there's also "PE Injection" that refers to process hollowing. MS Word allows users to run scripts / cheats via scripts through the RunPE function, making it just harder to prove by date of execution. Most of the players also run codes written in color white, if the SSer doesn't select a region on the canvas - scripts are invisible.

Bypassers from that period also corrupted their CMD (either making them 100% transparent, for example, or making text match the terminal background and so on)

Suspending Threads & Tokens

A more recent (2021~2022) development of bypasses has led to methods that bypass services and drivers (for example, BAM) without actually disabling them. For example, suspending sechost.dll Thread of Sysmain will prevent it from logging .pf traces!



- List

Sysmain: sechost.dll
pcasvc: pcasvc.dll
dps: dps.dll
eventlog: wevtsvc.dll
dusmsvc: wlanapi.dll
cdpusersvc_[numbers]: ucrtbase.dll

Disabled reg inheritance & telemetry

It is possible to disable [almost everything] on the device, including Registry, Activitiescache, JumpLists, it is also possible to disable reg inheritance (for example, of BAM) the same way they disable [directory inheritance].

Chapter VII

📑Anti-Forensic Labs

Bypassers mostly take inspiration from Anti-Forensic Labs, Github (“skidding” source codes to resell them), YouTube or old gen Bypassers (for example, Dream Client’s Task Scheduler Bypass has been adapted by many cheaters and newgen Bypassers, improving its bypassability to sell it as handmade manual bypasses!)
Of course, in this Chapter we will not talk about already known and popularized Bypasses, we’ll cover yet undiscovered, smarter, more effective and complicated bypass methods that will confuse a lot of ScreenSharers.

CompTIA

CompTIA is an IT company that equips students with quite a lot of forensic labs at labsimapp. There are many sources we can learn from about these labs and challenges, for instance, a YouTube Account - vMomentum - elaborates on a lot of these labs.



Blue & Red Team DFIR trainings, infosec, sources for individual training, blogs and more can also be found at SecurityNewspaper web;



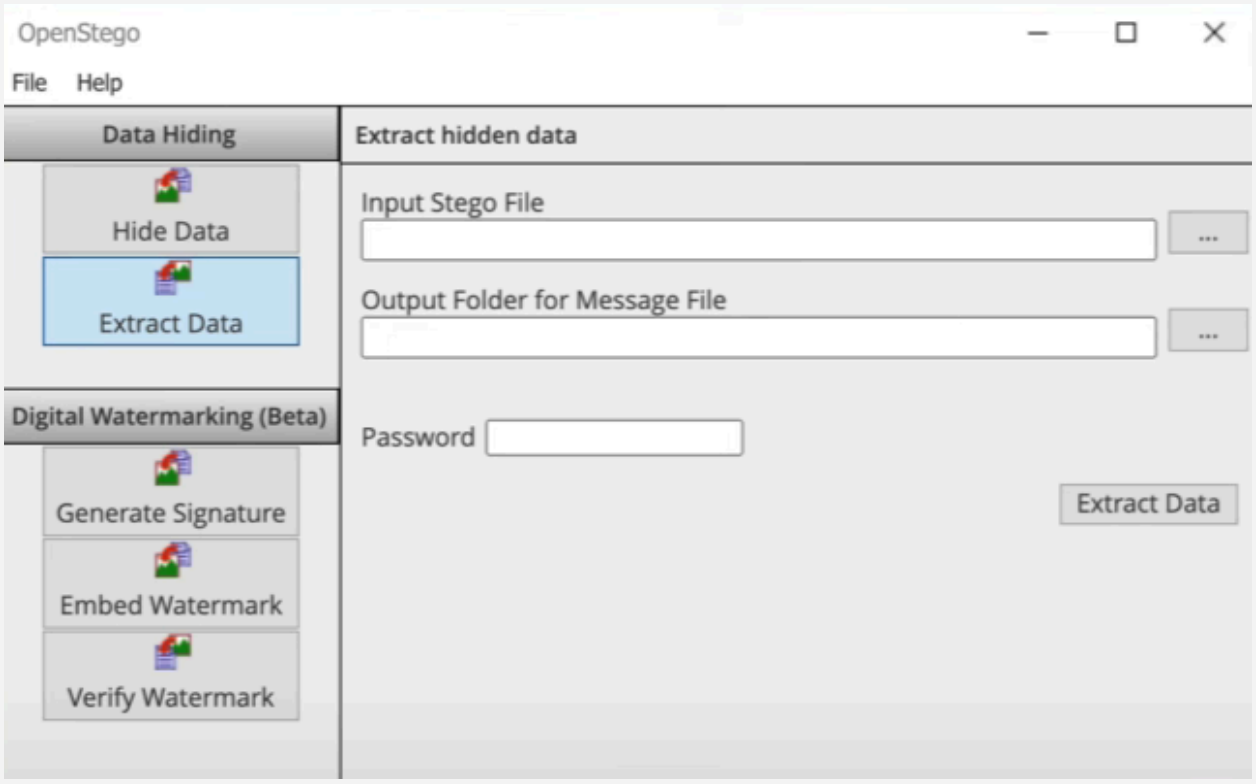
Letsdefend offers both FREE and PAID Blue Team DFIR Courses and tutorials! for example, did you know what a JWT-Attack was before?

Anti-Forensics

Bypasses and Anti-Forensic techniques mentioned below have yet not been used by Bypassers in gaming communities.

OpenStego & Fake Signatures

While Steganography is manually simple to detect as it leaves a lot of traces on the device, utilities such as OpenStego from 2017~2021 make creating such files 10 times easier and more complicated. Using OpenStego, users are also capable of faking Signatures...



Faking signatures is possible with many other utilities as well, in fact, because many cheats get their source codes leaked (especially C# / C++) it becomes possible to not only change the cheat & bypass but to compile it with fake signatures by supplying version files.

Villain

I decided to implement this “bypass” (unpopular but potentially possible in the future) to demonstrate the capabilities of Powershell (will be the best demo for an upcoming universal detections method) - Villain, bypassing the signature check-based Antivirus using Powershell or a Python code.
This method bypasses MPLog & Defender Logs as well. Why is it a relatively useful method? because faking a signature becomes easier to do without leaving a trace on the device (otherwise, faked signatures can often be flagged by the Antivirus even if they were configured on a secondary device / different PC)

DLL Proxying

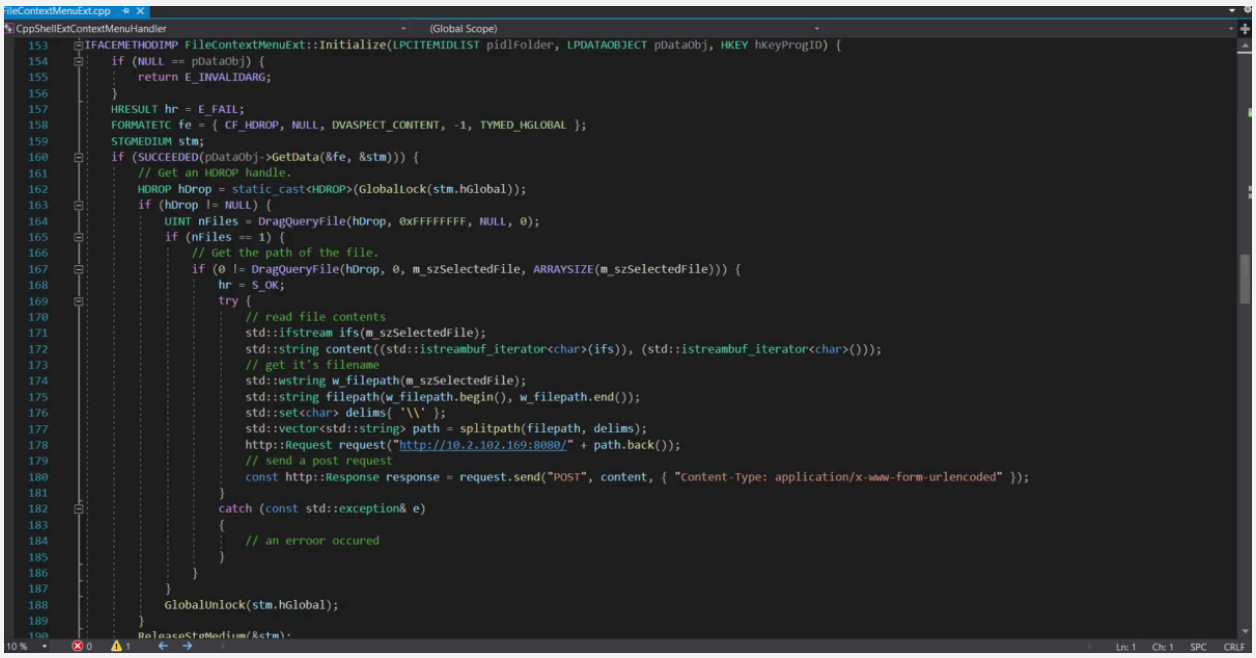
You’ll often hear this term in 1.9+ Minecraft communities, it’s always individuals who have never tried it before nor understand it completely talk about it. Bad news - it is actually a thing; good news - it is super easy to detect.
DLL Proxying example 1
DLL Proxying example 2
DLL Proxying blog
Every experienced ScreenSharer immediately has an idea how to detect it after checking it out.

DLL Hijacking

Unlike DLL Proxying, DLL Hijacking has the most potential to be used by nerdy bypassers (last time I have personally come across this method was somewhere around August 2023), detecting this bypass with Universal Red Lotus methods is simple, though it is still a more realistic & interesting concept to be aware of.
DLL Hijacking example 1
DLL Hijacking example 2 (C#/C++)
DLL Hijacking blog
Detecting this bypass becomes challenging if Bypassers mix it with hex replacement and position files logically on the system, allowing them to seamlessly delete them even before SS (for example, deleting the folder .dll is located in, hiding a folder under junctions, etc)

COM Hijacking

Although simple to detect (as there aren't many .com files on Windows), COM Hijacking is till a powerful methodology, dangerously powerful as it acts the same way Task Scheduler does (it can launch cheats on startup or upon a specific event occurs), for example, bypassers can set a "malware" (cheat) to launch every time they open Incognito or any specific MS Office file. What's making it worse is that this mixed with UAC Hijacking and a cheat that doesn't show a GUI or requires hotkeys to launch the GUI - can increasingly confuse any ScreenSharer and any SS Tool if it's also mixed with an ADS bypass (considering they can also remove the ADS from a system without accessing external utilities)



COM Hijacking blog

Extensionless Files

The most known types of a file used in bypasses are either renamed extensions, or no extensions at all! Extensionless files are smartly usable, as Bypassers can fake them to seem like they're folders (by using hidden characters), launch them using WMIC or because every ScreenSharer filters .exe / .dll or other extensions in the SS - avoid getting caught by any SSer if they forget extensionless files exist.

Shellcode Injection

As the name suggests, Shellcode Injection refers to injecting a “malicious” code into a process or compiling executables with hidden shellcode in them, this methodology is often related to Linux rather than Windows (mainly since detecting it on Windows is pretty simple), Bypassers may utilize VSCode to initiate local & remote shellcode injections, that sentence is scary on its own when we imagine someone actually knowledgeable doing it... in Minecraft and FiveM communities, shellcode is still an executed file that loads a malicious code upon launching it.

Powershell Remoting

Enabling powershell remoting only works on private networks or domains. With Powershell Remoting, users can send script blocks to target devices designated to initiate specific commands, execute files on that device or more. Typically, this level of remote access on a device isn’t scary, but it can be used to tamper with an ongoing ScreenShare, change evidences of execution, launch any stealth applications, close SS Tools or hijack them and so on - if the SSer has no idea this is a possibility.

HTA exploit

HTML can be used to execute files on the local device. You heard that right. There are multiple ways of doing so:

Example 1

Example 2

Example 3

They all work the same way and do the same process. This exploit is pretty old and not too hard to detect.

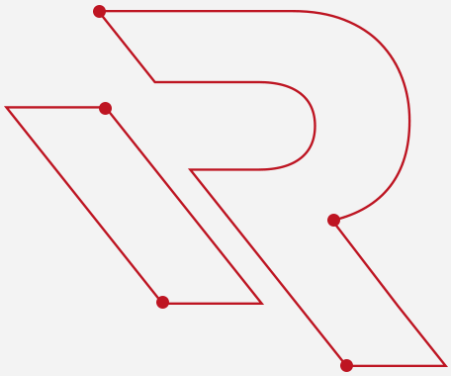
Shellcode Exploit

A more complicated Shellcode “injection” exploit has been found very recently for Win11; users are able to download and execute files. “Download and Execute”, they can also delete it.

Exploits

Most recent and consistent exploits for Win11 can be found at SearchSploit with step-by-step guides and tutorials.

Red Team PL



And well, for powershell payloads, IOCs and so on you can find some information at redteam.pl

Chapter VIII

Blue Team DFIR🧿

Digital Forensics

Blue Teamers counter the Red Teamers in the field of Digital Forensics, just like ScreenSharers and Bypassers. We, ScreenSharers, are the Blue Team. Red Lotus has developed ScreenShare practices relying on public DFIR utilities, and we have also developed our own approaches to cheats / bypasses spread over in the Community. Thus, before covering all the never-before-seen bypasses and Red Team anti-forensics, let's begin with Universal methods for detecting malicious files regardless of what bypass or execution was used!
For that, we'll be using a lot of DFIR sources, utilities and approaches to general malware detections.



Here are the repositories to begin with:
https://github.com/mesquidar/ForensicsTools#live-forensics
https://academy.cyber5w.com/courses/
All Resources - Magnet Forensics
AndrewRathbun/DFIRMindMaps
AndrewRathbun/DFIRArtifactMuseum
https://dfirmadness.com/
windows-forensic-artifacts/execution
https://github.com/mikeroyal/Open-Source-Security-Guide

The following Chapter will cover necessary logic to read every relevant artifact.

Activities Cache

Introduction to Activities Cache

General Understanding

When installing Windows, users are met with security options they choose to enable or have disabled, disabling options that suggest your data tracking will install windows with services like Diagtrack disabled and Activities Cache encrypted.

“Windows activity history keeps track of activity on a device, such as application and services usage, files opened, and websites browsed.” - [Source]

The service and the ‘ActivitiesCache.db’ database existed before the 1803 upgrade (May 2018), but with limited functionality. Another possibly related* activity store location is at:‘Software\Microsoft\Windows\CurrentVersion\CloudStore\Store\Cache\DefaultAccount\$$windows.data.task flow.shell activities\Current’; which means that we are able to access this Artifact starting from Windows 10 Builds before and after 1803.

Win+R & msinfo32 will display what version the user is currently on.

Item	Value
OS Name	Microsoft Windows 11 Home Single Language
Version	10.0.22631 Build 22631

On these versions, let’s presume we want to use Registry data to read user activity. While examining a Twitter post to analyze the given artifact, user @gazambelli has mentioned:

“So far I've seen that \xC6\x32 can be used to separate each artifact. The pattern is always: AbsolutePathToExecutable + Executable + Title. The title can be the title of the web page that was viewed or the folder name that was explored with explorer.exe.”

Further examining the given registry entry, Red Lotus has concluded that following this method in an actual ScreenShare is far from necessary, as there are dangers of being exposed with user emails. Therefore, to analyze Activities Cache on a device with Diagtrack active (sometimes, activities can be encrypted even if diagtrack is turned on), we recommend using WxTCmd.exe by Eric Zimmerman. | Introduction to WxTCmd

- Cite (Windows Timeline 18)

AMCache

Introduction to AMCache

General Understanding

Amcache is one of the most “useless” Artifacts for ScreenSharing if we want to prove date of execution rather than determine whether or not files were installed on the device. In other words, if we for some reason want to determine installation of applications on Windows 10, we may refer to this artifact. If we want to view Recent files on Windows 7, we can also refer to this artifact!
SS Community has already demonstrated its usage back in 2022 for ScreenSharing. I’ll cite their learning material.
1)[https://f001.backblazeb2.com/file/EricZimmermanTools/AmcacheParser.zip]
2)[https://f001.backblazeb2.com/file/EricZimmermanTools/TimelineExplorer.zip]

- Video Demonstration

The default directory to Amcache is `C:\Windows\AppCompat\Programs\Amcache.hve`
An alternative to Amcache for Windows 7 is syscache.hve - a file inaccessible without using FTK Imager and visiting System Volume Information. SS Community has created Ruedas to analyze just that with a Video Demonstration.

Another alternative to both Ruedas and AMCache is RecentFileCache.bcf - present on Windows 7 and Windows 10 at `C:\Windows\AppCompat\Programs\`
https://f001.backblazeb2.com/file/EricZimmermanTools/RecentFileCacheParser.zip
`RecentFileCacheParser.exe -f C:\Windows\AppCompat\Programs\RecentFileCache.bcf --csv .` or, because RecentFileCache may be in use, copy it to a different Directory (for example, downloads) and parse as shown in the clip.

And of course, Amcache is present on Windows 8 as well.
If you’re working on a Python project, feel free to refer to the Amcache parser in PY.

Extended comprehension: anssi-coriin_2019-analysis_amcache

Windows Tasks

Introduction to Task Scheduler

General Understanding

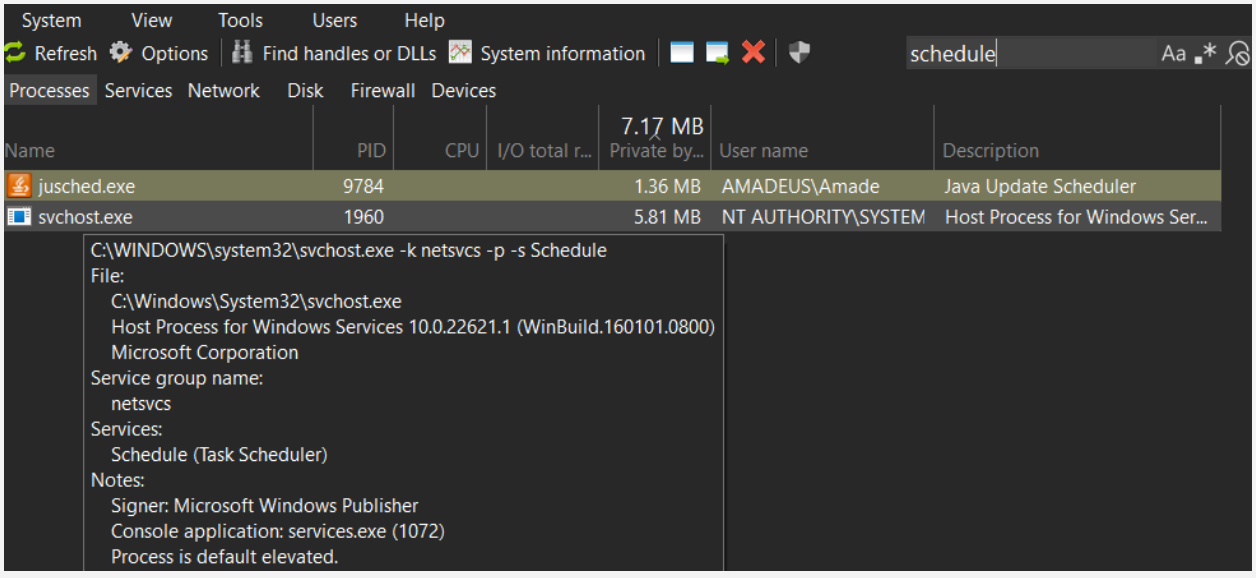
Information given in this section is purely developed, studied and comprehended by Red Lotus, thus this is a purely original approach to detecting all Scheduler related bypasses. To support the understanding of this approach, it is highly recommended to learn environment variables from the two provided sources: 1 | 2 > Why is this important? In practice, knowing what environment variables are and how they function will help you understand the whole logic of the following methodologies more easily.

Precautions

To properly conduct the ScreenShare with this approach, you shall manage to do step 1 of this section within the first 15 minutes, you'll need System Informer and Journal-Tool

Step I

After you're done installing System Informer and before you download Journal-Tool, immediately enable the Kernel-Mode driver & filter "Schedule" Service for <Command> first, copy selected results to one Notepad and re-filter for <Action - copy these results to a second Notepad. It is necessary to do this step within the first 15 minutes at least, considering a lot of data can be lost from memory very quickly.



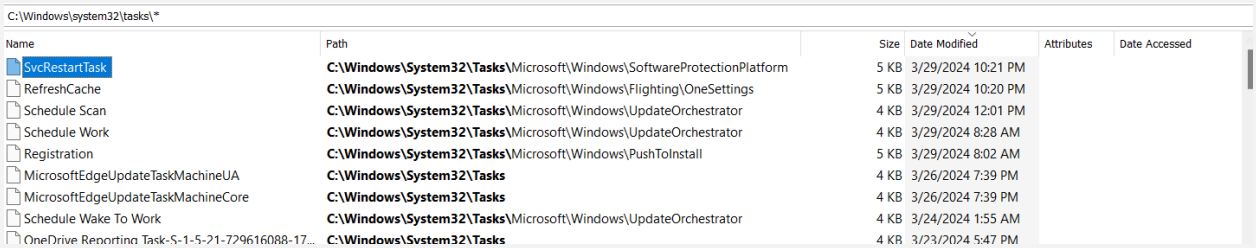
Step II

Download Journal-Tool and filter Deletions with the search of "C:\Windows\System32\Tasks" - this will show deleted tasks. If a task file is deleted before ScreenShare, search its name in both Notepads. The <Commands> results will display what the scheduled task launched on startup - or, whether or not it is a Windows generated task. How do we know if it's a Windows generated task? its path will be one of the environment variables instead of a drive letter!

Feel free to search files found in these two logs using Search Everything (or Journal Tool itself to determine whether or not they're deleted);

Step III

Using Search Everything, filter in search "C:\Windows\system32\Tasks*" and sort by "Date Modified" - this way you'll see which tasks were modified before ScreenShare.



Keep in mind that SvRestartTask often updates its date of Modification, however, it is still recommended to search all of the modified tasks before SS in <Actions notepad as the Bypasser might have edited out their tasks.

Step IV

If you see C:\Windows\System32\cmd.exe (or cmd.exe scheduled from any other directory) and considering you've kept System Informer up without refreshing it this whole time (as by refreshing all stored data in services will be erased after around 15 minutes) you can either ban for scheduled CMD (Bypass Attempt) or attempt detecting cheats on the device. Keep in mind that because CMD is scheduled, any cheat trace in DPS, Csrss, SearchIndexer or even Explorer and PCasvc (consider their uptime - of explorer and pca only - is under 12hrs) is justifiable to ban for.

Step V

What else can you do? What more is there to know about Scheduler besides all known Registry Explorer methods through TaskCache/TaskTree? - well of course. There's more. This is a method explained in specific Red Lotus Webinars without posting it elsewhere or repeating anywhere. For this method, you'll only need bstrings and baca's signature checker.

Use bstrings.exe and run the following command first (keep in mind that you'll have to save <Command and <Action notepads somewhere as we'll be needing them):

bstrings.exe -d C:\Windows\System32\Tasks --ls :\ -o .\tasks3.txt

Save the <Command> and <Action Notepads as tasks1 and tasks2 in the same directory with tasks3. Keep that directory outside of where bstrings is (for example, create a new folder and keep all of them there)

Then, using bstrings bulk parse the folder you've saved all extractions in, you can run the command one by one for <Command and <Action yet again! for example:

bstrings.exe -d "Mydirectory" --ls Command -o .\commands.txt

Keep in mind to not include the "<" character while parsing with bstrings itself. In the commands.txt notepad, replace all <Command> and </Command> words as blanks (spaces) to separate only file paths.

Do this again for Action instead of Command and copy all results into one notepad. Name it "paths.txt". Now you have all present and removed paths ready to be scanned with baca's signature checker. Run the signature checker in the results folder and analyze results. Of course, it would be cheating to tell you what to look for next, wouldn't it?!

Step VI

For this final step, if System Informer doesn't work or you want to overdo the <Command or <Action checks, you can use MagnetProcessCapture to extract all service results and parse with bstrings for Action or Command! This will log far more data and results will be more promising!

Step VII

And of course, to cite SS Community (Kendo) for their first ever method for Scheduler detections - Registry Explorer can also be used to detect scheduled task deletions, task owners, even unicode character tasks that can bypass System Informer!

- Imgur Album of the method:

EVTX

EVTX is an eventlog file extension. Windows event log sources (Event Viewer sources) are located in `C:\Windows\System32\winevt\Logs\` thus if we were to go by the previous logic, we can use Search Everything to copy all .evtx files into a specific folder then scan it using bstrings for all odd looking file logs it might have ever logged but that'd be too boring to demonstrate here!

Thus, I believe that you are a very enthusiastic reader who will learn evtx enumerations first!

The SANS Institute has released a sheet defining EVTX Property logs:

EVTX Property logs

Source:	Thesoftwarethatloggedtheevent,whichcanbeeitherap rogramname, suchas"SQLServer",oracomponentofthesystemorofal arge program,suchasadrivername.Forexample,"Elnkii"indic atesan EtherLinkIIdriver.
Event ID	Anumberidentifyingtheparticulareventtype.Thefirstlin eofthe descriptionusuallycontainsthenameoftheeventtype.Fo rexample, 6005istheIDoftheeventthatoccurswhentheEventLogs erviceis started.Thefirstlineofthedescriptionofsuchaneventis" TheEventlog servicewasstarted."TheEventIDandtheSourcecanbeus edby productsupportrepresentativestotroubleshootsystem problems.
Level	Aclassificationoftheeventseverity.Thefollowingevents everitylevels canoccurinthesystemandapplicationlogs: Information.Indicatesthatachangeinanapplicationorco mponenthas occurred,suchasanoperationhassuccessfullycomplete d,aresource hasbeencreated,oraservicestarted.

	Warning.Indicatesthatanissuehasoccurredthatcanimpactserviceor resultinamoreseriousproblemifactionisnottaken. Error.Indicatesthataproblemhasoccurred,whichmightimpact functionalitythatisexternaltotheapplicationorcomponentthattriggered theevent. Critical.Indicatesthatafailurehasoccurredfromwhichtheapplication orcomponentthattriggeredtheeventcannotautomaticallyrecover. Thefollowingeventseveritylevelscanoccurinthesecuritylog: SuccessAudit.Indicatesthattheexerciseofauserrighthas succeeded. FailureAudit.Indicatesthattheexerciseofauserrighthas failed. IntheEventViewernormallistview,thesearerepresentedbyasymbol.
Log	Thenameofthelogwheretheeventwasrecorded.
Computer	Thenameofthecomputeronwhichtheeventoccurred.Thecomputer nameistypicallythenameofthelocalcomputer,butitmightbethename ofacomputerthatforwardedtheeventoritmightbethenameofthelocal computerbeforeitsnamewaschanged.
ProcessorTime	Theelapsedexecutiontimeforuser-modeinstructions,inCPUticks.
UserTime	Theelapsedexecutiontimeforuser-modeinstructions,inCPUtimeunits.
CorrelationId	Identifiestheactivityintheprocessforwhichtheeventisinvolved.This identifierisusedtospecifysimplerelationshipsbetweenevents.

ProcessID	The identificationnumberfortheprocessthatgeneratedtheeevent.
Thread ID	Theidentificationnumberforthethreadthatgeneratedtheeevent.

Event IDs

[Randy's Windows Security Log Encyclopedia (ultimatewindowssecurity.com)]

Useful EventLog Tools

[FullEventLogView]

[Alternative to standard event viewer of Windows]

[Event Log Explorer]

ELEX (Event Log Explorer) can be used to monitor logged events in live instances.

Usage

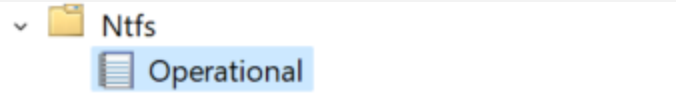
Windows Event Logs can be used to determine when users Logged on, whether or not their Explorer crashed, for example, also to check PCA logs, Journal Deletion, Time change, event log deletions, USB logs, virtual mount logs and so on.

Step I - User Activities

Using event ID 4626 we can determine whether or not the current user configured file, registry or folder permissions onto themselves. ID 4647 is user Logoff time, IDs 4704 and 4705 refer to assigning or revoking user rights; ban evaders may use the same device with different accounts, this can be caught using the event ID 4720 and 4722, to trick SSers they can also disable them - shown in ID 4725 and 4726 shows account deletion. 4738 refers to account change.

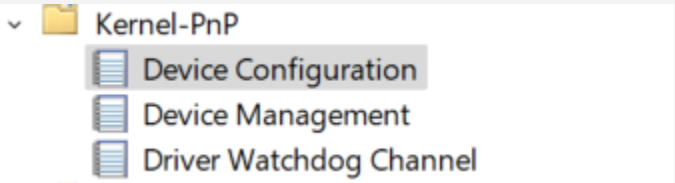
This can be useful to target ban evaders or alt accounts who use multiple Windows users to evade. For servers that don't freeze ingame, it could be useful to know user activity event IDs.

Step II - NTFS Operational



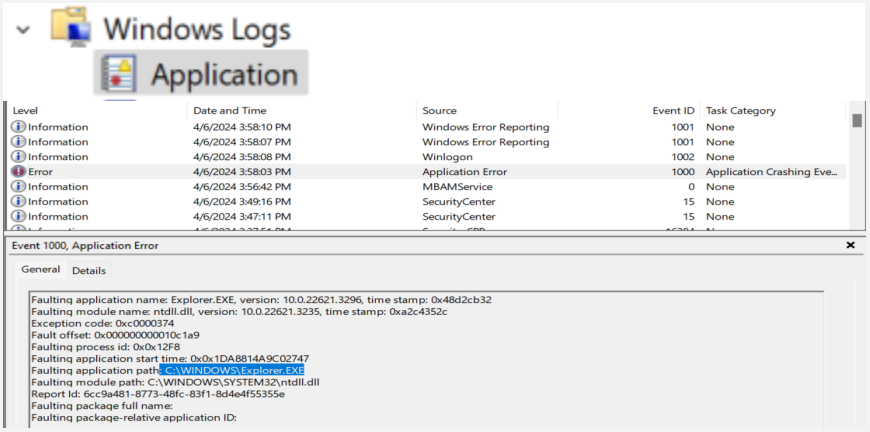
Using event ID 4 on NTFS-Operational event logs, we can detect mounted and dismounted drives. Relevant IDs are also: 9, 10, 158, 147, 149. In general, check every event ID on a specific date. This type of event data can also be used as a replacement for HWID marks, logging disk IDs are more specific and useful than HWIDs.

Step III - Kernel-PnP



Event IDs - 400, 411, 410, 430 in Kernel-PnP sect. Management/Config display generally logged data regarding USB devices, including Virtual Mounts, dismounted or mounted drives and so on. Check relevant date events for further intel. Main channels are Device Configuration and Device Management. Dismounted USBs will be logged as “surprisingly dismounted”.

Step IV - Application IDs



Under Windows Logs - Application it is possible to gather and check a lot of information. For example, if we would like to confirm whether or not a player lies while saying their Explorer crashed, it is possible to check that by analyzing a followup chain of IDs 1000, 1001, 1002, these IDs correspond to an app hang - app crash and app reload instances. They mainly always follow one another if an app crash was actual. However, it is worth noting that even if someone’s Explorer crashed but you find they’ve accessed Process Hacker 2 / System Informer during the day at any given instance, it is safe to ban for it, considering usually app hangs are caused (although by analyzing data in these event Ids you can check so what caused the hang) by [nullified bytes.]

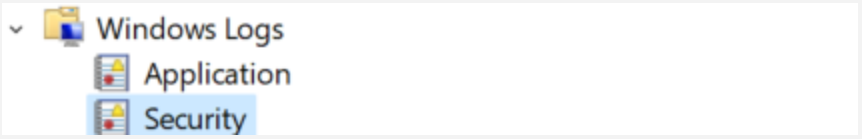
On the other hand, event ID 0 shows which service was stopped.

Event ID 1 shows which service was started.

Event IDs 1035 and 1040 show MSI installations (some cheats are .msi based)

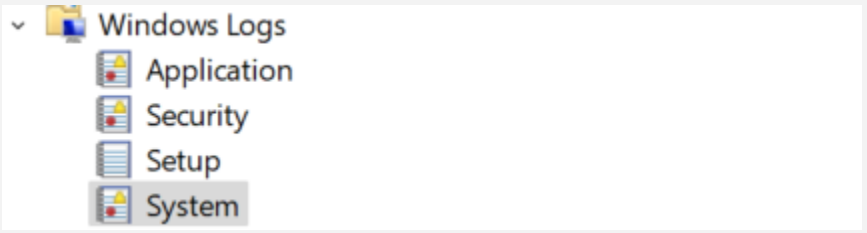
Event ID 1023 shows NET runtime errors.

Step V - Security



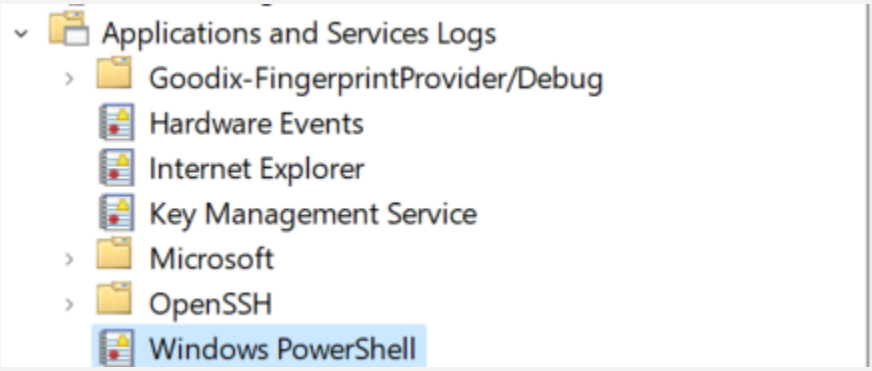
Security Event logs relate to user activities on the device. It can show User Logins 4624, special logon 4672 (with custom privileges), most importantly - system time change 4616. As well as ID 1102 - Event Log clear, which means that an event log was cleared by the user. This ID task often also shows which event Log was cleared. Bannable offense.

Step VI - System



The System Event log can be used to determine when users closed System Informer or Process Hacker (ID 1) when they loaded it (ID6) or check if users modified service start settings, for example, of Sysmain (ID7040) or HTTPService logs (ID112) or device shutdown (ID1074) so on and so forth! seems to be the most useful channel so far, right?

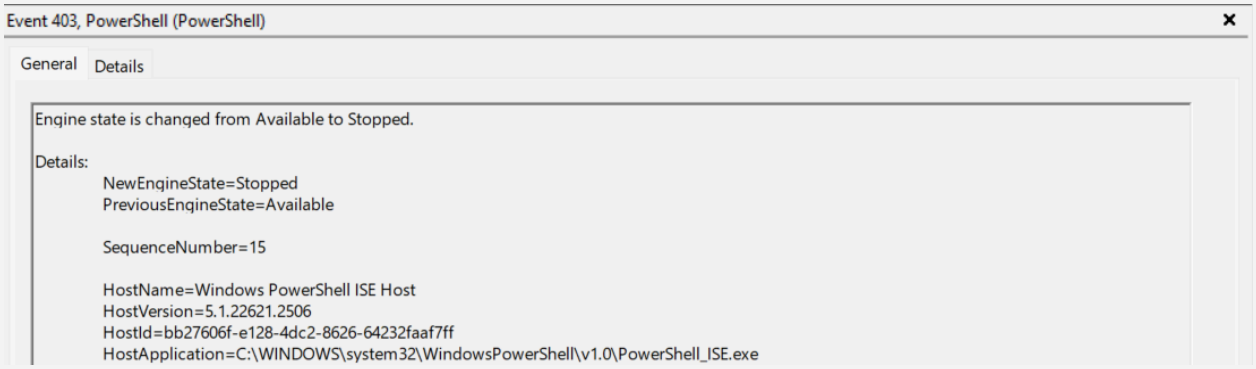
Step VII - Windows PowerShell



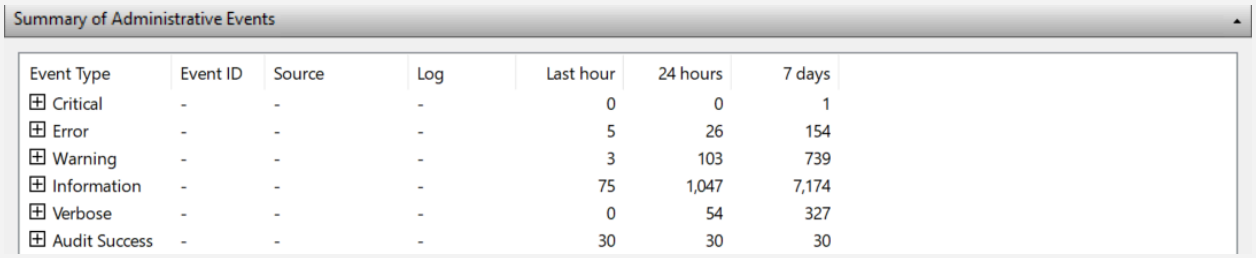
Under Application and Services logs, Windows PowerShell logs powershell commands from both PS and Powershell ISE (which does not generate psreadline logs unlike the regular powershell), however, we would need to learn to differenciate between user initiated commands and automated scripts such as these on AMD for example: *HostApplication=powershell.exe -ExecutionPolicy Restricted -Command $Res = 0; $Infs = Get-Item -Path ($env:WinDir + '\inf*.inf'); foreach ($Inf in $Infs) { $Data = Get-Content $Inf.FullName; if ($Data -match '\[defaultinstall.nt(amd64|arm|arm64|x86)\]') { $Res = 1; break; } } Write-Host 'Final result:', $Res;*

It is an automated AMD script, I've highlighted relevant info that tells us that above. So, how do we get around with this? Well, Event ID 403 in Powershell shows if the terminal

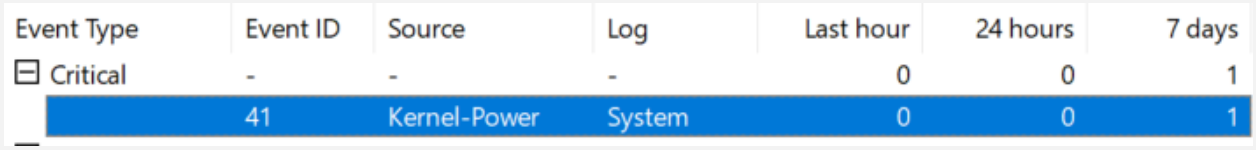
was even launched! Thus any command or script after 403 is most likely user executed. After all, you can just copy the command and re-run it to see what it does.



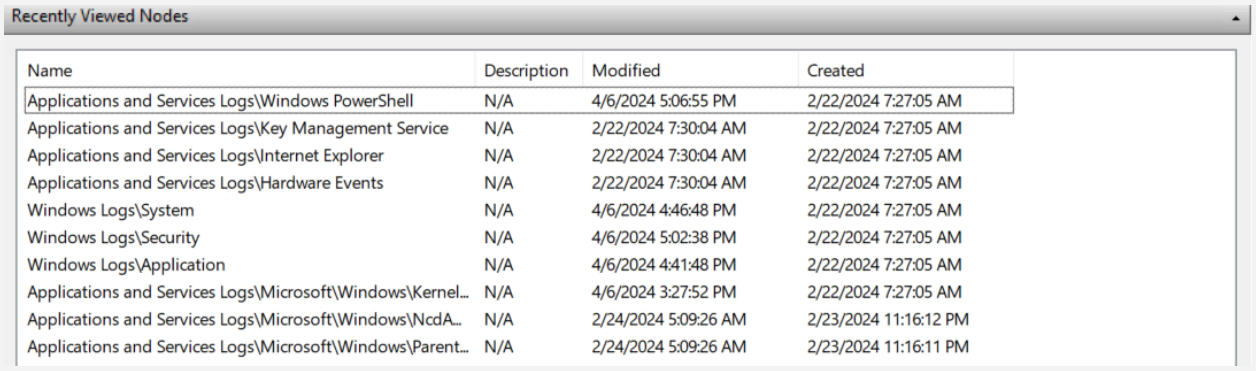
Step VIII - Summary of Administrative Events



At last to not dive too deep into every single event log, we can generally view every severity of Event IDs under Summaries. Enrolling the Event Types will display which channels have updated, with Event ID, Source, Log and time span displayed next to it.

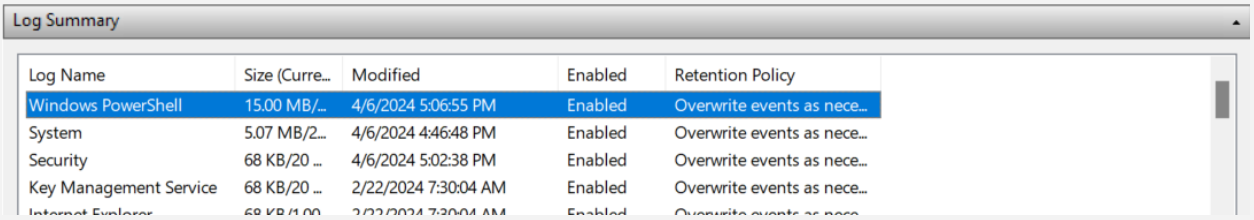


Step VIV - Recently Viewed Nodes

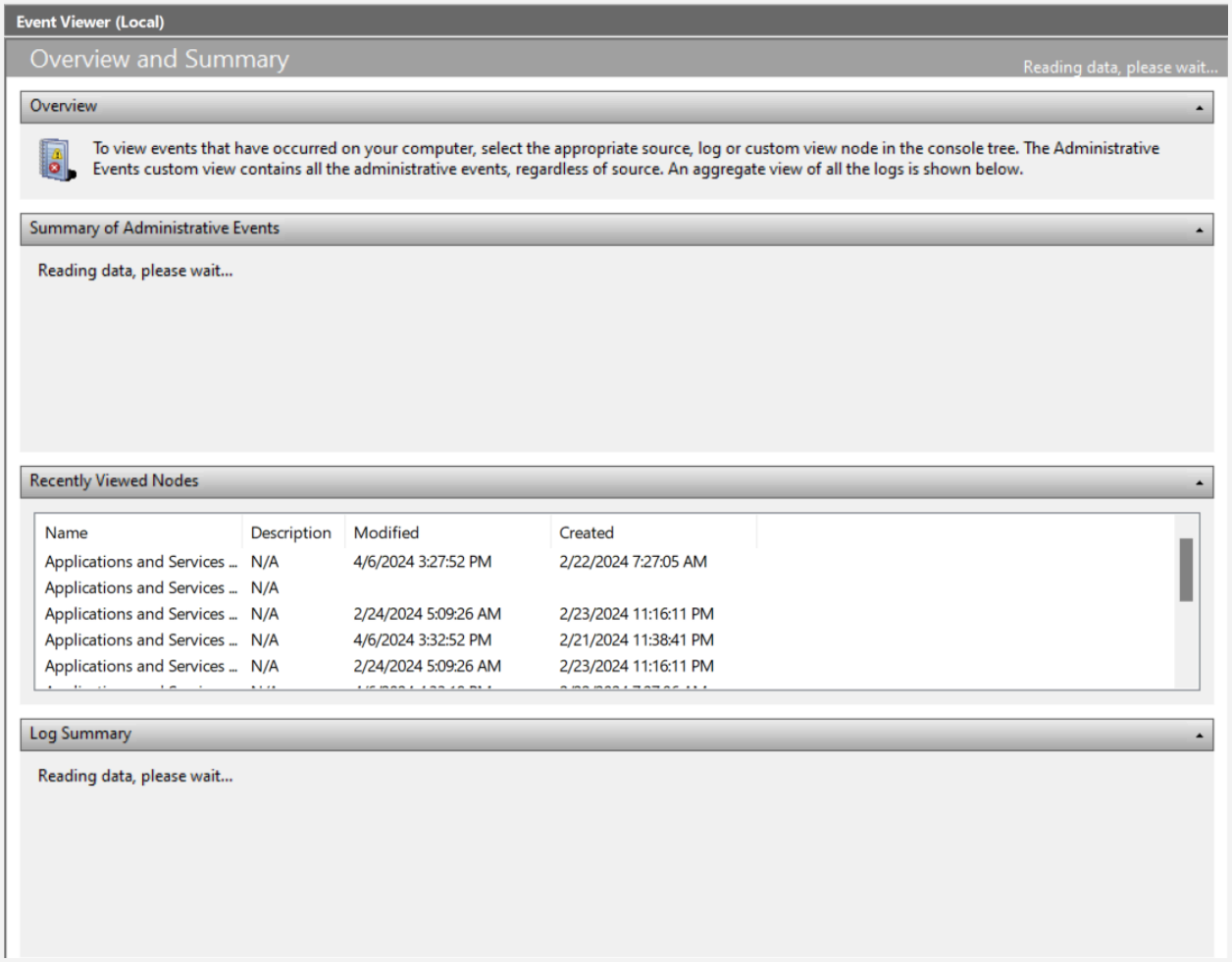


With this, if users use Event Viewer itself to clear logs, we will be able to determine which nodes they accessed recently. It logs user accessed Nodes in session, with dates of modification presenting last access / visit times.

Step IX - Log Summary



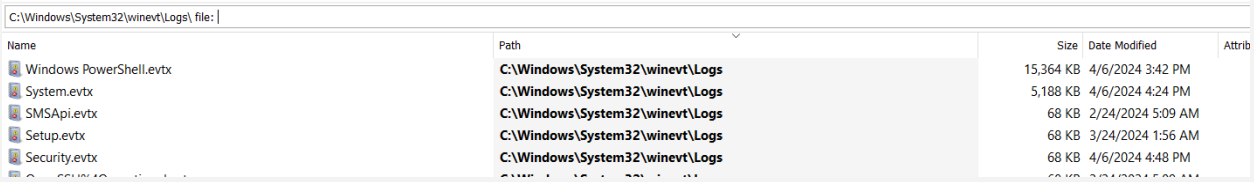
By far the most useful section in Event Viewer. It displays Logs with Date Modified (new event IDs being logged) their state (enabled / disabled inheritance) with the possibility to double click on them, which takes us to the selected node to analyze them. For example here, if we double click on Windows PowerShell we will be taken to the PS Event Logs themselves! All of this is loaded when launching the event viewer.



If loading the event viewer returns an error, open CMD and query eventlog service itself by initiating “sc query eventlog”; it is bannable to have this service disabled at all cost.

Step X - Parsing with Bstrings

As mentioned above, we are able to parse .evtx files using Bstrings. On Windows 11 they are located in C:\Windows\System32\winevt\Logs\ - thus we have the option to use search everything and copy all of these channels into a dedicated folder, for example, C:\SS1 to then automatically scan them with bstrings for, let's say, service checks or all loaded file path checks!



In search everything - C:\Windows\System32\winevt\Logs\ file: then, you can CTRL+A Ctrl+C & paste it (CTRL+V) in the C:\SS1 folder to avoid Admin privilege overlaps. Follow this demonstration video for a mock check of loaded file paths in event logs! alternatively, you can use Powershell itself, cd to C:\SS1 and initiate command findstr /C:".exe" * | Out-Gridview for technically the same results, although this might be bit more confusing to do.

Get Bstrings here

Demonstration Video - Bstrings

Demonstration Video - Powershell Findstr

. . .

Step XI - Hayabusa

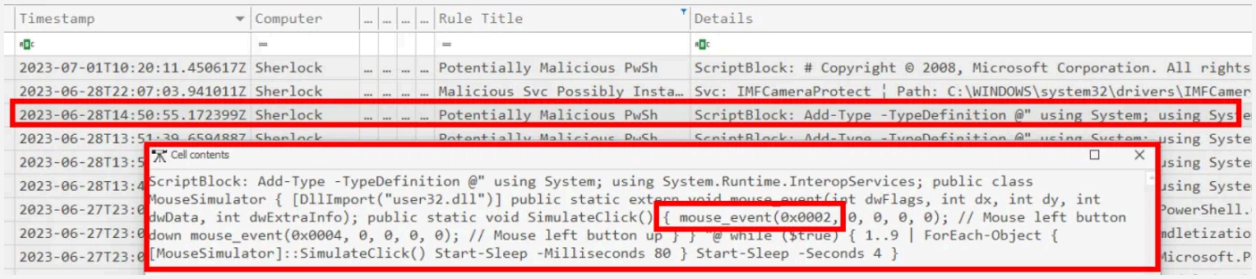


Hayabusa is an event log parser developed by YAMATO Security, Red Lotus has already held webinars in regards to explaining this utility and its use in General, however, we have never mentioned it is capable of working on [Linux with Timesketch] as well. Besides that, here's how it works on Windows:

[Download Here]

Select the Hayabusa build (I'm on `hayabusa-2.6.0-win-x64` in this case) and launch the following command: `hayabusa-2.6.0-win-x64.exe csv-timeline --output CSVOutput.csv -d C:\Windows\System32\winevt\Logs --HTML-report HTMLOutput.html --ISO-8601`

Enable log parsing without restrictions, HTMLOutput.html will show general information regarding detected traces, however, CSVOutput.csv is the data we'll read using Timeline Explorer to look for whatever we need. We'll be able to analyze specific rules by modifying "Rule Title". For example here, we've just found an auto clicker powershell script by analyzing Malicious PWSH.



Here's a timeline explorer filter to automatically load useful rules:

[Rule Title] In ('Alternate PowerShell Hosts - PowerShell Module', 'Change PowerShell Policies to an Insecure Level - PowerShell', 'Defender Alert (Severe)', 'Logon (System) - Bootup', 'Logon (Unlock)', 'Malicious Svc Possibly Installed', 'Possible Hidden Shellcode', 'Potentially Malicious PwSh', 'Powershell Create Scheduled Task', 'Powershell File and Directory Discovery', 'Proc Exec', 'Suspicious Service Installation', 'Usage Of Web Request Commands And Cmdlets - ScriptBlock', 'VHD Mounted', 'VHDX Mounted', 'Windows Defender Malware Detection History Deletion', 'Windows Defender Threat Detection Disabled', 'Windows Service Terminated With Error')

[Here's an extended information regarding Timeline Explorer and how filters work;]

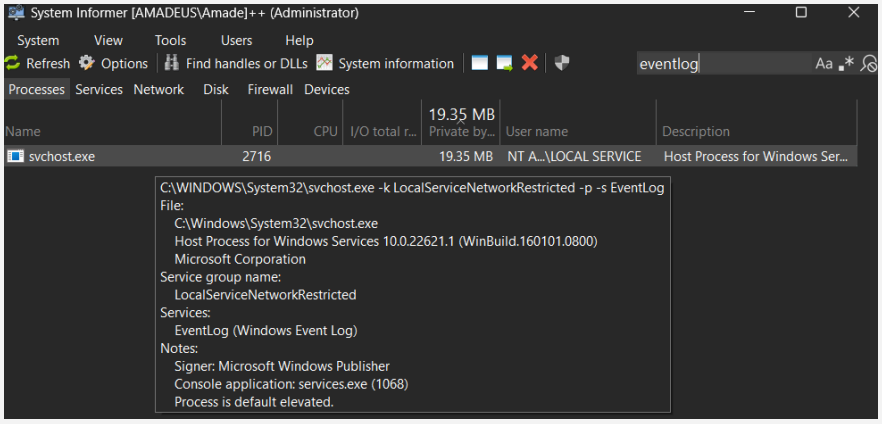
Step XII - Chainsaw

Yet another event log parser, older and more versatile in a sense of modifiability - Chainsaw can allow users to create their own Hunt rules, modify the way they're sorted and so on. How does it differ from Hayabusa? What are its advantages? well, there's not much difference besides their purposes. Chainsaw has powerful threat hunting modules and Sigma rules (also modifiable by users) that orient on experience of DFIR specialists while hunting anomalies on the device. It is able to detect the Word bypass, for example (as MS word logs in Event Logs, both Hayabusa and Chainsaw are capable of catching odd behavior) - Here are two Demos of the utility;

Demo 1

Demo 2

Step XIII - Eventlog Service



Finally, Event Log is a service on Windows. Thus you can simply filter specific checks immediately while waiting for let's say Echo or Ocean to finish scanning, you can check its uptime in General properties and experiment with detections. For example, if you filter "warning" in this service, you'll get Warning severity flags right away. Of course, previous steps are recommended. It is also recommended you test these out on your own Device, especially for VBA bypasses themselves if you have MS Word... Did I forget to mention that the VBA script logs in WinPrefetchview under WORD/MSWORD itself too? Hmm...

Step XIV - Eventlog AutoScans

If all of these steps are too much and you just want to detect specific IDs on the device, you can use Powershell and create a script. For example, let's take this line in a code that checks for Event IDs:

```
$EventIDs = @(3079, 1102, 2003, 7, 7036, 7045, 1001, 1002, 104, 1074, 4663, 7036, 4688, 4660, 4616, 11707)
$EventLogs = @("Application", "System", "Security", "Windows PowerShell", "Windows PowerShell Operational", "Setup", "ForwardedEvents", "Internet Explorer")
```

In $EventIDs you may replace specific numbers yourself (here for example, 3079 is a USN Journal clearance check in $EventLogs, which means it checks every specified Event Log name for the Event ID thus the Powershell will print a lot of errors before finding 3079 in its proper log); So the second part of this script would be something like:

```
$Results = @()

foreach ($EventLog in $EventLogs) {
    $Events = foreach ($EventID in $EventIDs) {
        Get-EventLog -LogName $EventLog -InstanceId $EventID -Newest 10
    }
    $Results += $Events
}

$Results | Export-Csv -Path ".\InstantParse.txt"
```

Which means that for every EventID detected in EventLogs it will scan the Newest 10 entries of that ID and print that in InstantParse.txt file after the scan is finished. The scan is finished when Powershell is done printing Errors (it prints Errors in the terminal to avoid being printed in the .txt results itself); The finished script looks like this - modify as you wish:

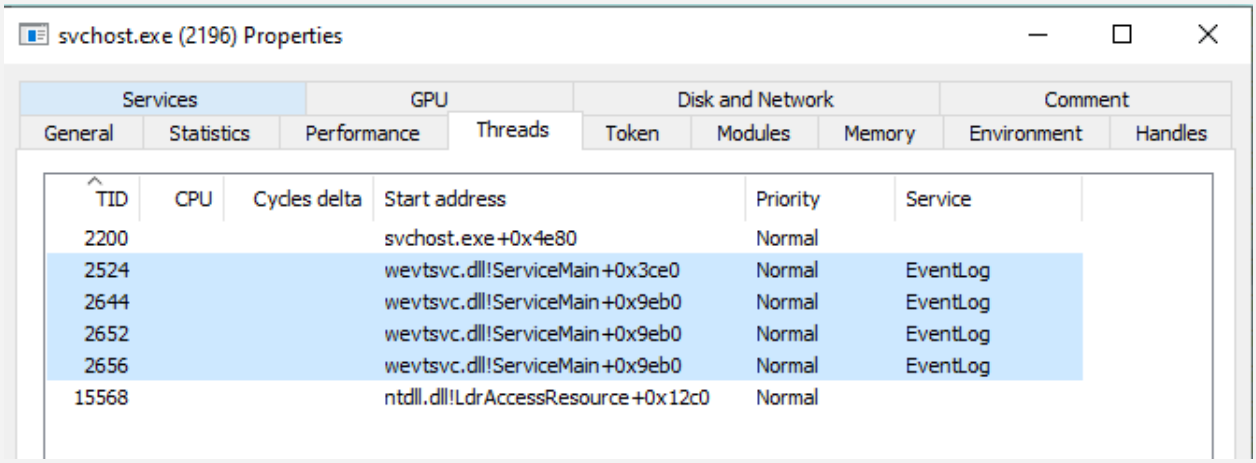
```
$EventIDs = @(3079, 1102, 2003, 7, 7036, 7045, 1001, 1002, 104, 1074, 4663, 7036, 4688, 4660, 4616, 11707)
$EventLogs = @("Application", "System", "Security", "Windows PowerShell", "Windows PowerShell Operational", "Setup", "ForwardedEvents", "Internet Explorer")
$Results = @()

foreach ($EventLog in $EventLogs) {
    $Events = foreach ($EventID in $EventIDs) {
        Get-EventLog -LogName $EventLog -InstanceId $EventID -Newest 10
    }
    $Results += $Events
}

$Results | Export-Csv -Path ".\InstantParse.txt"
```

Step XV - Hunting for EventLog Anti-Forensics

Bypassing Event Logs is possible by suspending necessary Threads using Process Hacker, System Informer or Powershell itself. Here are the threads necessary for Event Logs to properly function:



Here is an automated compilable code that can be used to suspend these Threads in the Eventlog service:

```
#include <iostream>
#include <Windows.h>
#include <Psapi.h>
#include <TlHelp32.h>
#include <dbghelp.h>
#include <winternl.h>
#pragma comment(lib, "DbgHelp")
using myNtQueryInformationThread = NTSTATUS(NTAPI*)(
		IN HANDLE		ThreadHandle,
		IN THREADINFOCLASS ThreadInformationClass,
		OUT PVOID		ThreadInformation,
		IN ULONG		ThreadInformationLength,
		OUT PULONG		ReturnLength
		);
int main()
{
		HANDLE serviceProcessHandle;
		HANDLE snapshotHandle;
		HANDLE threadHandle;
		HMODULE modules[256] = {};
		SIZE_T modulesSize = sizeof(modules);
		DWORD modulesSizeNeeded = 0;
		DWORD moduleNameSize = 0;
		SIZE_T modulesCount = 0;
		WCHAR remoteModuleName[128] = {};
		HMODULE serviceModule = NULL;
		MODULEINFO serviceModuleInfo = {};
		DWORD_PTR threadStartAddress = 0;
		DWORD bytesNeeded = 0;
```

```
	myNtQueryInformationThread NtQueryInformationThread =
(myNtQueryInformationThread)(GetProcAddress(GetModuleHandleA("ntdll"), "NtQueryInformationThread"));
	THREADENTRY32 threadEntry;
	threadEntry.dwSize = sizeof(THREADENTRY32);
	SC_HANDLE sc = OpenSCManagerA(".", NULL, MAXIMUM_ALLOWED);
	SC_HANDLE service = OpenServiceA(sc, "EventLog", MAXIMUM_ALLOWED);
	SERVICE_STATUS_PROCESS serviceStatusProcess = {};
	# Get PID of svchost.exe that hosts EventLog service
	QueryServiceStatusEx(service, SC_STATUS_PROCESS_INFO, (LPBYTE)&serviceStatusProcess, sizeof(serviceStatusProcess),
&bytesNeeded);
	DWORD servicePID = serviceStatusProcess.dwProcessId;
	serviceProcessHandle = OpenProcess(MAXIMUM_ALLOWED, FALSE, servicePID);
	snapshotHandle = CreateToolhelp32Snapshot(TH32CS_SNAPTHREAD, 0);
	EnumProcessModules(serviceProcessHandle, modules, modulesSize, &modulesSizeNeeded);
	modulesCount = modulesSizeNeeded / sizeof(HMODULE);
	for (size_t i = 0; i < modulesCount; i++)
	{
		serviceModule = modules[i];
		# Get loaded module's name
		GetModuleBaseName(serviceProcessHandle, serviceModule, remoteModuleName, sizeof(remoteModuleName));
		if (wcscmp(remoteModuleName, L"wevtsvc.dll") == 0)
		{
			printf("Windows EventLog module %S at %p\n\n", remoteModuleName, serviceModule);
			GetModuleInformation(serviceProcessHandle, serviceModule, &serviceModuleInfo,
sizeof(MODULEINFO));
		}
	}
	# Enumerate threads
	Thread32First(snapshotHandle, &threadEntry);
	while (Thread32Next(snapshotHandle, &threadEntry))
	{
		if (threadEntry.th32OwnerProcessID == servicePID)
		{
			threadHandle = OpenThread(MAXIMUM_ALLOWED, FALSE, threadEntry.th32ThreadID);
			NtQueryInformationThread(threadHandle, (THREADINFOCLASS)0x9, &threadStartAddress,
sizeof(DWORD_PTR), NULL);

			# Check if thread's start address is inside wevtsvc.dll memory range
			if (threadStartAddress >= (DWORD_PTR)serviceModuleInfo.lpBaseOfDll && threadStartAddress <=
(DWORD_PTR)serviceModuleInfo.lpBaseOfDll + serviceModuleInfo.SizeOfImage)
			{
				printf("Suspending EventLog thread %d with start address %p\n", threadEntry.th32ThreadID,
threadStartAddress);
				# Suspend EventLog service thread
				SuspendThread(threadHandle);
				Sleep(2000);
			}
		}
	}
	return 0;
}
```

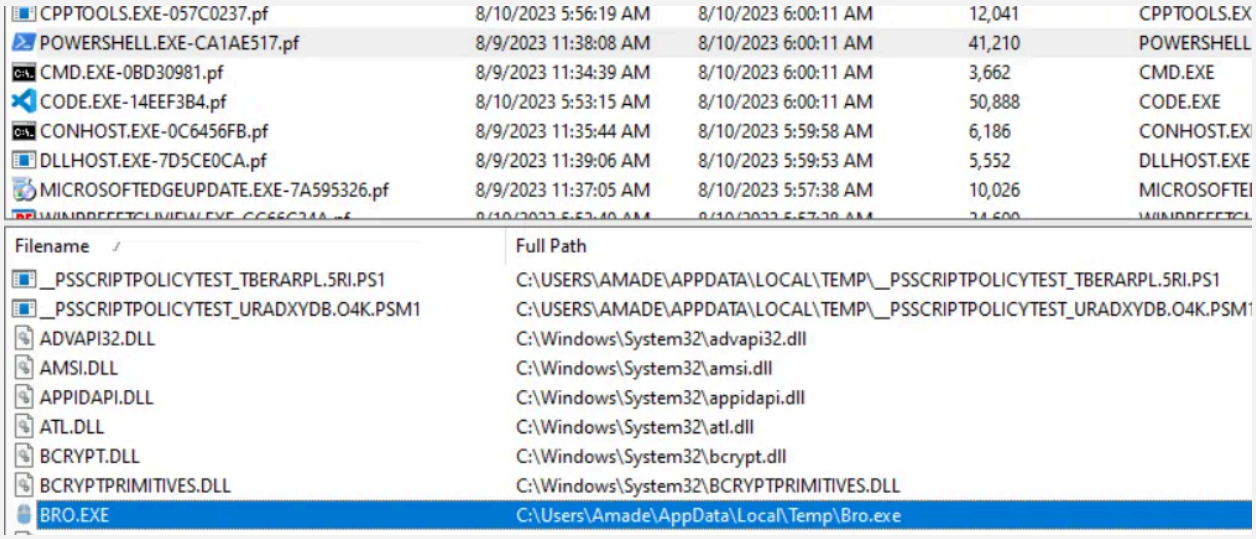
Of course, ScreenSharers can ban cheaters for having these specific Threads suspended or this code initiated separately (using VSCode, for example);

VSCode

Introduction to Scripting Environments

General Understanding

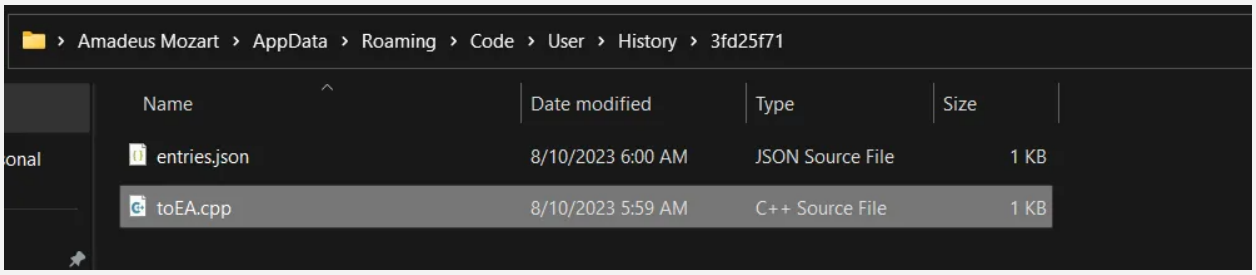
Many cheaters use VSCode itself to run scripts that load Cheats or clear Logs, suspend Threads and so on. They also use Powershell scripts to load executables, for example as shown in the image below, where I used Registry Hijacking to bypass UAC.



CPPTOOLS.EXE-057C0237.pf	8/10/2023 5:56:19 AM	8/10/2023 6:00:11 AM	12,041	CPPTOOLS.EX
POWERSHELL.EXE-CA1AE517.pf	8/9/2023 11:38:08 AM	8/10/2023 6:00:11 AM	41,210	POWERSHELL
CMD.EXE-0BD30981.pf	8/9/2023 11:34:39 AM	8/10/2023 6:00:11 AM	3,662	CMD.EXE
CODE.EXE-14EEF3B4.pf	8/10/2023 5:53:15 AM	8/10/2023 6:00:11 AM	50,888	CODE.EXE
CONHOST.EXE-0C6456FB.pf	8/9/2023 11:35:44 AM	8/10/2023 5:59:58 AM	6,186	CONHOST.EX
DLLHOST.EXE-7D5CE0CA.pf	8/9/2023 11:39:06 AM	8/10/2023 5:59:53 AM	5,552	DLLHOST.EXE
MICROSOFTEDGEUPDATE.EXE-7A595326.pf	8/9/2023 11:37:05 AM	8/10/2023 5:57:38 AM	10,026	MICROSOFTE

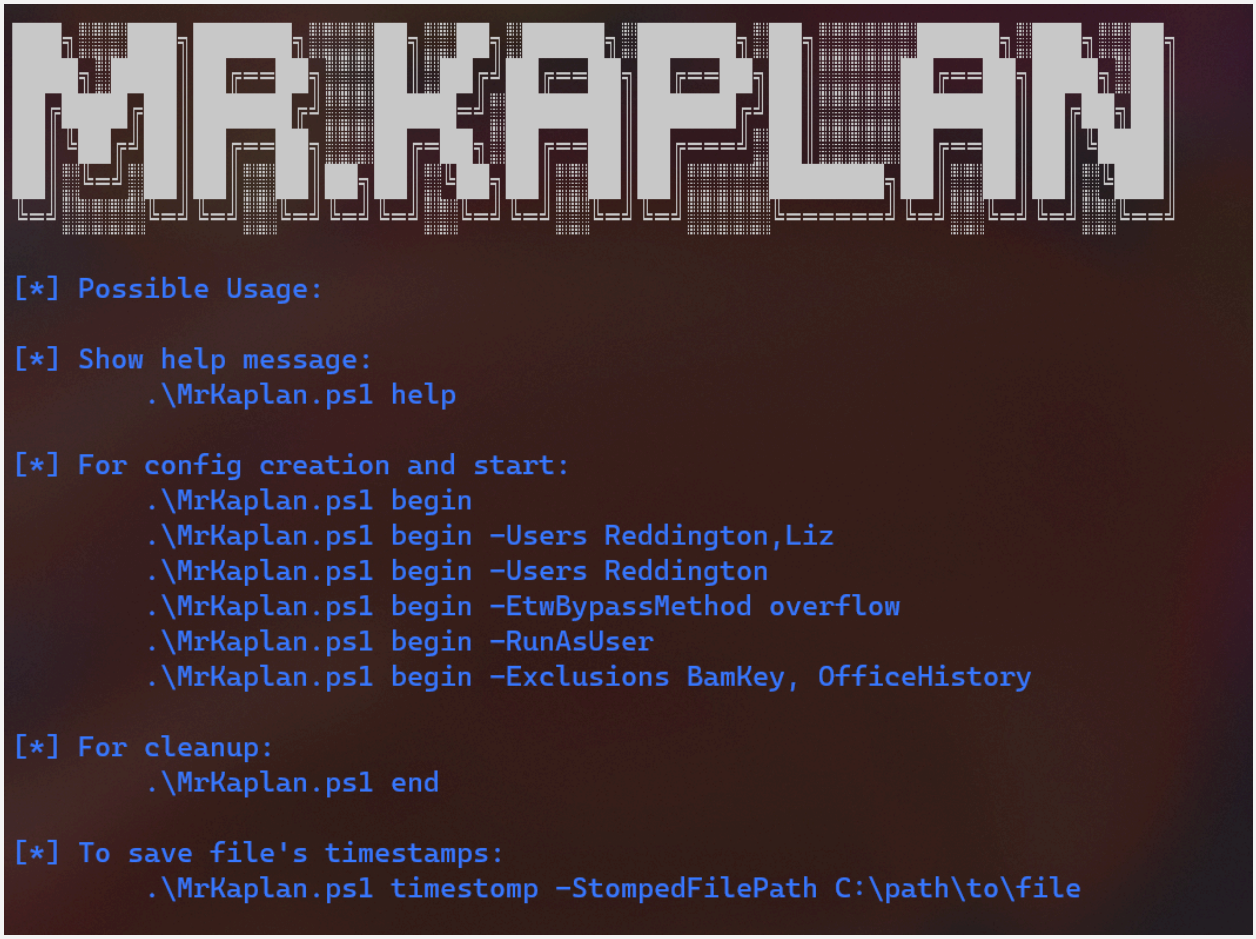
Filename	Full Path
__PSSCRIPTPOLICYTEST_TBERARPL.5RI.PS1	C:\USERS\AMADE\APPDATA\LOCAL\TEMP__PSSCRIPTPOLICYTEST_TBERARPL.5RI.PS1
__PSSCRIPTPOLICYTEST_URADXYDB.O4K.PSM1	C:\USERS\AMADE\APPDATA\LOCAL\TEMP__PSSCRIPTPOLICYTEST_URADXYDB.O4K.PSM1
ADVAPI32.DLL	C:\Windows\System32\advapi32.dll
AMSI.DLL	C:\Windows\System32\amsi.dll
APPIDAPI.DLL	C:\Windows\System32\appidapi.dll
ATL.DLL	C:\Windows\System32\atl.dll
BCRYPT.DLL	C:\Windows\System32\bcrypt.dll
BCRYPTPRIMITIVES.DLL	C:\Windows\System32\BCRYPTPRIMITIVES.DLL
BRO.EXE	C:\Users\Amade\AppData\Local\Temp\Bro.exe

In VSCode, `C:\Users\%username%\AppData\Roaming\Code\CachedExtensionVSIXs` is a path that shows what launchers they have, for example of C++, Python, Java and so on. On the other hand, `C:\Users\%username%\AppData\Roaming\Code\User\History\` quite literally logs every single script that the user initiated. Thus checking USN Journal for either extensions in VSIXs folder (let's say, .cpp) or the History directory (where specific folders get removed) we will be able to check if they launched and deleted scripts from VSCode.... Or just straight up find the code itself!



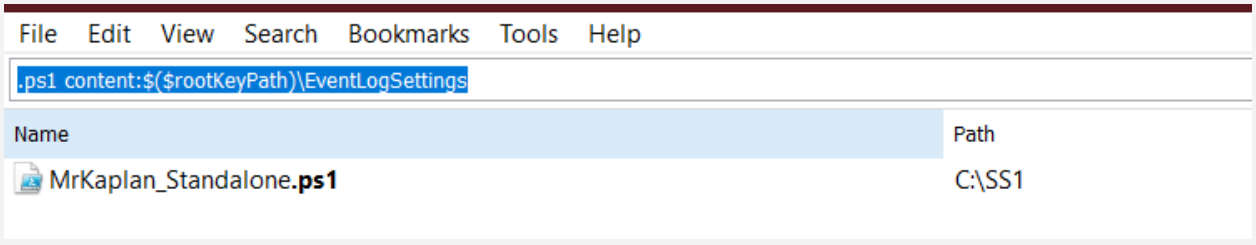
Opening this file will reload the code they executed. Using this, we can also detect shellcode injection as many bypassers just use VSCode for it. All of these extensions are also logged in WinPrefetchView, of course, just like for every other scripting environment. That's just life 🙂

Mr.Kaplan



Usually, MRKaplan is a Red Team Anti-Forensics utility reliant on Powershell that can be used to disable BAM inheritance, run files by custom runtimes and more. The file is open-source on Github and is quite unknown for bypassers. However, we can detect this using Search Everything by analyzing any string within the contents of the .psm1 or .ps1 scripts of the utility!

[Idov31/MrKaplan]



For example, scan all .ps1 files that check for eventlog settings on the device which is not a normal file behavior! `.ps1 content:$($rootKeyPath)\EventLogSettings`

YARA

Introduction to YARA - a File detection utility

General Understanding

There's a reason I brought up Mr.Kaplan. In its github page you will be met with a .yara file which displays an entire code that can be used to detect the utility on the system regardless of what they do with it (Rename, move, or anything) as long as it isn't deleted (otherwise use USN Journal to detect it.)

Let's review the code and understand what it does:

```
rule MrKaplanStandalone {
    meta:
        description = "A rule to detect MrKaplanStandalone."
        author = "Idov31"
        date = "2022-04-15"

    strings:
        $imports1 = /[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String(.*) | Invoke-Expression/i nocase
        $imports2 = /[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String(.*) | iex/i nocase

        $s1 = "MrKaplan.ps1" ascii nocase
        $s2 = "Clear-Evidence" ascii nocase
        $s3 = "EventLogSettings" ascii nocase
        $s4 = "runAsUser" ascii nocase
        $s5 = "PSHistory" ascii nocase
        $s6 = "C:\\Users\\$($user)\\AppData\\Roaming\\Microsoft\\Windows\\PowerShell\\PSReadLine\\ConsoleHost_history.txt" ascii nocase
        $s7 = "HKCU:\Software\MrKaplan" ascii nocase
        $s8 = "Invoke-StompFiles" ascii nocase
        $s9 = "Clear-Files" ascii nocase
        $s10 = "Clear-Registry" ascii nocase
```

```
        $s11 = "Invoke-RestoreEtw" ascii nocase
        $s12 = "Invoke-LogFileToStomp" ascii nocase
        $s13 = "Invoke-SuspendEtw" ascii nocase

    conditions:
        any of $imports* and 3 of ($s*)
}

rule MrKaplan {
    meta:
        description = "A rule to detect MrKaplan."
        author = "Idov31"
        date = "2022-04-15"

    strings:
        $imports1 = "Import-Module .\\Modules\\Registry.psm1" ascii nocase
        $imports2 = "Import-Module .\\Modules\\Files.psm1" ascii nocase
        $imports3 = "Import-Module .\\Modules\\Eventlogs.psm1" ascii nocase
        $imports4 = "Import-Module .\\Modules\\Utils.psm1" ascii nocase
        $imports5 = "ipmo .\\Modules\\Registry.psm1" ascii nocase
        $imports6 = "ipmo .\\Modules\\Files.psm1" ascii nocase
        $imports7 = "ipmo .\\Modules\\Eventlogs.psm1" ascii nocase
        $imports8 = "ipmo .\\Modules\\Utils.psm1" ascii nocase



        $s1 = "MrKaplan.ps1" ascii nocase
        $s2 = "Clear-Evidence" ascii nocase
        $s3 = "EventLogSettings" ascii nocase
        $s4 = "runAsUser" ascii nocase
        $s5 = "PSHistory" ascii nocase
                                                                    $s6          =
"C:\\Users\\$($user)\\AppData\\Roaming\\Microsoft\\Windows\\PowerShell\\PSReadLine\\
ConsoleHost_history.txt" ascii nocase
        $s7 = "HKCU:\Software\MrKaplan" ascii nocase
        $s8 = "Invoke-StompFiles" ascii nocase
```

```
        $s9 = "Clear-Files" ascii nocase
        $s10 = "Clear-Registry" ascii nocase
        $s11 = "Invoke-RestoreEtw" ascii nocase
        $s12 = "Invoke-LogFileToStomp" ascii nocase
        $s13 = "Invoke-SuspendEtw" ascii nocase

    conditions:
        4 of $imports* and 3 of ($s*)
}
```

Source : [MrKaplan/MrKaplan.yara]

As it is evident, the yara script hunts any file that contains 13 specific strings and 8 imports, then under the condition of 4 imports and at least 3 strings detected in one file - flags it as MrKaplan. The YARA documentation can be found here [Welcome to YARA's documentation! — yara 4.5.0 documentation]

How is this useful? - if you get your hands on cheat files and use HxD or Detect it Easy on them to extract and analyze imports, entropy values, strings and so on, you can create a YARA script that scans all files on the device for that file. You can also limit sizes, etc. For example, here's a rule that detects all files that require Admin privileges to be executed on the device:

```
rule requireAdmin
{
    meta:
        description = "An example script"
        threat_level = 3
        in_the_wild = true
    strings:
        $c = "level="requireAdministrator""
    condition:
        $c
}
```

Command Lines, Usage and [Download]:

```
YARA 4.5.0, the pattern matching swiss army knife.
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID
Mandatory arguments to long options are mandatory for short options too.
      --atom-quality-table=FILE        path to a file with the atom quality table
  -C, --compiled-rules                 load compiled rules
  -c, --count                          print only number of matches
  -E, --strict-escape                  warn on unknown escape sequences
  -d, --define=VAR=VALUE               define external variable
```

```
-q,  --disable-console-logs            disable printing console log messages
     --fail-on-warnings                fail on warnings
-f,  --fast-scan                       fast matching mode
-h,  --help                            show this help and exit
-i,  --identifier=IDENTIFIER           print only rules named IDENTIFIER
      --max-process-memory-chunk=NUMBER    set maximum chunk size while reading process
memory (default=1073741824)
-l,  --max-rules=NUMBER                abort scanning after matching a NUMBER of rules
           --max-strings-per-rule=NUMBER           set  maximum  number  of  strings  per  rule
(default=10000)
-x,  --module-data=MODULE=FILE         pass FILE's content as extra data to MODULE
-n,  --negate                          print only not satisfied rules (negate)
-N,  --no-follow-symlinks              do not follow symlinks when scanning
-w,  --no-warnings                     disable warnings
-m,  --print-meta                      print metadata
-D,  --print-module-data               print module data
-M,  --module-names                    show module names
-e,  --print-namespace                 print rules' namespace
-S,  --print-stats                     print rules' statistics
-s,  --print-strings                   print matching strings
-L,  --print-string-length             print length of matched strings
-X,  --print-xor-key                   print xor key and plaintext of matched strings
-g,  --print-tags                      print tags
-r,  --recursive                       recursively search directories
     --scan-list                       scan files listed in FILE, one per line
 -z,  --skip-larger=NUMBER             skip files larger than the given size when scanning a
directory
-k,  --stack-size=SLOTS                set maximum stack size (default=16384)
-t,  --tag=TAG                         print only rules tagged as TAG
-p,  --threads=NUMBER                  use the specified NUMBER of threads to scan a directory
-a,  --timeout=SECONDS                 abort scanning after the given number of SECONDS
-v,  --version                         show version information
```

[Usage of the commands and examples]

Golden, then Ocean were the first ever SS Tools to implement YARA into their detection system. Red Lotus has created alternatives to YARA for Lotus+ with dozens of detections.

Customizing Collectors

Introduction to DFIR Collection

Implementing YARA into RL Collector



RLCollector is a Red Lotus Digital Forensic collector of many artifacts that can be imported into Timeline Explorer and read separately. Its source code & the python code can be downloaded from here, which allows you to modify downloadable resources that are zipped to launch them with specific commands. For example, in the code you see:

```
import os
import requests
from zipfile import ZipFile
from tqdm import tqdm

file_urls = [

"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/bstrings.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/EvtxECmd.zip",
    "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/PECmd.zip",
    "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RECmd.zip",
"https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SBECmd.zip",
  "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SQLECmd.zip",
 "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SrumECmd.zip",
  "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/SumECmd.zip",
    "https://f001.backblazeb2.com/file/EricZimmermanTools/net6/WxTCmd.zip"
]
```

To which you can add more downloadable files, for example, YARA and its .yara rule files. Then, you see:

```
import subprocess

# List of commands to be executed
commands = [
    'bstrings.exe -d C:\\Windows\\System32\\Tasks --ls Command --sa -s -o
.\\TaskArguments.txt',
```

```
                                                    'bstrings.exe            -d
C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows  --ls  .exe
--sa -s -o .\\processedEXES.csv',
                                                    'bstrings.exe            -d
C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows  --ls  .dll
--sa -s -o .\\processedDLLS.csv',
                                                    'bstrings.exe            -d
C:\\ProgramData\\Microsoft\\Search\\Data\\Applications\\Windows  --ls  :\\
--sa -s -o .\\processedPATHS.csv',
    'PECmd.exe -d C:\\Windows\\Prefetch --csv .',
    'SBECmd.exe -d C:\\Windows\\system32\\config --csv .',
    'SrumECmd.exe -d C:\\Windows\\System32\\sru --csv .',
    'SumECmd.exe -d C:\\Windows\\system32\\LogFiles\\Sum --csv .',
                'RECmd.exe    -d    C:\\Windows\\system32\\config    --bn
.\\BatchExamples\Kroll_Batch.reb --nl false --csv .',
    'EvtxECmd.exe -d C:\\Windows\\System32\\winevt\\Logs --csv . --dedupe'
]
```

To which you can add an option of the yara command within the current directory, as 'yara64.exe \rules\rule.yara [options] [output]` (refer to yara documentation to get used to commands), this way not only will you be able to download and collect specific DFIR Artifacts but also immediately start a scan of all suspicious files on the device!

The script can be compiled into an .exe using autopytoexe.

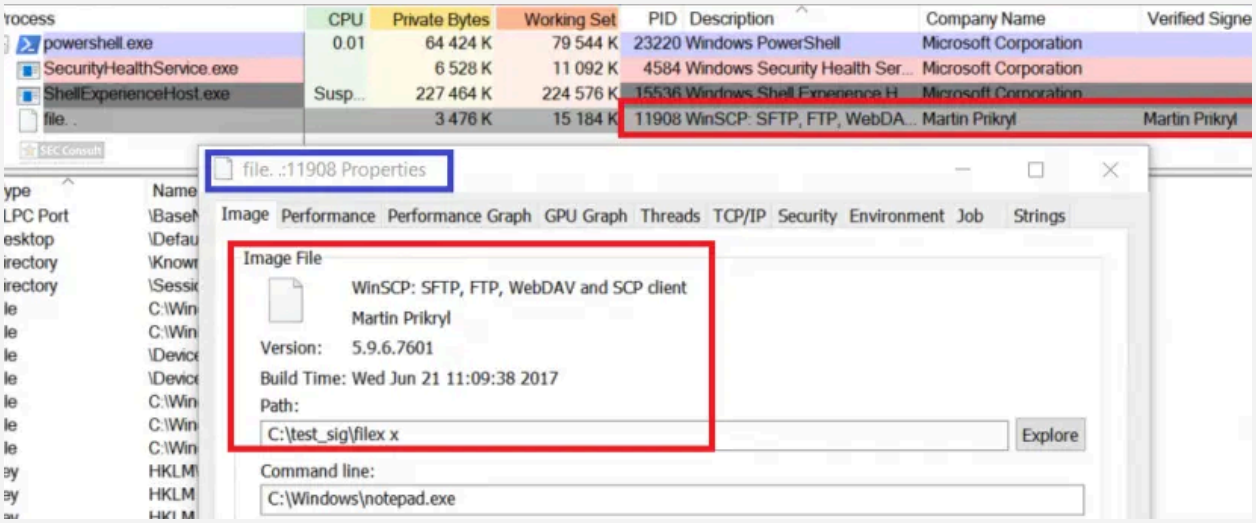
For further codes you might want to create, for file signature checks in Python you can use the following script:

```
def verify_authenticode_signature(file_path):
    try:
        sign_info = win32api.CryptQueryObject(
            win32api.CERT_QUERY_OBJECT_FILE,
            file_path,
            win32api.CERT_QUERY_CONTENT_FLAG_PKCS7_SIGNED_EMBED,
            win32api.CERT_QUERY_FORMAT_FLAG_BINARY,
            0,
            None,
            None,
            None,
            None,
            None,
            None
        )
        if sign_info['signers'][0]['verified'] == 1:
            return True
    except:
        pass
    return False
```

ADS Detections

Introduction to ADS and Junctions

Detecting Alternate Data Streams



As shown in the blog mentioned above (Anti-Forensic techniques), a weird file in C:\Test_sig\ directory is executed, it has no extension nor an icon, because it is an Alternate Data Stream, a file that is hidden as a data stream. While creating an ADS or applying it to a file/folder, executing then deleting the Stream itself, USN Journal will obviously create a “Stream Change” entry. Try it for yourself:

type Cheatfile.exe >> C:\Windows\pathtowherever\pathorfile:Zone.Identifier

The type command will create an ADS file. For example, let’s do `type AnyDesk.exe > C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier`

Then execute this ADS using forfiles (as it cannot be executed regularly) by running `forfiles /p C:\windows\system32 /m notepad.exe /c C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier`

Then let’s use AlternateStreamView to find and delete this Stream (specify discordptb folder for example) - we’ll be met with a Stream Change | Close on this file itself, which is more than enough to ban for if caught doing before SS. As you see in the screenshot above as well, file.: has a : mark after it. What does it mean? - it’s an ADS identifier, the file hidden as an ADS will always end with : and Zone.Identifier or $DATA will always be assigned to it. For example, to complicate things we’ll hide Anydesk as: `type AnyDesk.exe > C:\Users\%username%\AppData\Roaming\discordptb:Zone.Identifier:$DATA`

Thus with forfiles we’d need to execute it alongside with $DATA and usual detection methods might be more challenging to detect these with. If the file is kept on the device,

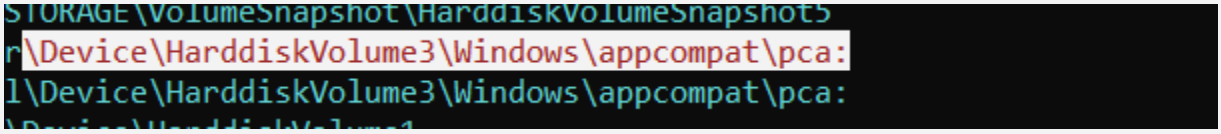
Alternate Stream View will be able to find it, exporting and modifying the extension of ads to .exe is more than enough as it also shows date of modification when launching the cheat. You can select an entire drive to scan too.

If the ADS is executed using a scheduled CMD, it will create a BAM entry but it won't log in Prefetch. Users also usually delete the ADS logged in BAM or even worse - rename the key which is practically impossible to detect. Though assuming they don't rename it, here are some steps we can take to detect ADS on the device.

Step I - Recovering BAM Entries

It is recommended to follow this step immediately when starting the SS, even before checking the recycle bin or running SS Tools, to avoid any data being lost. Use bstrings.exe and run the following command to print all the results that have logged into BAM: `bstrings.exe -f "C:\Windows\system32\config\SYSTEM" --ls harddiskvolume`

We follow the logic mentioned above and check file paths that end on :



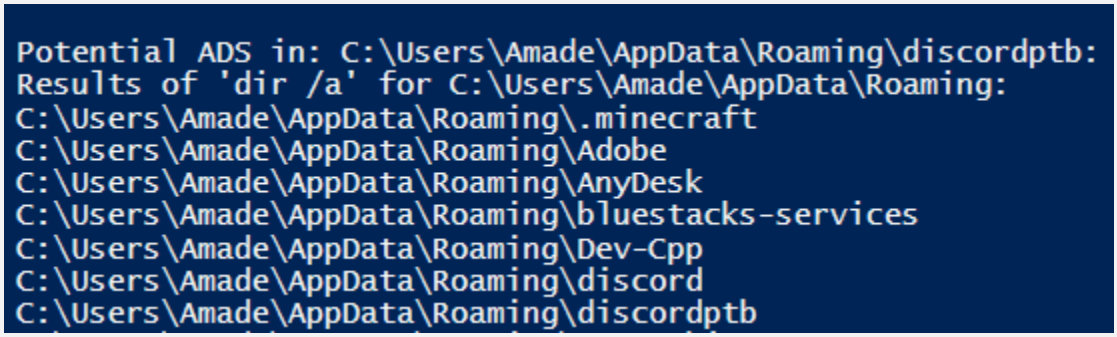
Super easy!

Step II - Using System Informer

Using regex `^([a-zA-Z]:\\.+)\\?$` in services like Explorer, Pcasvc, Searchindexer, Csrss, Diagtrack (if enabled) - we will export all file paths that have been logged, including ADS. Right click under Paths and save it into notepad, then use the following Powershell Script and provide the saved .txt results:

```
$file = Read-Host "Provide txts of SearchIndexer or Csrss or Explorer"
if (-not (Test-Path $file -PathType Leaf)) {
    Write-Host "Unidentified"
    exit
}
Get-Content $file | ForEach-Object {
    $line = $_.Trim()
    if ($line -match "^([a-zA-Z]:\\.+)\\?$") {
        $path = $Matches[1]
        if ($path.EndsWith(":")) {
            Write-Host "Potential ADS in: $path"
            $parentDirectory = Split-Path $path -Parent
```

```
        Write-Host ("Results of 'dir /a' for $parentDirectory`:")
        Push-Location $parentDirectory
        Get-ChildItem -Force | ForEach-Object {
          $_.FullName
        }
        Pop-Location
        Write-Host
    }
  }
}
```



Use the AlternateStreamView to find if the file is still on the device, otherwise use JournalTrace to check if “Stream Change | Close” was logged on the folder before ScreenShare.

Step III - Using $MFT

ADS is a very old method. SS Community has developed MFT parsing methods in the past to find all zone.identifier or :$DATA files on the device.

Run the command using MFTECmd.exe as follows:

`MFTECmd.exe -f C:\$MFT --csv .` To export MFT as a .CSV readable file. Put it into Timeline Explorer and search for the Zone.Identifier. That will detect all files, both legit and irregular ones (for example a folder having :zone.identifier whilst also being 20,000kb large - regular Zone IDs are at max 1050~kb or below in size) although there isn’t much difference between using this and the AlternateStreamView itself. Previous methods are advisable.

CHAPTER IX. Windows Registry

1. Introduction

Most often do beginners and intermediate ScreenSharers struggle with Registry Explorer after they're done exhausting all methods for one or two checks they are aware of or have learnt from "Advanced" ScreenSharers. Veterans of the community are undoubtedly skillful and experienced, but those blinded by catching too many bypassers (most of which tend to be weak) have stopped learning as early as 2021~2022 and currently it is hard to find anybody who can recommend this utility to be used at its fullest extent.

Before I explain what Registry Explorer does or what its alternatives are, let's discuss a thing or two about Regedit itself.

Windows Registry is an apparent database of system or application settings. Upon installing Windows, for example, JumpLists and Diagtrack will have their settings preset to be disabled per installation if users refuse tracking functions of Windows, or, for instance, installing scripting environments (VSCode, Golang) as well as even text editing applications or DLL injectables (Notepad++ or HxD; RemoteDLL64) will also generate their shell application settings, boot on startup settings and so on, these will create Registry Entries.

Many online sources say that via Registry users are able to make high or low-level changes to the system, take it for granted, even malware utilizes Registry to make elevation-level (so, high) changes to the system, prime example being the UAC bypass, disabling inheritance and much more.

Contrary to that, applications, services and utilities like system optimizers (Hone for example) or any open-source github projects usually and most definitely ask users to make registry backups before initiating changes to the system settings because of irreversible damages they may hold on certain devices. *For example in the early 2018~2019s when creating custom OS themes was a great trend, many msi installers of windows themes would brick user devices because of how negatively their system settings reacted to registry modifications.*

Registry Sources and Characteristics

Registry data is stored in "configuration" files located in c:\windows\system32\config. The directory stores databases for "registry hives", *[they're called hives because the original windows NT developer... hated bees.]* Essentially, the registry is separated into multiple divisions that store branches of keys and their subkeys maintained under hive (or in other words - configuration) files. These Hives are SOFTWARE, SYSTEM, SAM. . . Each individual user is defined under their individual Hives that store settings characteristic to that specific user. For example, if there are two separate users on the same device, each user will share the given User hives and

display their configurations. Following that point, if user A was to install RemoteDLL64, user B would be able to view effective Registry configurations relevant to A's settings by viewing their user Hive. **Details like these come handy when looking for Ban Evaders and Alts as this logic doesn't require invading privacy further by checking emails, even if nobody really does or should be doing that.**

Registry Values

Registry Value is data stored under subkeys. Values exist in different types, string values, QWord & DWord values, expandable, binary and multi-string values. These are "files" displayed to the right when enrolling subkeys of registry Hives. Lifewire perfectly explains what each value stands for and what their functionalities are. In practice, rarely but realistically ScreenSharers come across Value manipulations of BAM, PCA, Activities Cache, JumpLists, Recent Docs and so on. In my own practice an expandable Variable was used, more like created by the user to launch cheats via Win+R; how did it work? **creating an expandable Registry Value means that any application or user action initiating the pre-defined setting (in this case - launching an executable) will be "expanded" into whatever the user had specified it to do.** So, if the Value is **REG_EXPAND_SZ** set "SHELL:RECENT" (just an example, modifying Windows shell values shouldn't be possible but who knows...), that is meant to launch %USERPROFILE%\DOCUMENTS\CHEAT.PDF, it'll launch that CHEAT.PDF file every time shell:recent is called, in other words, when calling SHELL:RECENT then the defined variable will be expanded, launched.

Registry Value Types

[A registry value can store data in one of several formats, such as a string value or an integer value. When you store data in a registry value—for instance by calling the RegSetValueEx function—you can indicate the type of data being stored by specifying one of the types in the table below. When you retrieve a registry value, functions such as RegQueryValueEx use these types to indicate the type of data retrieved.]

(source) **|** *Registry value types - Win32 apps | Microsoft Learn*

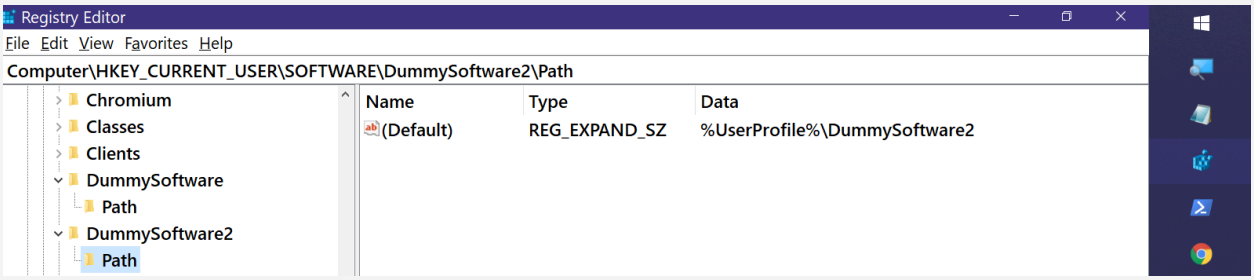
`REG_BINARY`	Binary data in any form
`REG_DWORD`	A 32-bit number.
`REG_DWORD_LITTLE_ENDIAN`	A 32-bit number in little-endian format. Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_DWORD in the Windows header files.

`REG_DWORD_BIG_ENDIAN`	A 32-bit number in big-endian format. Some UNIX systems support big-endian.
`REG_EXPAND_SZ`	A null-terminated string that contains unexpanded references to environment variables, for example, %PATH%. It's either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions. To expand the environment variable references, use the ExpandEnvironmentStrings function.
`REG_LINK`	A null-terminated Unicode string that contains the target path of a symbolic link that was created by calling the RegCreateKeyEx function with REG_OPTION_CREATE_LINK.
`REG_MULTI_SZ`	A sequence of null-terminated strings, terminated by an empty string (\0). The following is an example: String1\0String2\0String3\0LastString\0\0. The first \0 terminates the first string, the second-from-last \0 terminates the last string, and the final \0 terminates the sequence. Note that the final terminator must be factored into the length of the string.
`REG_NONE`	No defined value type.
`REG_QWORD`	A 64-bit number.
`REG_QWORD_LITTLE_ENDIAN`	A 64-bit number in little-endian format. Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_QWORD in the Windows header files.
`REG_SZ`	A null-terminated string. It's either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.

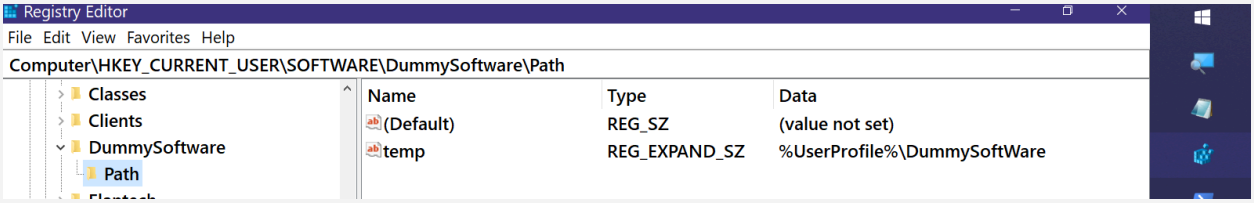
Registry Value manipulations

Linking Registry Value types is no accident. Let's discuss the bypass I mentioned above, the expandable Registry Value that creates a launchable application. To demonstrate it before explaining, using Powershell to create an expandable value that launches applications is quite simple. I'll cite an experiment done by Keith Miller. Here's the script Keith came up with:

```
$Splat = @{ 'Path' = 'HKCR:\SOFTWARE\DummySoftware2\Path'
'Value = '%UserProfile%\DummySoftware2'
 'ItemType' = 'ExpandString'
 'Force' = $False }
 New-Item $Splat
```



With this approach, it is interestingly possible to create launchable Executables through the temp env. for example;



Let's also review what author of the post asked prior to Keith's answer; in this blog from 2021 we encounter an expandable value manipulation using the reg-add command, syntax of the command looks like this:

reg add <keyname> [/v valuename | /ve] [/t datatype] [/s separator] [/d data] [/f] [/reg:32 | /reg:64]

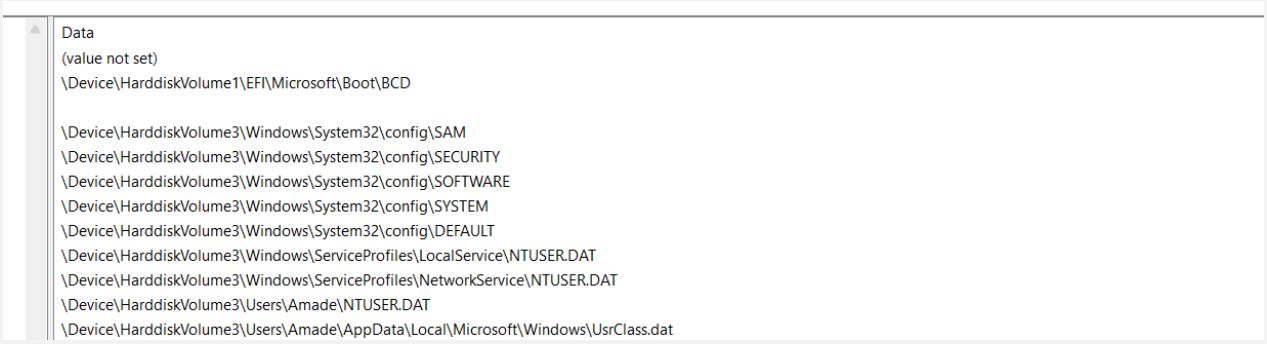
Following the Syntax, we'll first need to choose a Key (HK) like the author did as an example - HKCR\Drive\shell\Cmder\command; then, we'll set the Value name after /v (which stands for value), for example (Default) which is also shown in the previous image or whatever, let's call it (Whatever), /t stands for the data type so we'll specify it to be REG_EXPAND_SZ as shown in the previous image too with /d - data, to specify which data to expand to when a function is called. More parameters can be viewed at "reg add | Microsoft Learn";

All and all the final product looked like this: **reg add HKLM\Software\Ohmygoodness /v Path /t REG_EXPAND_SZ /d ^%systemroot^%\cheatfile.exe**

If all of that still sounds confusing, unfortunately there are practically no video demonstrations of this function online, so refer to sources that this page was based on windows - How to use REG_EXPAND_SZ from the commandline? - Stack Overflow | windows - Creating registry key with expandable string value named (Default)? - Super User

Registry Hivelist

Before we proceed further, it is important to know of this verifiable ability of checking what the given Regedit sources its data from, so just like with Last Activity Viewer, for example we can view which Registry keys or “dependencies” it gathers its data from to display - registry also has a Subkey that shows all of its sources and that is HKLM (Hkey Local Machine) subkey under Control **Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\hivelist**



After being told of this, it all comes down to a more understandable way of explaining what registry really is. As we can see, **registry appears to be a database of configurations applied by system or user-added files on the device that both the system and numerous other applications, as well as user-added applications use as their settings of behavior. Same goes for services as they tend to query registry entries to function, which is why it is possible to disable some of them by modifying Registry Values.**

Registry Forensics

Forensically analyzing any Windows artifact means **collecting evidence items** to further prove suspicious behavior, like paying attention to detail on persistent event IDs, looking at system behavior after CMD in USN Journal or $MFT and so on. It is often enough to identify these unusual chains of actions that regularly should not be occurring on the system, the challenge is to identify them correctly. To develop that “correct” understanding and logic, a ScreenSharer must first learn general ideas behind artifacts and functions of the system. Let’s review the basis, typicalities of Windows Registry.

Regedit.exe in Prefetch is not normal Windows behavior, viewing it in BAM or Prefetch usually means that the Player had accessed it some time during the given date. To speculate what could have happened, first it is worth noting that Regedit “remembers” last accessed keys and subkeys, checking user actions in regedit itself is practically impossible unless the ScreenSharer opens regedit that players forgot to close subkeys of. Therefore, paying attention to files before and after regedit.exe, as well as uptime of regedit (SRUM time or activities cache focus time) can indicate suspicious behavior. The date of regedit execution can be used for $UsnJrnl analysis or to deduct actions that followed. For example, seeing

Regedit.exe being accessed before the game (FiveM, Minecraft), uptime of which lasts until before AnyDesk, we can speculate that the user launched the game with regedit to hide evidence of its usage. Thus it is only logical to begin analyzing regedit with utilities like RECmd, Bstrings or Registry Explorer.

Sometimes it is impossible to check the uptime of applications or it takes too much time to do so. In such cases what the ScreenSharer shall orient on is the general information easily gathered via temp, recent and prefetch directories, matching times, speculating what could have happened and what to look for.

Reg.exe is a CLI based registry application preinstalled on Windows. As shown in previous examples, it can be used to modify registry values. Usually it's only the user who can launch this application, it'll log in prefetch that can be used as an indication of something being modified in Registry itself. Something removed, for example. Recent day bypasses use reg.exe to clear data from BAM because their bypasses rely on executable cheats, all of which logs in BAM. Diagtrack service can often display what command was initiated. However, when diagtrack is turned off and there is no real way of checking commands history, logically building an evidence-based conclusion is absolutely enough to issue penalties for. After all, no good ScreenSharer should let go of someone who has accessed reg.exe after CMD with CURL logs in Prefetch (or deleted which logged in Journal), before AnyDesk or near the game.

It is important to determine whether or not these applications were actually launched by the user. Reg.exe and Regedit.exe are 99% of the time **always run by the Player** but, how do we tell what is right? here is a checklist to keep in mind when analyzing uncertainties:

Pay attention to execution dates - If for example reg.exe is logged under exact hours, minutes of some other applications (for example driver booster) then it is logical to assume this was not done by the user. If for example regedit.exe was launched after accessing any desk or running scanners like Ocean or Echo then it is a fact that the user did not run them.

Link one behavior to the other - Nothing on Windows devices is a separate action. Even deleting a file has an alternative to go into the bin or not via shift deleting. Both of these actions create pop-up windows. It is logical to assume every single action leaves traces elsewhere. This is the case for injections and executions as well. Even when an application crashes it creates hang and crash logs be it in crashdumps or event logs.

Pay attention to detail - Every tiny bit of information is a detail. Distinguish between necessary and unnecessary details. For instance, when looking for renamed extensions, every extension is a necessary detail except for .dlls and .exes or .jars or whatever initial extension a ScreenSharer is looking for. It is also a necessary detail to look at execution dates and sort by date modified practically everywhere except for $MFT and its different access columns. So is knowing that Ocean generates a string scanner to avoid Ocean killers under different characters and so are paths necessary details.

Memorize or understand path permissions - Not every path can be modified by the user. Sometimes file paths can be corrupted or require special privileges to be accessed. This will be the case when trying to open Registry Explorer through Search Everything, as the file requires Admin perms and Search Everything's meant to only display applications. Paying attention to paths is exceptionally necessary when looking for things like ADS or RunDll/Regsvr (or any other prefetch) injections. For example, RUNDLL will always load Globalization or WOW64/Sytem32 applications first thus a ScreenSharer shall look for files run from paths

besides the two. It is also important to pay attention to Index in Prefetch as most cheats load their cheat file as 4th in the list of Index thus it becomes a bit easier to detect renamed extensions.

Remember registry - Whether it's a file execution, dll injection, value modification or addition, folder access, shell folders access, user logon or application hang, scheduled task creation, modification or deletion, unicode character files or folders, jumplists or alternate data streams - registry logs it all. Although sometimes temporarily, it still logs everything. When there is no other option left, registry approaches often come in handy the most.

Analyzing Registry Changes

RegShot is one of the most useful tools that allows "dynamic analysis" by comparing two Registry snapshots with each-other. It's basically old string comparator softwares used to dump strings from services for ScreenSharing, thus it is used on the ScreenSharer's device if they wish to test bypasses and find manual traces in the registry. For example, if we want to see what changes take place in Registry when installing .msi applications or hex editor applications - this way we can also extract relevant registry keys and their subkeys to then check on users and verify that a suspicious behavior occurred. [Seabreg/Regshot: Regshot is a small, free and open-source registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one - done after doing system changes or installing a new software product (github.com)]

Usage:

---

(1) Click the "1st shot" button
It pops up a menu which contains several items:
(A) "Shot", to take a snapshot only, and it will not be kept if you exit Regshot program;
(B) "Shot and save...", to take a snapshot of your registry and save the whole registry to a "hive" file and you can keep it in your hard disk for future use;
(C) "Load...", to load a "hive" file previously saved.
If you want to monitor your file system, just check the "Scan Dir [dir..]" checkbox and input the folder names below it. Note: Regshot has the ability to scan multiple folders, just separate them with ";", Regshot will also scan the subfolders of the current folders you entered.
(2) Run any program that may change your windows registry or the file system
(3) Click the "2nd shot" button
(4) Select your output LOG file type, "text" or "HTML", default is "text"
(5) Enter your comment for this action into the "comment field", e.g.: "Changes made after winzip started". The comment will only be saved into the comparison log files not into "hive" files
(6) Click the "Compare" button

Regshot will do the comparison job now (auto detect which shot is newer), when it is finished,

Regshot will automatically load the comparison log as you defined above, the log files are saved in the directory where "Output path" is defined, default is your Windows Temp Path, it was named as the "comment" you input, if the "comment field" is empty or invalid, the log will be named as "~resxxxx.txt" or "~resxxxx.htm" where "xxxx" is 0000-9999.

(7) Click the "Clear" button

The two snapshots (or separately) previously made will be cleared and you can begin a new job.

Note: "Clear" does not erase the log files!

(8) To quit Regshot, just click the "Quit" button

(9) You can change the language of the Regshot at main window, all words are saved in the file "language.ini". View it for details!

(10) New to 1.7: regshot.dat now changed to regshot.ini, skip dirs and skip registry keys are included. The executable's size is smaller!

(11) New to 1.8:

File shots are now saved in hive file.

"UseLongRegHead" option added in regshot.ini to compatible with undoReg (1.46)

"UseLongRegHead=1" means using "HKEY_LOCAL_MACHINE" instead of "HKLM", default 0

Do not compare shots saved with different "UseLongRegHead" options! (limit gone in 1.9.0)

(12) New to 1.9.0:

x64 and Unicode versions.

You can compare shots saved with different "UseLongRegHead" options.

Loading ANSI hive files in Unicode builds and vice versa is not supported.

What should you be looking for in the final results? As seen in its demo of dynamic malware analysis, it's advised to pay attention to the extraction sizes, check the second snapshot (so the "infected" one) for additional registry values or subkeys generated and keep them for future checks of scripts you may create. For example, here's a script that checks for an input registry key and its subkeys that the user provides (powershell):

```
$keyPath = Read-Host "Enter the registry key path (e.g., HKCU:\Software\Microsoft)"

if (Test-Path $keyPath) {
    Write-Host "Registry key '$keyPath' exists."

    $subKeys = Get-ChildItem -Path $keyPath -Recurse | Where-Object { $_.PsIsContainer }

    if ($subKeys.Count -gt 0) {
        Write-Host "Subkeys under '$keyPath':"
        foreach ($subKey in $subKeys) {
            Write-Host $subKey.Name
        }
    } else {
```

```
    Write-Host "No subkeys found under '$keyPath'."
  }
} else {
  Write-Host "Registry key '$keyPath' does not exist."
}
```

Thus, with this you analyze Registry behavior of cheats and implement them for personal detections. Possibilities are almost limitless.

Utilizing VirusTotal is probably the best registry analysis approach alongside with many others, it links us to other virus analysis results as well.

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	171920	172032	6.5	d77650c42fe7f26e50ffd3a1eb04824b	1013751.12
.rdata	176128	76322	76800	5.83	57d42e7dbe8bbf2a4f453a4a584667c3	2101595.5
.data	253952	13128	3584	1.83	ab6bbf08f3667724642db2d3bfd413c7	589548.12
.pdata	270336	8976	9216	5.36	189d7d6cd859f2885b41151291df8e67	291309.31
_RDATA	282624	348	512	2.82	8073ac06255148b944b04f6a459b3f91	57765
.rsrc	286720	61944	61952	7.33	ea06cf7460cdc7bec119e2d78f419f3d	308723.84
.reloc	352256	1884	2048	5.24	ec92fd910e5f98e8bd04ca09d1b19e26	17510.75

For example here, the higher file entropy the more packed (obfuscated) it is, if one of the section names say ".themida" (in any form like themida or @.themida or @themida) and so on, the fine in practice often appears to be a cheat because cheat devs use Themida to obfuscate their clients.

And in behavior, Registry actions can be also analyzed.

Registry actions

Registry Keys Set

+ HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\GoogleUpdaterInternalService126.0.6441.0\ImagePath
+ HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\GoogleUpdaterInternalService126.0.6441.0\Start
+ HKEY_USERS\S-1-5-21-4270068108-2931534202-3907561125-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.exe\OpenWithProgids\exefile

Collecting Registry data

Red Lotus has already made a Collector [https://cdn.discordapp.com/attachments/1178373626156826695/1212553955515699210/RLCollector.exe?ex=663b6d1f&is=663a1b9f&hm=efd868b3af9811fb3f0de238013ab881a435f6a2d79bf9c263c9b91a6558de29&] that can be used to Collect numerous Artifact data including of the Registry itself.

For Registry collection it relies on Eric Zimmerman's RECmd [https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RECmd.zip]

The command run to collect pre-defined registry entries (that come with the utility under .reb files) was: RECmd.exe -d C:\\Windows\\system32\\config --bn .\\BatchExamples\Kroll_Batch.reb --nl false --csv .

Essentially we're providing RECmd with a directory to scan which is the config folder with all the registry files in it and parse batch examples, practically extracting all the necessary information we'd need from Registry itself. As the channel linked above already displays its collections, we won't delve deep into this utility. However, let's explain what RECmd is and how it functions.

RECmd.exe is a CLI (command line) based utility that can access and read data from Registry. Think of it as reg.exe but with more possibilities and extraction formats as results can be collected as .csv files that can then be read using Timeline Explorer. The results could also be used by bstrings to immediately hunt for necessary data in the whole registry. This approach becomes handy after analyzing cheat / malware behavior as we already know what to look for exactly.

Let's review its command options

```
    d           Directory to look for hives (recursively). -f or -d is required.
    f           Hive to search. -f or -d is required.
    q           Quiet mode. When true, hide processing details. Default is FALSE
    kn          Display details for key name. Includes subkeys and values
    vn          Value name. Only this value will be dumped
    bn          Use settings from supplied file to find keys/values. See included sample file for examples
    csv         Directory to save CSV formatted results to. Required when -bn is used.
    csvf        File name to save CSV formatted results to. When present, overrides default name
    saveTo      Saves --vn value data in binary form to file. Expects path to a FILE
    json        Export --kn to directory specified by --json. Ignored when --vn is specified
    jsonf       File name to save JSON formatted results to. When present, overrides default name
    details     Show more details when displaying results. Default is FALSE
    Base64      Find Base64 encoded values with size >= Base64 (specified in bytes)
    MinSize     Find values with data size >= MinSize (specified in bytes)
    sa          Search for <string> in keys, values, data, and slack.
    sk          Search for <string> in key names.
    sv          Search for <string> in value names
    sd          Search for <string> in value record's value data
    ss          Search for <string> in value record's value slack
    literal     If true, --sd and --ss search value will not be interpreted as ASCII or Unicode byte strings
    nd          If true, do not show data when using --sd or --ss. Default is FALSE
    regex       If present, treat <string> in --sk, --sv, --sd, and --ss as a regular expression. Default is FALSE
    dt          The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd
HH:mm:ss.fffffff
    nl          When true, ignore transaction log files for dirty hives. Default is FALSE
    recover     If true, recover deleted keys/values. Default is TRUE
    vss         Process all Volume Shadow Copies that exist on drive specified by -f or -d . Default is FALSE
    dedupe      Deduplicate -f or -d & VSCs based on SHA-1. First file found wins. Default is TRUE
    sync        If true, the latest batch files from
https://github.com/EricZimmerman/RECmd/tree/master/BatchExamples are downloaded and local files updated.
Default is FALSE
    debug       Show debug information during processing
    trace       Show trace information during processing
  Example: RECmd.exe --f "C:\Temp\UsrClass 1.dat" --sk URL --recover false --nl
      RECmd.exe --f "D:\temp\UsrClass 1.dat" --StartDate "11/13/2014 15:35:01"
```

```
RECmd.exe --f "D:\temp\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
```

--d

The --d switch refers to the “directory” of files to collect data from. So if we were to use FTK Imager to extract all config files into a separate directory (this prevents new data from being written into the registry and old data from potentially being lost or overwritten) we would specify that directory to be analyzed. However, often extractions of this kind corrupt the hives. Thus RECmd comes with RLA that can potentially correct these errors. The github page shows exactly how it works.

It is worth noting that the switch requires a target directory and not of RECmd, thus specify directories that have hives in them. Also, this utility can be used to collect data from alternative Windows user accounts that are still stored within their dedicated folders. NTUser and account hives can usually be found to be system hidden in c:\users\%username% as well. In this case we may specify separate usernames that we’re interested in.

--f

This command line orients on specific hives that one would want to scan. For example, if we determine that a certain malware’s behavior is traced in the SOFTWARE hive, there will be no need to potentially affect and also potentially overwrite all the hives using the --d switch. Or, if the --d switch prints errors it may often be because some files can’t be accessed in certain directories. --f can often get along with that.

--bn

As shown in the command above, we specified batch examples after the --bn switch that recursively took every single example command and ran it instantly. This command can be used to initiate collections that users come up with themselves. Hierarchically, RECmd shall begin with the --d switch, then --q switch to make the scan quiet (it speeds up the scan speed) and then any order given in examples.

--sa

Ignoring every other switch, --sa will look for specific strings in the entirety of Registry, this is usually an ASCII scan (excludes Unicodes) but using REGEX we can also print out Unicode characters - or simply do not specify --sa and let the --bn do its work. --regex must be used alongside with --sa, thus with --regex you must have --sa including a regex string.

Combining

In the end, RECmd command must include the following things:

1) --f or --d to scan
2) --bn or --sa with --regex (or without)
3) --q to speed up the scan (or without)
4) --sa is mandatory (it’s string to look for)
5) --csv or --csvf or --json (in the end to output a results file)
6) --dedupe before --csv to deduplicate repeating results and lines
7) --vss to also parse volume shadow copies (recovers data from previous windows snapshots) - snapshots refer to backups windows automatically makes. Not previous windows installations.

Dumping Registry data

If a ScreenSharer uses FTK Imager to extract a registry hive (or any other way of copying config registry hives), specifically the SYSTEM hive, bstrings can read potentially residue data still kept on it before everything gets overwritten. What does it mean? - if users clear their BAM (background activity moderator) logs and the ScreenSharer makes a copy of their SYSTEM hive (on their own system, obviously) they can then use the following command:

```
bstrings.exe -f "C:\Windows\system32\config\SYSTEM" --ls harddiskvolume
```

To read that deleted data before it gets overwritten! (as mentioned above, making a copy or extracting these hives prevents them from being updated / overwritten)

Alternatively, using the --regex switch can print only the file paths! **^([a-zA-Z]:\\.+)\\?$**

Registry Explorer

Registry Explorer comes with a Manual that explains everything necessary. In fact, the DFIR communities have created python scripts that allow the usage of Reg Explorer into utilities like Autopsy. source: Autopsy-Registry-Explorer/regparser.py at main · 0xHasanM/Autopsy-Registry-Explorer (github.com)

However, for ScreenSharing purposes utilities like Autopsy, Volatility and MemProcFS will not be covered in Red Lotus handbooks (mainly because they already have their own documentations and there's nothing new Lotus can offer regarding them); they may be released if Lotus+ members request to with specific approaches on how to detect certain anomalies.

[Eric Zimmerman's tools] **Registry Explorer -** is a GUI based RECmd that revolutionized working with Windows Registry. Before proceeding, let's follow the license requirement.

MIT License

Copyright (c) 2021 Eric Zimmerman

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

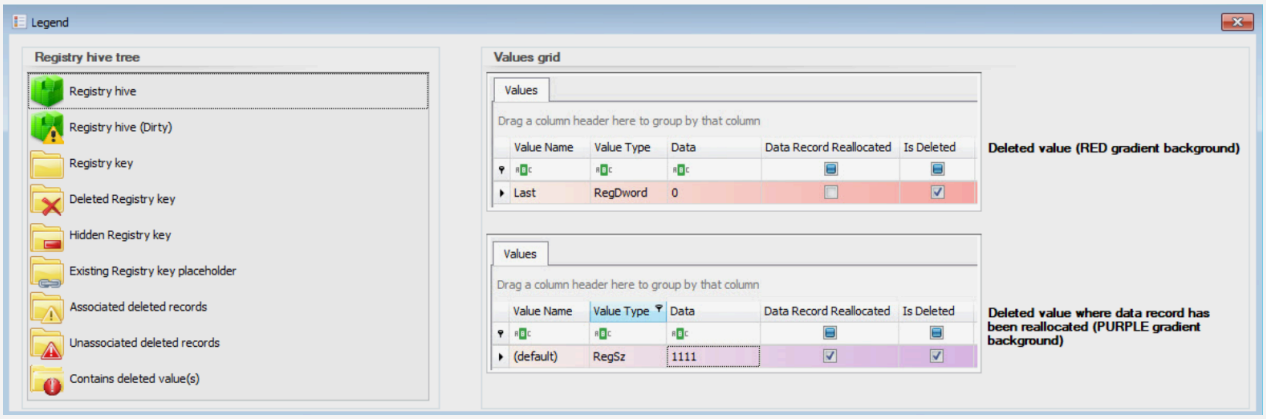
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

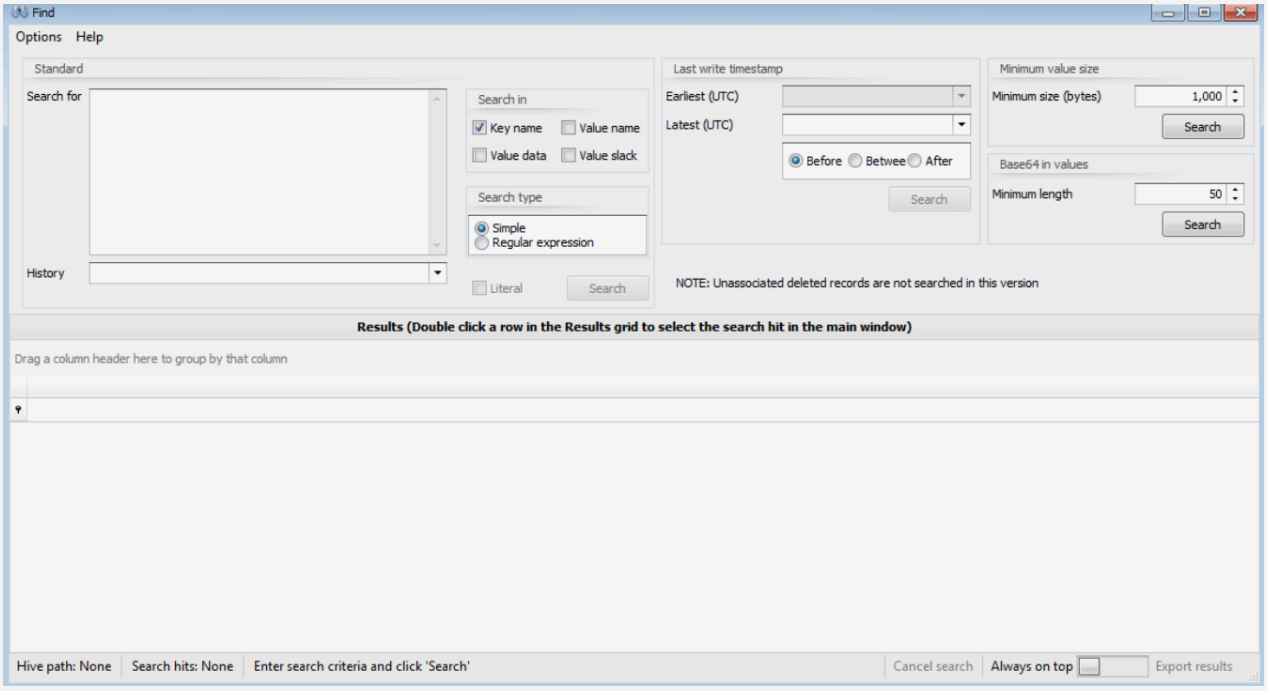
With that out of the way, let's review the most useful features (for ScreenSharing) with this utility.

Presenting icon definitions:



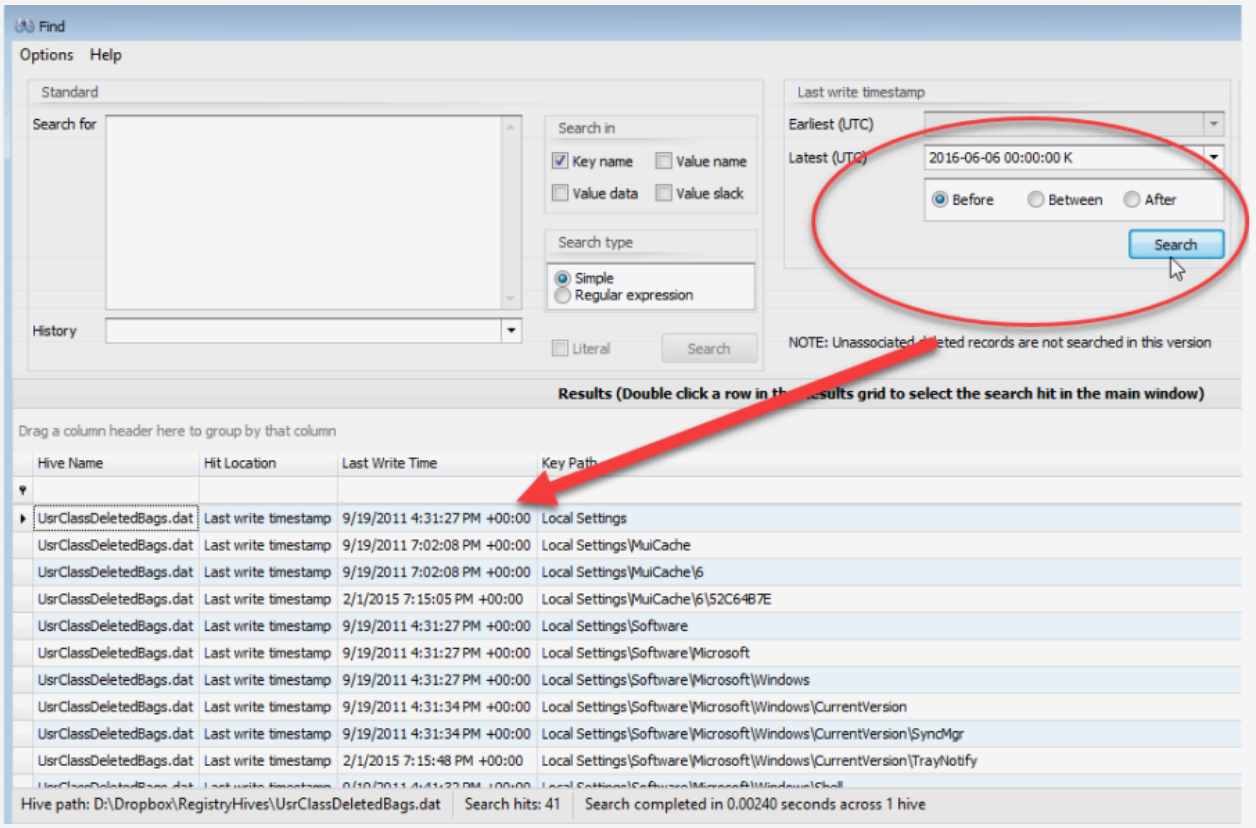
Paying attention to the last Icon in the Hive Tree section, we see a circled warning sign that we will most often see on BAM and MuiCache or ComDLG hives. What do we do and what does it mean? - it means that the given hive we are looking at includes deleted values, these values will not be displayed or highlighted in practice with BAM, however **if you see a circled warning icon on BAM or any hive - feel free to ban for it as they display only during current system boot.**

Another useful feature of Reg Explorer is the speed and ease of searching. By pressing CTRL+F we will be met with the following window:



Here, in “search for” we can type any word (string) to look for. For example, USB to look for every value containing the word “USB” or .exe or .dll or C:\ D:\ E:\ and other drive letters and much much more. Make sure to toggle value name, data and value slack in “Search in”

As explained in the original .pdf handbook, there are a lot of practical details that can be utilized while in this menu, such as searching for specific dates in UTC



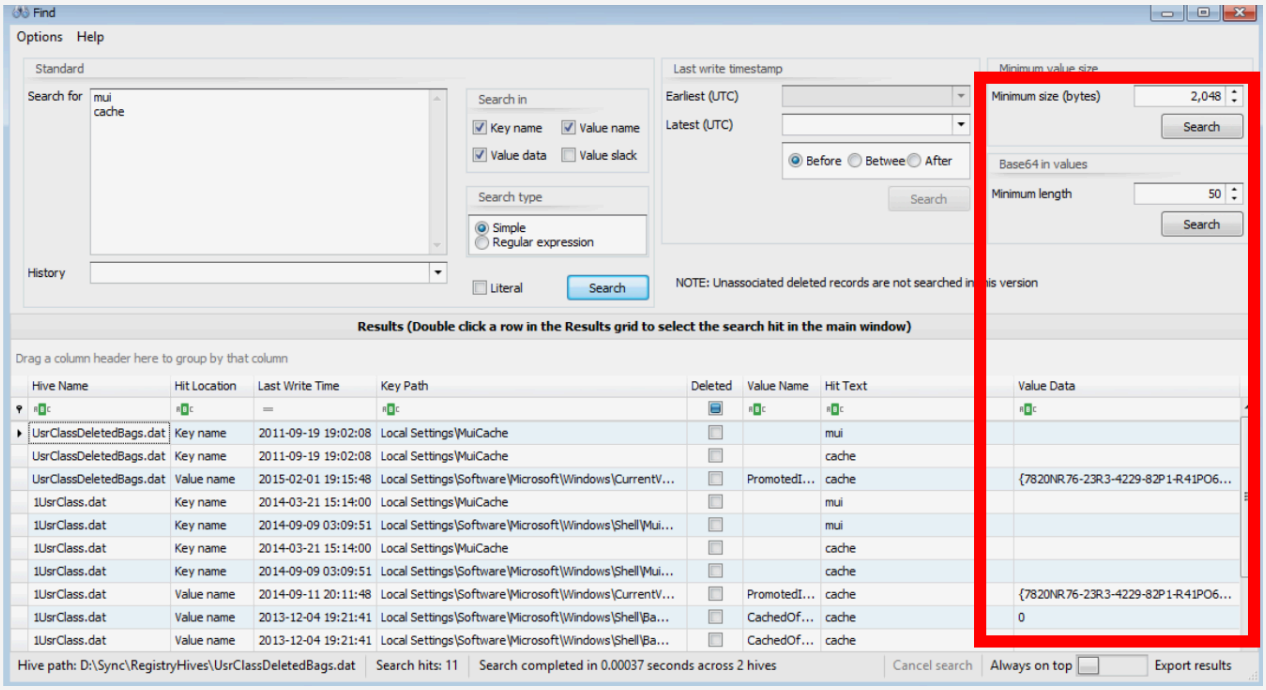
To check user's current date in UTC refer to any website(s), for example [Current UTC — Coordinated Universal Time (timeanddate.com)]();

More logical suggestions will be discussed below using source, explanations and images taken from the official handbook (comes with the download of registry explorer)

Analytic approach to value sizes

If we combine the knowledge this Chapter talked about previously regarding hunting malware behavior via Registry, human analysis can be utilized to first hunt the behavior of malwares, cheats - be it harmless autoclickers or .jar and .dll files or utilities that can be used to bypass with, we can mark their value sizes and conclude an min-max range that will ease their search. For example, let's say Karma Client's value size logged as 2,150 during first registry comparing tests and the results varied between 2,150~3,130 in a span of 10 different segments, **we can provide reg explorer with a min-max size range and analyze the results (or even export them to be used with other methods)**

More extended information about value sizes is provided in the .pdf handbook on page 48~49~50;

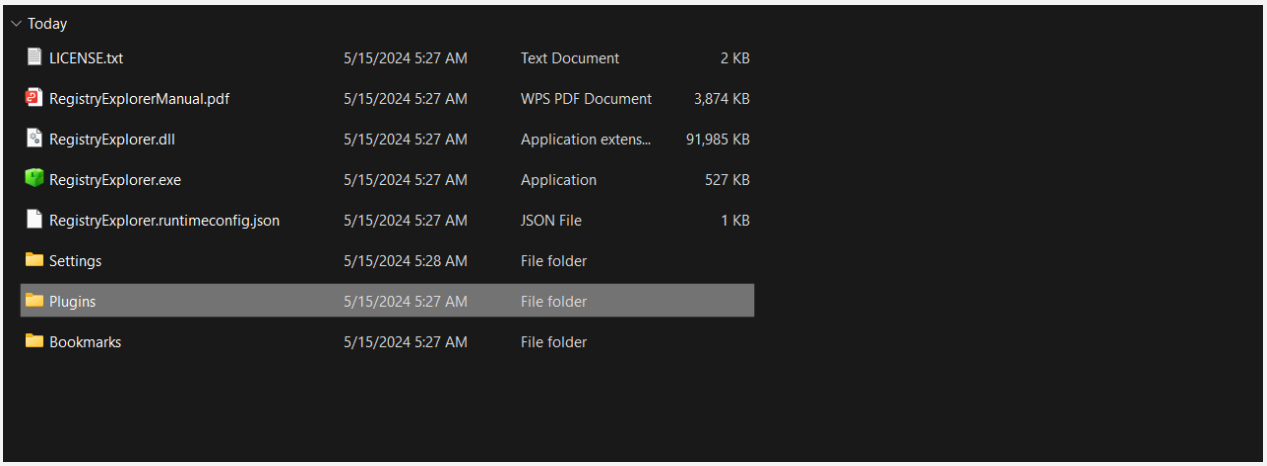


Make sure to read the official .pdf handbook as it includes more than enough information. Besides - for over 3 years the SS Communities had no idea CTRL+F was an option until I decided to read the handbook and share that piece of information, respect the authors shall we.

Adding our own plugins (for programmers)

https://github.com/EricZimmerman/RegistryPlugins

Registry Explorer relies on plugins that it comes with. It can be found in one of the folders when unzipping Registry Explorer.



The repository I linked above contains all source codes for its plugins, they'll need to be compiled into .dlls if we want to add any changes to it. Bookmarks and settings can be edited without compiling as they're just "targets" reg explorer will load upon launch.

Troubleshooting

Registry Explorer should be **run as Administrator** to attain access to configuration files. In settings, make sure to enable "recover deleted values" (or have them disabled if you want to) before loading hives, as said in the handbook of this utility - they take effect **while** loading hives, thus they don't update live and you'll have to reload all hives when making changes to them.

Make sure to run the utility from its directory, launching it via search everything might return access errors.

Information in Registry Explorer is displayed in UTC time-zone by default. For most values like ComdLG, MuiCache, ShimCache and so on - date of modification refers to the last entry in the hives. Thus if you find the date modified by ComDLG on .exes to be days before SS, it means that the last entry recorded of .exe files in ComDLG was made so long ago, thus it is not worth wasting time on going through them.

Registry Explorer's dates serve as evidence of execution. Latest entries usually start from the bottom to top.

Conclusion (Registry)

There are many ways Registry approaches - especially Registry Explorer can be combined with other utilities. For example, as Tome II Book I already explained - Registry Explorer can be used to detect unicode characters and scheduled tasks. With this in mind, every ScreenSharer can load Registry Explorer right as they begin ScreenSharing **to avoid overwriting registry data and losing evidence** for bans. As I already mentioned, registry explorer takes static snapshots of the registry and doesn't update live. This can come handy immediately just like Scheduler service and Bstrings on BAM analysis does.

Any information obtained from other evidence of execution or while parsing services with system informer can be used with Registry Explorer. For example, finding **volume** or **localhost** access during ScreenShare can be an indicator (if volume\ is in WinPrefetch for example or localhost already on the device with wsl) of suspicious behavior - data of which does not get lost in $MFT or Registry hives.

Access on files launched with scheduler still gets logged in muicache ad shimcache - thus with CTRL+F option in Reg Explorer we can instantly find all traces of specific files (for example, certain cheats found in DPS) all over Registry for more accuracy.

In the end, Registry Explorer and registry approaches can play a crucial and undeniably important role when detecting malicious behavior on the system or bypass attempts. If learnt

well, ScreenSharers will develop a general logic when conducting ScreenShare and can use Registry as their prime or last resort.

CHAPTER X. USN Journal

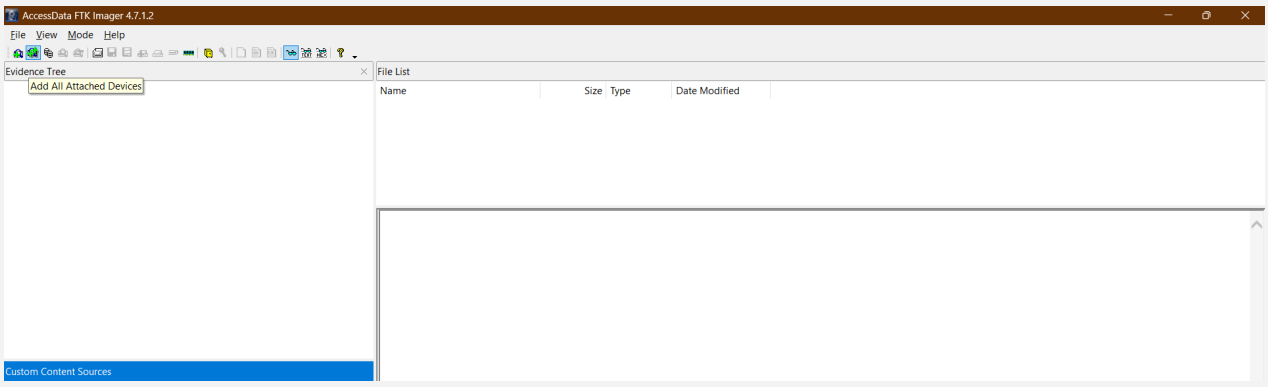
Introduction to $J

$J is a USN Journal metafile on NTFS drives (unlike FAT32) that keep all behavioral patterns of applications or initiated actions (by user or the system) on the device. For instance, shift deleting files, although don't update $recycle.bin date of modification will still be logged in USN Journal as deleted|close reason. $J is not only useful in ScreenShares but also the **only** resource ScreenSharers have when dealing with file modifications, hex changes, attribute changes and so on. $J includes alternate streams (metafiles), $UsnJrnl and $MAX in which $UsnJrnl is a "Changes Journal" that holds all the information about system changes done by the user or the system itself, whereas the $MAX file holds information regards changes of $UsnJrnl. By this, even if users manage to hook $UsnJrnl with via Kernel level access, $MAX file will still hold that information, thus Date Modified on $MAX equals date modified of $UsnJrnl that results in enough proof to ban for Bypass Attempt.

In this Chapter, we will cover the usage of $MFT with $J, discuss what can be done to analyze USN Journal and why it is the only resource advantageous over numerous bypasses.

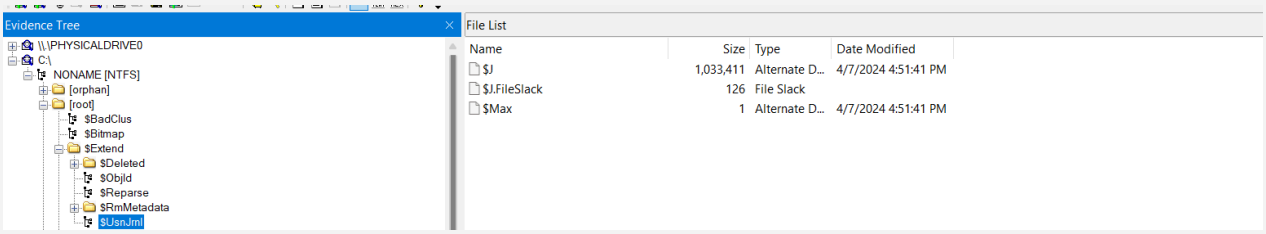
FTK Imager and $UsnJrnl

One way to view modification dates of $J and $UsnJrnl is by analyzing $MFT. However, since FTK Imager already does that and builds a file tree with directories with more potential uses, we will cover an approach to detect Journal modification (journal clearance or hooking) and $MAX file modification.



After attaching all devices to FTK Imager, navigating through the $Extend directory will lead us to the $J file holding $UsnJrnl and $MAX files.

It is important to note that by right clicking these files, ScreenSharers can extract all data of USN Journal to avoid further overwriting it or making confusing changes to them - especially when MFTECmd is capable of parsing the Journal further extracting static data more useful for gathering evidences.



In the following screenshot, Date Modified corresponds to when changes were made to the USN Journal (or the Max file) whereas the .fileslack **means that the $J file was overwritten - deleted while applying changes to it.** Hence why there are no dates of modifications on the fileslack.

Modification dates are usually sorted in UTC. Sometimes they glitch out and display absurd dates - in which case we can export them and parse them with MFTECmd (or use MFTECmd to parse the $MFT then check if that helped dates of modification) after which we'll analyze the latest and oldest entry. This way (unlike with JournalTrace) we can accurately determine when the Journal began logging changes on the system, whereas JournalTrace and many other Journal viewer utilities often display only recent day / month modifications, therefore losing potentially crucial data.

$J.slack file can sometimes hold the last bit of data $UsnJrnl logged before erasure, or if hooked - it'll hold the data that was "hidden" or as they call it "deleted" from Journal. But remember that deleting, overwriting or manipulating the given metafile ($J) is **not possible.** Attempting to do so usually bricks or crashes devices.

Uses of $UsnJrnl

Many ScreenSharers do not know it (and many have never thought of it either) but USN Journal and $MFT can be used to detect Time Stomping and Hex modifications on files. Timestomping is a technique that changes attributes (dates of modification) of files, be it their prefetch entries. This behavior logs as **Basic Info Change | Close** in USN Journal, thus any basic info change on a .pf file or .exe or any extension of a cheat shall be bannable. Hex modification logs as **Data Overwrite** or **Data Extend | Data Truncation** sometimes **Data Extend | Close** since an entire functionality of the file is modified. Let's delve deep into these anomalies and discuss with references.

Referencing the Habook forensic team:

“Since “$J” has records of all changes on files and folders in a volume including deleted files, this opens the doors for digital forensics investigators and threat hunters to empower their analysis with information such as the following:

Detection of malicious tools and bad files that were present at some point in time within the file system, which provides insight into suspicious/malicious user activity.

Detection of “Timestomping” activity, a technique used by attackers which is the alteration of timestamps of files to confuse investigators during their analysis.

Extending “Prefetch” artifacts value, which each contain the dates for the last 8 times an executable was run, which is a limitation of the artifact. This limitation can be overcome to gain the dates for more executions of an executable (Subject to the limitations of USN Journal below).”

Extending Prefetch artifacts value means that we can analyze data extending entries of Journal on prefetch traces like Conhost / Consent to determine if they were further modified - or when they were modified besides their last 8 entries. Let’s review some examples, though first:

We gather Registry hives and load up Registry Explorer at the start of the ScreenShare. We have also gathered $MFT (or installed Search Everything) with SRUM and Event Logs and kept them for references for the future; just in case - we also used PECmd to export all necessary data that we can quickly skim through and used RL Collector for every other artifact that can hold dates of execution to not waste time on their collections later:

Scenario A:

In WinPrefetchView, we see two suspicious sequences: Regsvr, RunDLL, Conhost and Consent have the last 8 entries (9 including latest modification date) only a couple minutes apart. This is suspicious and leads to assumptions of **overwriting** previous data. Using $UsnJrnl we will be able to determine all sequence entries and gather dates of modification that took place before the last 8~9 entries that are clearly meant to confuse the ScreenSharer or overwrite data in existing entries **because that way** it becomes much harder to prove injections of .dll files or renamed extensions right away. On Conhost’s example, we are met with the following:

Update Timestamp	Parent Path	Name	Update Reasons
=		conhost	DataExtend|DataTruncation|Close
2022-09-26 08:22:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:04:04	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:04:07	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:07:57	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 10:22:16	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 10:28:02	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:01:53	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:06:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:10:06	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:55:45	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:59:12	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 11:01:37	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:36:03	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:47:18	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:49:30	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:23:32	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:45:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 11:47:40	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close

Thus, by selecting beyond 8 dates of modifications on Conhost - we can copy the Update Timestamp of that and filter using that timestamp **in the** Journal itself (to see what files were modified as they will be logged) or use that specific timestamp in other evidences of execution to see which file initiated and logged in Conhost.

Scenario B:

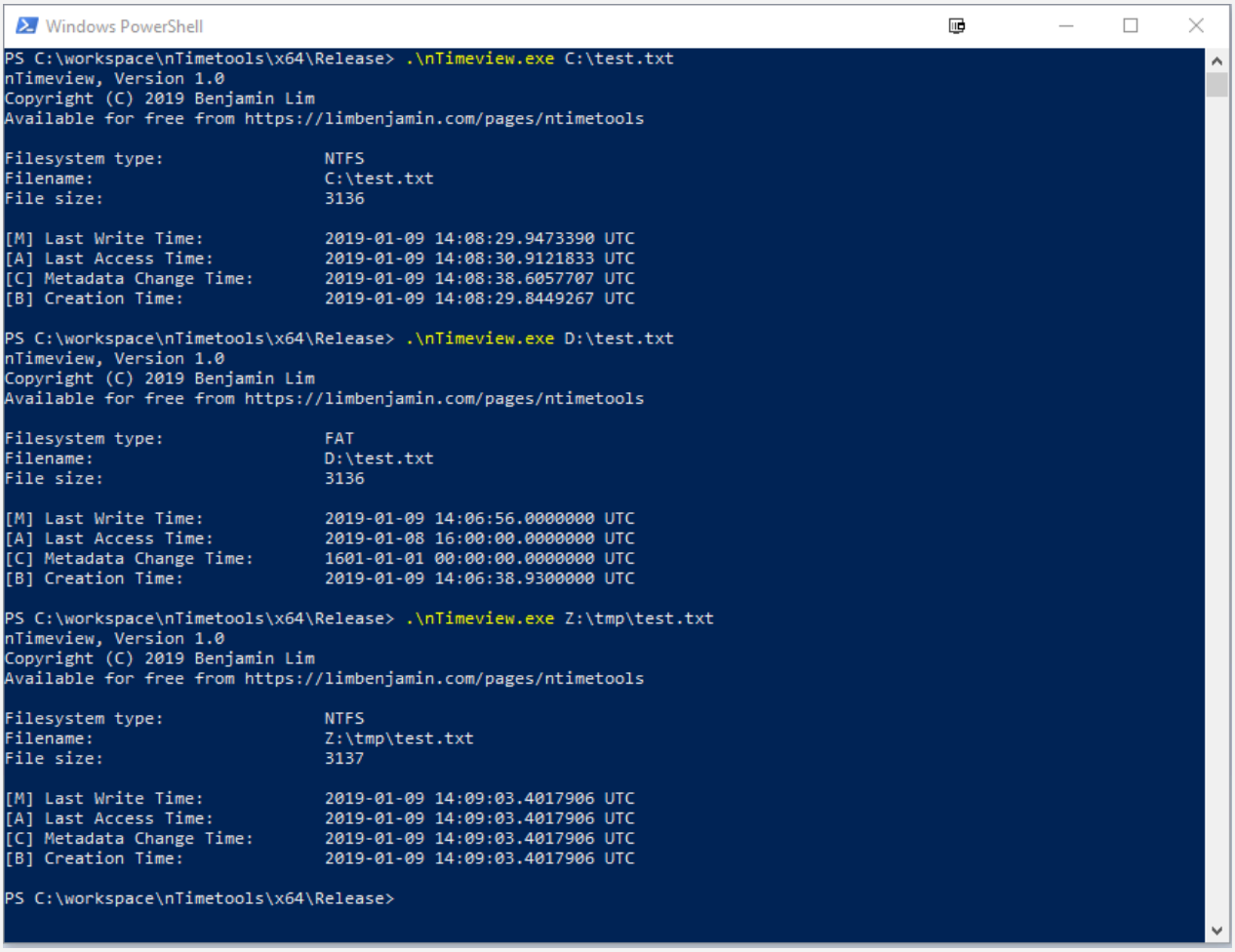
In WinPrefetchview we see a .pf trace of slinky that was executed months ago. However, the file itself was logged both in csrss and DPS. Using USN Journal with Basic Info Change | Close - we will see that .pf trace modification and conclude that the file had Read-Only attribute applied to avoid further update sequences per launch. Enough to ban for.

Scenario C:

We generally want to detect files that have their attributes modified. For example, calc process hollowing tools. We will utilize Journal the same way as in Example B and read the results.

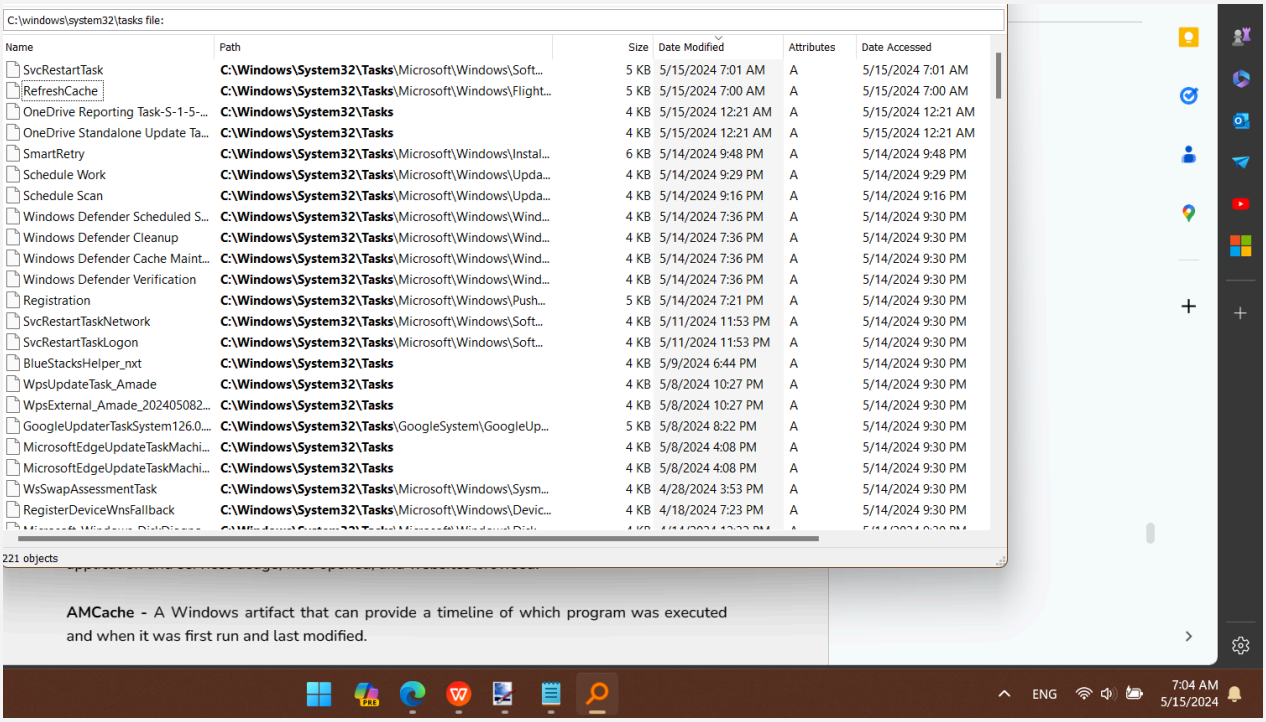
Update Timestamp	Parent Path	Name	Update Reasons
=	.\users\user\desktop\article	calcx.exe	
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	FileCreate
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate|BasicInfoChange
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate|BasicInfoChange|Close
2022-10-24 16:34:55	.\Users\User\Desktop\article	calcx.exe	BasicInfoChange
2022-10-24 16:34:55	.\Users\User\Desktop\article	calcx.exe	BasicInfoChange|Close

Changing timestamps of files can be done using any date modification application such as nTimeStamp.exe for example. Here's how it'd look:



Scenario D:

We open Search Everything and analyze scheduled tasks. We are met with the following image:



Certain tasks have dates modified 3 minutes before ScreenShare. Such behavior is usually normal with SvcRestartTask and RefreshCache though we will use them as examples for now. How do we tell if the file was modified before ScreenShare?

Solution (A)

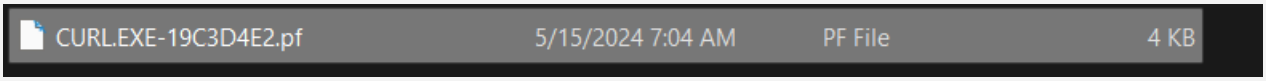
If the Bypasser used right click > open as, it logs in WinPrefetchView under **Open As**; any .exe or task logged there is enough to ban for.

Solution (B)

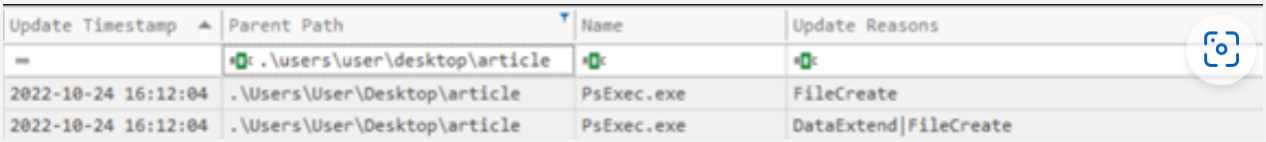
Data Extend | Close or Data Extend | Data Truncation | Close will be enough to ban for as these sequences do not update on scheduled task files by default.

Scenario E:

In Prefetch we see a Curl trace but we are met with the same problem as in Example A. Is it over? can we no longer find any evidence? are we doomed? do we ban right away? the answer is no to each of these questions. Here's why:



By analyzing Journal and finding its all previous entries (if nothing found in Curl using WinPrefetchView) we can copy the date and **use that date on File Create | Close entries!**



Update Timestamp	Parent Path	Name	Update Reasons
-	.\users\user\desktop\article		
2022-10-24 16:12:04	.\Users\User\Desktop\article	PsExec.exe	FileCreate
2022-10-24 16:12:04	.\Users\User\Desktop\article	PsExec.exe	DataExtend|FileCreate

This means a file was created. In this case - downloaded.

Scenario F:

There is a bypass spread in the community that relies on Hex modifying cheat files; this bypassed every SS Tool and every ScreenSharer on the platform. Nobody can prove anything

because it's impossible to recover the old file so they issue false bans on people who they assume use the bypass. What to do?

Refer to Journal. Data Extend | Data Truncation | Close or Data Extend | Close **especially** Stream Change | Close or Data Overwrite | Close on .exe files (or any file that displays hashmismatch and invalid signature in signature checks) **is enough to ban for as that's what hex overwriting displays.**

WhiteLotus.exe	15/05/2024 07:13:02	Data extend | Data truncation
WhiteLotus.exe	15/05/2024 07:13:02	Data extend | Data truncation | Close

Resources & Reasons Table

[Introduction to MFTECmd - NTFS MFT and Journal Forensics]

Value	Meaning
USN_REASON_BASIC_INFO_CHANGE 0x00008000	A user has either changed one or more file or directory attributes (for example, the read-only, hidden, system, archive, or sparse attribute), or one or more time stamps.
USN_REASON_CLOSE 0x80000000	The file or directory is closed.
USN_REASON_COMPRESSION_CHANGE 0x00020000	The compression state of the file or directory is changed from or to compressed.
USN_REASON_DATA_EXTEND 0x00000002	The file or directory is extended (added to).

USN_REASON_DATA_OVERWRITE 0x00000001	The data in the file or directory is overwritten.
USN_REASON_DATA_TRUNCATION 0x00000004	The file or directory is truncated.
USN_REASON_EA_CHANGE 0x00000400	The user made a change to the extended attributes of a file or directory. These NTFS file system attributes are not accessible to Windows-based applications.
USN_REASON_ENCRYPTION_CHANGE 0x00040000	The file or directory is encrypted or decrypted.
USN_REASON_FILE_CREATE 0x00000100	The file or directory is created for the first time.
USN_REASON_FILE_DELETE 0x00000200	The file or directory is deleted.
USN_REASON_HARD_LINK_CHANGE 0x00010000	An NTFS file system hard link is added to or removed from the file or directory.

	An NTFS file system hard link, similar to a POSIX hard link, is one of several directory entries that see the same file or directory.
USN_REASON_INDEXABLE_CHANGE 0x00004000	A user changes the FILE_ATTRIBUTE_NOT_CONTENT_INDEXED attribute. That is, the user changes the file or directory from one where content can be indexed to one where content cannot be indexed, or vice versa. Content indexing permits rapid searching of data by building a database of selected content.
USN_REASON_INTEGRITY_CHANGE 0x00800000	A user changed the state of the FILE_ATTRIBUTE_INTEGRITY_STREAM attribute for the given stream. On the ReFS file system, integrity streams maintain a checksum of all data for that stream, so that the contents of the file can be validated during read or write operations.
USN_REASON_NAMED_DATA_EXTEND 0x00000020	The one or more named data streams for a file are extended (added to).

USN_REASON_NAMED_DATA_OVERWRITE 0x00000010	The data in one or more named data streams for a file is overwritten.
USN_REASON_NAMED_DATA_TRUNCATION 0x00000040	The one or more named data streams for a file is truncated.
USN_REASON_OBJECT_ID_CHANGE 0x00080000	The object identifier of a file or directory is changed.
USN_REASON_RENAME_NEW_NAME 0x00002000	A file or directory is renamed, and the file name in the USN_RECORD_V2 structure is the new name.
USN_REASON_RENAME_OLD_NAME 0x00001000	The file or directory is renamed, and the file name in the USN_RECORD_V2 structure is the previous name.
USN_REASON_REPARSE_POINT_CHANGE 0x00100000	The reparse point that is contained in a file or directory is changed, or a reparse point is added to or deleted from a file or directory.
USN_REASON_SECURITY_CHANGE 0x00000800	A change is made in the access rights to a file or directory.

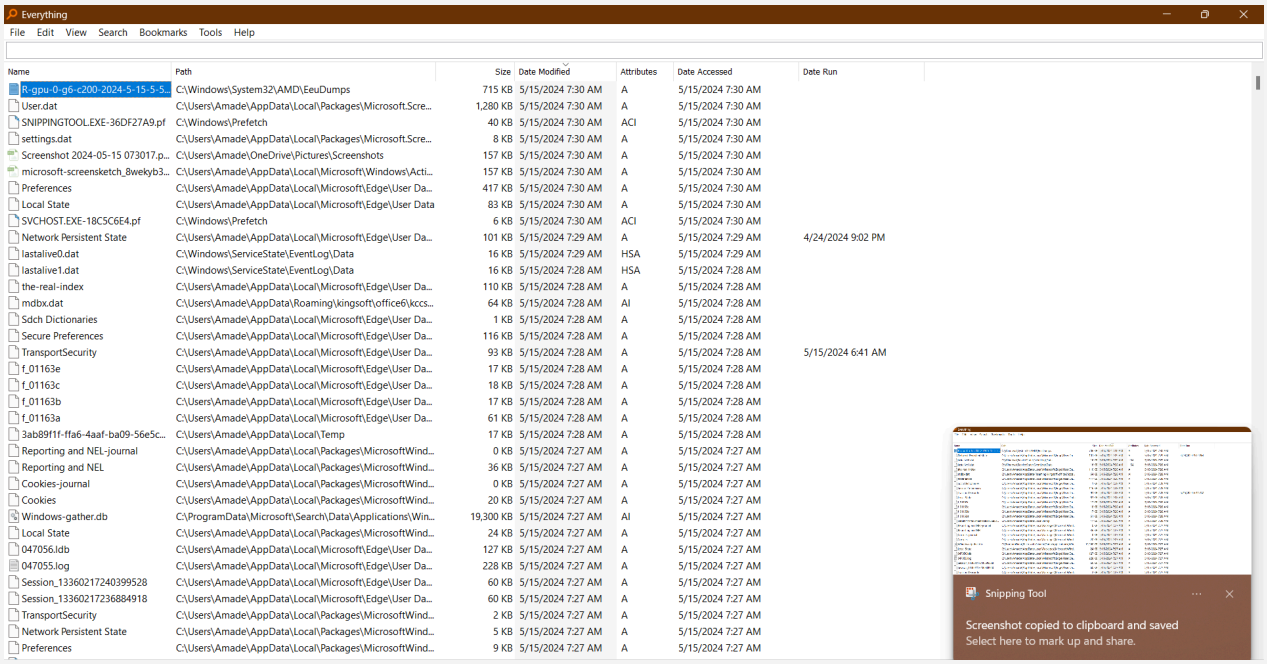
USN_REASON_STREAM_CHANGE 0x00200000	A named stream is added to or removed from a file, or a named stream is renamed.
USN_REASON_TRANSACTED_CHANGE 0x00400000	The given stream is modified through a TxF transaction.

Value	Meaning
USN_SOURCE_AUXILIARY_DATA 0x00000002	The operation adds a private data stream to a file or directory. An example might be a virus detector adding checksum information. As the virus detector modifies the item, the system generates USN records. USN_SOURCE_AUXILIARY_DATA indicates that the modifications did not change the application data.
USN_SOURCE_DATA_MANAGEMENT 0x00000001	The operation provides information about a change to the file or directory made by the operating system.

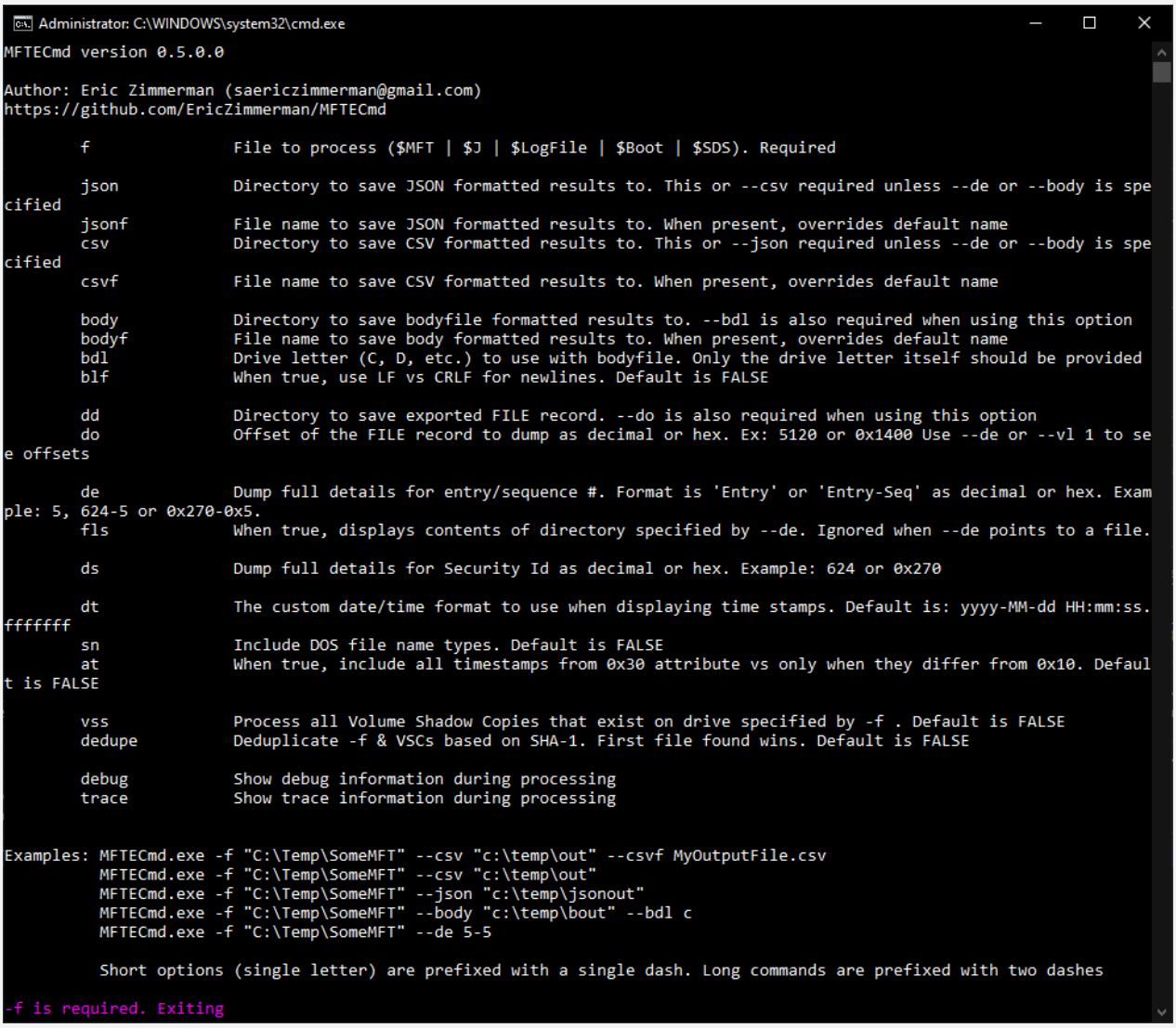
	A typical use is when the Remote Storage system moves data from external to local storage. Remote Storage is the hierarchical storage management software. Such a move usually at a minimum adds the USN_REASON_DATA_OVERWRITE flag to a USN record. However, the data has not changed from the user's point of view. By noting USN_SOURCE_DATA_MANAGEMENT in the SourceInfo member, you can determine that although a write operation is performed on the item, data has not changed.
USN_SOURCE_REPLICATION_MANAGEMENT 0x00000004	The operation is modifying a file to match the contents of the same file which exists in another member of the replica set.
USN_SOURCE_CLIENT_REPLICATION_MANAGEMENT 0x00000008	The operation is modifying a file on client systems to match the contents of the same file that exists in the cloud.

Monitoring Journal

If we want to detect a bypass or what it does to the system, we can obviously use snapshots of USN Journal (clear it, do the bypass, extract and save) over and over again. There are also utilities that monitor changes on the system and display them live. One of such utilities is.. Search Everything. Search Everything tracks the NTFS and displays live changes or updates. Thus, if we want to create a detection to the bypass - we can start by maximizing search everything in the background and **closely monitoring changes** while doing bypasses. For example, as shown in the screenshot by just screenshotting multiple changes were applied to the system - a prefetch was made and caches dates were modified, thus data was updated.



For more “closed” checks, for example to determine if replacing a file logs any special changes in USN Journal, using tools like MFTECmd or JournalTrace is sufficient. With MFTECmd, extracting USN Journal is pretty simple. Let’s review all of its switches first.

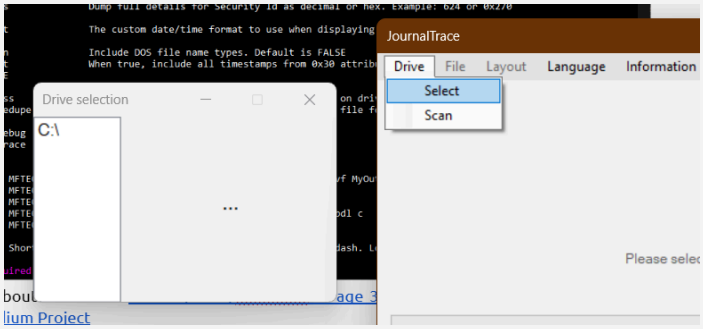


More about MFTECmd: [MFT Explorer/MFTECmd - Page 3 of 4 - AboutDFIR - The Definitive Compendium Project]

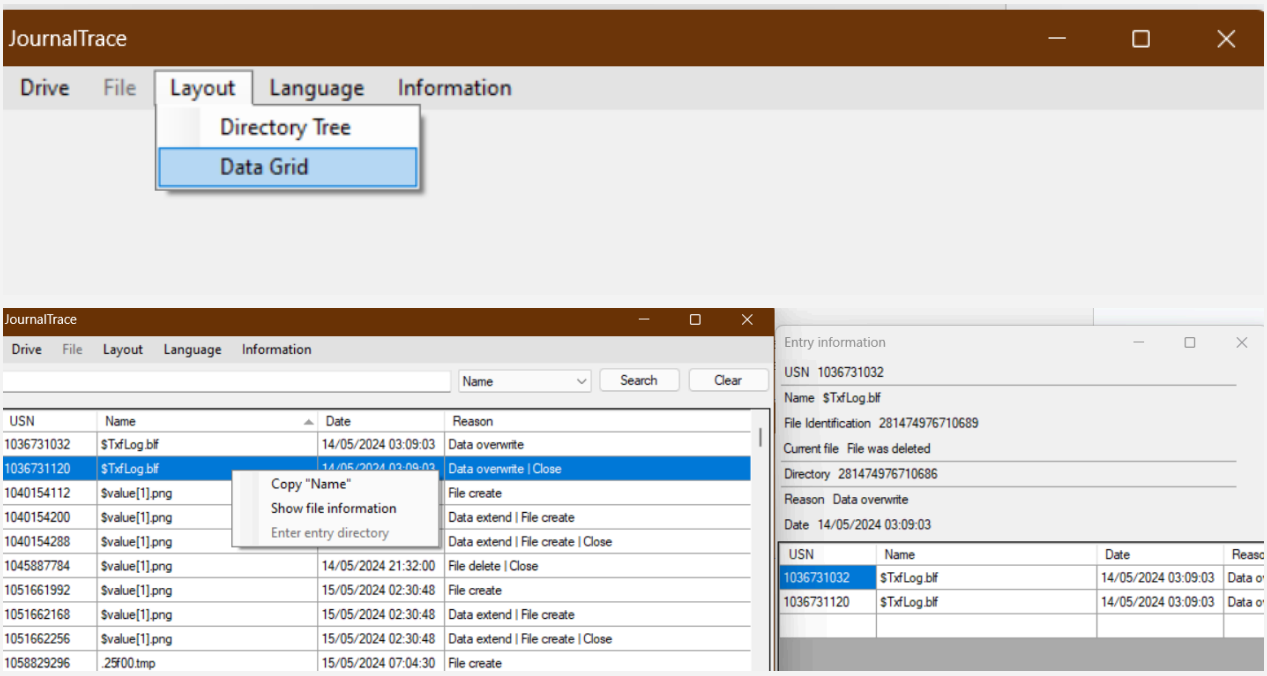
Thus, if we want to extract Journal only, we'll have to use the following command and read results in TimeLine Explorer (as shown in the 2. Video demonstration):

MFTECmd.exe -f C:\$Extend\$UsnJrnl:$J --csv .

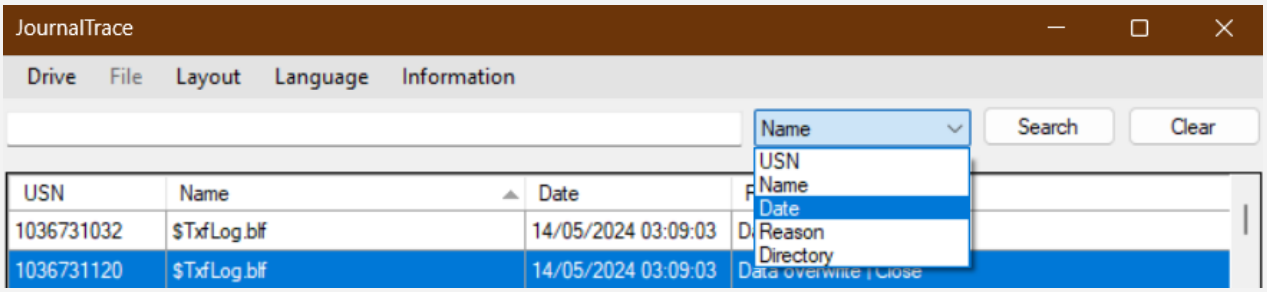
Though here's JournalTrace: [Releases · ponei/JournalTrace (github.com)]



Select a drive and scan it (you'll have to go back and forth to parse) and view the results via "Layout" > "Data Grid" (directory Tree returns a list of directories like via /dir) if you want to make changes to directories or get used to dir sequences.



For extended information you can right click on the file name and "Show file information", by right clicking you can also copy the dates and follow examples listed in above situational scenarios.



Name is for file names, Reason is for update sequences, directory is for directories and USN is for USN entry numbers (though that usually crashes JournalTrace)

You can learn more about USN Journal here:

[USN_JOURNAL_DATA_V2 - Win32 apps | Microsoft Learn]

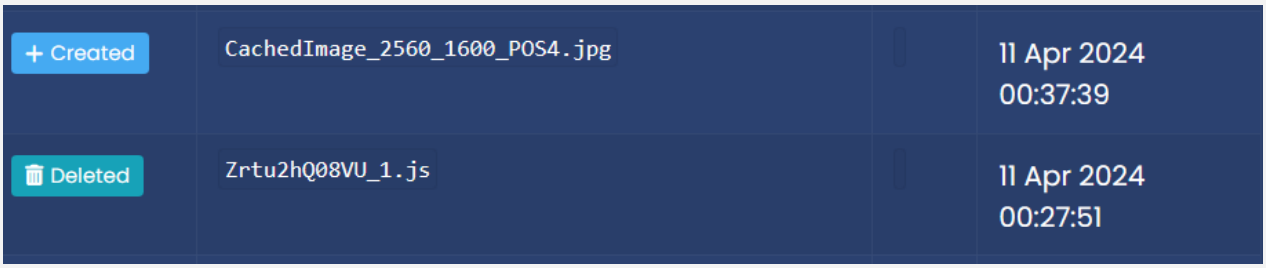
[USN_RECORD_V2 - Win32 apps | Microsoft Learn]

[USN_RECORD_V3 - Win32 apps | Microsoft Learn]

[USN_RECORD_V4 - Win32 apps | Microsoft Learn]

Human-System Operations

Although it is clear how to use Journal, how to read it, what its sequence numbers and reasons represent, still how do we tell if an action was initiated by the user and when was it initiated by the system itself? We will be using Echo scan results for these examples.



Point A:

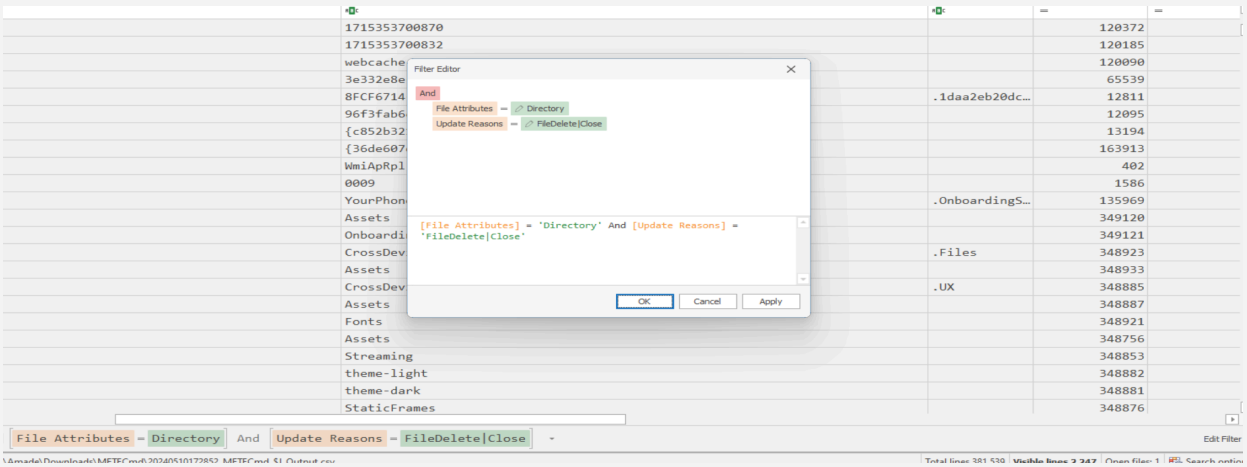
Usn Journal entries that do not include file paths are system initiated actions. For example, being me with .js or image.png[1] results in Echo (or JournalTrace) will always have either no file paths or sequence numbers for file paths (Directory);

	Directory
e: new name | Basic info change | ...	C:\Users\Amade\AppData\Local\Microsoft\Edge\User Data\Default\Network
	C:\Users\Amade\AppData\Local\Microsoft\Edge\User Data\Default\Network
	C:\Users\Amade
end	C:\Users\Amade
	283445301547631552
	283445301547631552

This can also mean that the root directory of a file was deleted. Although, practice has shown that it is very complicated to prove user initiated actions or bannable offenses without relying on other tools and evidence of execution when dealing with such scenarios. Let's review hypothetical cases.

Scenario A - Folder deletion:

When ScreenSharing, we notice a folder in the shell:recent being accessed that is no longer on the system. In Journal, we see a bunch of **non-executable** files deleted shortly before ScreenShare. Using a parsed $J .csv with MFTECmd will help us determine if the folder was deleted:



For this, we can use the following filter: **[File Attributes] = 'Directory' And [Update Reasons] = 'FileDelete|Close'**

	Update Reasons	File Attributes
	= FileDelete|Close	= Directory
8	FileDelete|Close	Directory
8	FileDelete|Close	Directory

And after that we can use System Informer (csrss, dps) or any other evidence of execution (especially Registry) to determine if the file was launched.

How do we determine if a deleted folder / file was a system operation? it will be useful to know what are considered “bad filename characters”;

Illegal Filename Characters

Do not use any of these common illegal characters or symbols in your filenames or folders:

- # pound
- % percent
- & ampersand
- { left curly bracket
- } right curly bracket
- \ back slash
- < left angle bracket
- > right angle bracket
- * asterisk
- ? question mark
- / forward slash
- blank spaces
- $ dollar sign
- ! exclamation point
- ' single quotes
- " double quotes
- : colon
- @ at sign
- + plus sign
- ` backtick
- | pipe
- = equal sign
- emojis
- alt codes

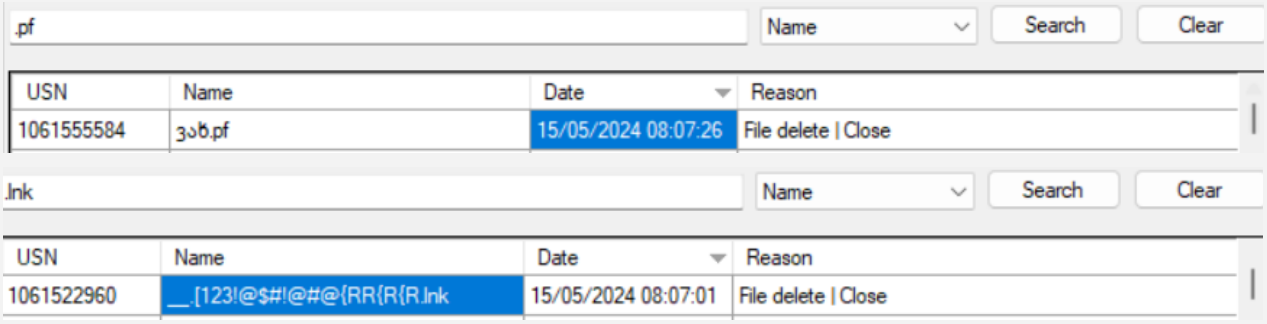
Most of these marks are **never seen in human made / named files and folders** unless of course they are aware that the ScreenSharer will rely on this logic. In such cases, to differentiate a human-system operation, we can rely on the **consistency of operations and file modifications in Journal** to conclude what really could’ve happened. For this, a combined use of Registry (or other evidence of execution) or JumpLists and Journal can be useful.

For instance, if we see a folder deletion in Journal **but the folder still exists** and while accessing the folder **we are met with warning popups** it is more likely a system operation than of the user. In which case we can try to copy files from that directory to other directories and launch them.

In short, **to distinguish user-system operations and actions, rely on evidence of execution, directory access and consistency of Journal and System Informer logs, keeping “bad characters” into account.**

Understanding Journal

USN Journal is best reliable and used when ScreenSharers already have something to cling on while analyzing system behavior. For example - no recent prefetch entries can already hint on **deletion of .pf** files and no .lnk files can hint on clearance of Recent docs or the very same .lnk files.

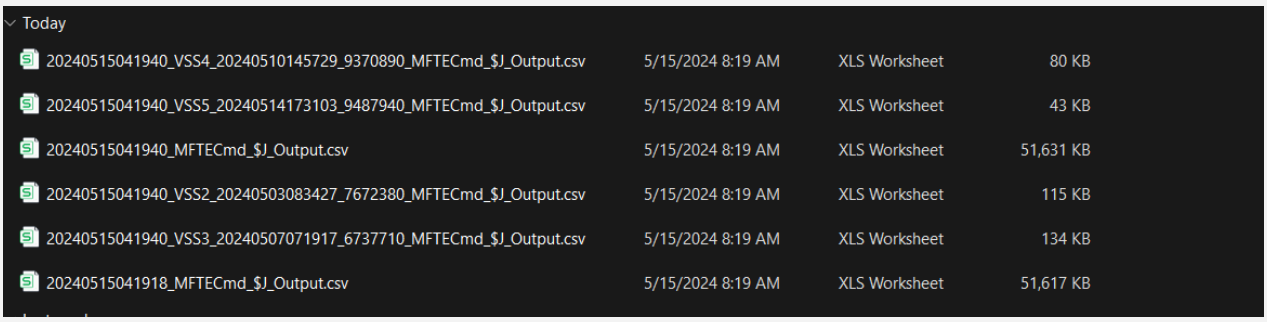


.lnk files are “link” files (thus in shell:recent) and .pf files are prefetch files - thus in the Prefetch folder. Most of this is already explained in Tome I of the Red Lotus guide (which is free) so we won’t be repeating the same information.

Unlike an unpopular opinion that unicode characters can be detected in Journal by filtering question marks, as you can see for my device, unicodes are still normally visible. Thus a conclusion: **not all unicode characters log as question marks on all systems.**

How do we get along with it? quite simple! use Timeline Explorer and exclude the entirety of the English Alphabet! well... that won’t really work, they can have extensions in English, after all... so, how can the Journal help with that and why did I bring it up? The point is, sometimes even Journal can be used to determine the full file name or file execution and modification, even if they are deleted by Alternate Data Streams.

Just search Stream Change in the MFT $UsnJrnl output and job’s done, it can even detect $DATA streams. **Oh, and don’t forget to parse VSS too MFTECmd.exe -f C:\$Extend\$UsnJrnl:$J --vss --csv .**



That way you can extract multiple $J outputs from multiple volume shadow copies of the system. Which means, if the Bypasser doesn’t know how to clear volume shadow copies then ScreenSharers will be able to (one in a thousand chances but still) recover Journal entries that are lost, overwritten or **even cleaned.** Although chances are slim and VSS usually can also auto clear itself or not create snapshots at all, it is still worth giving a try.

And yes, I didn't forget about unicode ADS, here's an example screenshot:

arent Sequence Number	Update Sequence Number	Update Reasons	File Attri
=			
393	1062144296	NamedDataExtend|NamedDataTruncation|Close	Archive
393	1062139848	NamedDataExtend|NamedDataTruncation	Archive
393	1062139672	NamedDataTruncation	Archive
393	1062055240	StreamChange|Close	Archive
393	1062054680	StreamChange	Archive

In this case, ADS modifications log as NamedData modifications (be it truncation or extend) alongside with StreamChange (while on .exe files for example it can mean file replacement - as Zone.Identifier of the file changes as well)

All and all if you cut in half entries in System Informer services, you can refer to USN Journal and filter by Name or Path - maybe you're on a system that can still read Unicode Characters.

Conclusion (Journal)

Journal is a non **bypassable** NTFS characteristic that logs all changes on the device. Changes can be traced using Search Everything (as well as other utilities found on Github) and monitored using Journal extracting utilities (Like Velociraptor, JournalTrace, MFTECmd) to develop bypass detections and understand the system better. It can often read Unicode characters and determine alternate data stream modifications, file-folder modifications and much more. Combined use of Registry and Journal can often be the only resources capable of detecting bypass attempts. Even though Journal can be cleared, it is simple to detect by $J and $MAX files via Event Logs or FTK Imager or MFTECmd. It is physically impossible to modify Journal, Microsoft itself and many. **If they do,** I quote DFIR Experts and Professors:

[You can delete or disable journaling and change journals with fsutil but that can have a big negative impact on any application using them to operate. If you disable or delete an active change journal all entries in the master file table (MFT) are changed to zero. I don't think you can actually get rid of MFT entirely, if that's what you're asking.]

Thus Journal is highly linked with $MFT and cannot be distinguished without. To fully grasp the abilities of Journaling, one would have to delve deeper into Windows systems - that is not the topic of this Tome nor within the range of our experience.

It is important to have something to cling on before analyzing Journal; if we still begin by analyzing USN Journal - we better have a snapshot taken before beginning the ScreenShare to avoid extending $UsnJrnl itself. It must be used with other evidence of execution or $MFT itself to determine things like Rundll, conhost / consent, regsvr (and so on) changes.

With that being said, it is important that ScreenSharers experiment this on their devices more and more - almost every day to fully grasp the possibilities. Get a file name, date - analyze.

Glossary

A

Anti-Forensics - Anti-forensics is more than technology. It is an approach to criminal hacking that can be summed up like this: Make it hard for them to find you and impossible for them to prove they found you.

Anti-Forensic Labs - Websites made for Red-Team Forensic students or experts to study at or experiment at.

Alternate Data Streams - A file hidden within another file. Feature on NTFS file systems (thus it will not be possible on FAT32 drives) to fork data of a specific file into another File or Folder.

Activities Cache - Windows activity history; keeps track of activity on a device, such as application and services usage, files opened, and websites browsed.

AMCache - A Windows artifact that can provide a timeline of which program was executed and when it was first run and last modified.

R

Red Team - Individuals (often hired by organizations to test the security of their networks) playing the role of threat actors, “hackers” against the Blue Teamers.

B

Blue Team - Individuals (also often hired by organizations to test the defensive capabilities against Red Teamers) who monitor and take down Red Team attacks.

“1. Registry

The registry is a database of configurations stored in "configuration" files located in c:\windows\system32\config. It comprises multiple divisions called "hives" (named as such due to historical reasons), which store branches of keys and their subkeys. Important hives include SOFTWARE, SYSTEM, and SAM. Each user has their individual hive, storing settings characteristic to that specific user.

1.1 Registry Hives

Registry hives are the divisions within the registry where data is stored. They are stored in configuration files and include hives like SOFTWARE, SYSTEM, and SAM, among others. Each user has their individual hive containing settings specific to them.

1.2 Registry Values

Registry values are data stored under subkeys in the registry. They exist in different types, including string values, QWord & DWord values, expandable, binary, and multi-string values. Registry values are displayed to the right when exploring subkeys of registry hives.

1.3 Registry Value Types

Registry value types denote the format in which data is stored in the registry. Types include REG_BINARY, REG_DWORD, REG_EXPAND_SZ, REG_MULTI_SZ, REG_QWORD, and REG_SZ, among others. Each type serves a specific purpose in storing data.

1.3.1 Registry Value Manipulations

Registry value manipulations involve linking different registry value types to achieve specific functionalities. Examples include creating expandable registry values to launch applications, which can be done using PowerShell or command-line tools like reg-add.

1.4 Registry Hivelist

Registry hivelist is a subkey within the registry that shows the sources from which Regedit retrieves its data. It provides information on the dependencies of the registry, allowing users to understand the sources of registry data.

2. Registry Forensics

Registry forensics involves analyzing the Windows registry to collect evidence and identify suspicious behavior. This process helps forensic analysts understand the system's activities and identify unusual actions that may indicate malicious activity.
2.1 Analyzing Registry Changes

Analyzing registry changes involves comparing two registry snapshots using tools like RegShot. This dynamic analysis helps detect changes made to the registry by comparing snapshots before and after system modifications or software installations. RegShot allows for easy comparison and extraction of relevant registry keys and subkeys for further analysis.

2.2 Collecting Registry Data

Collecting registry data involves using tools like RECmd to extract information from registry hives. RECmd is a command-line utility that can parse registry hives and collect data in various formats such as CSV or JSON. It can be used to search for specific strings or values within the registry and can also recover deleted keys/values.

2.2.1 Dumping Registry Data

Dumping registry data involves extracting residue data from registry hives using tools like bstrings. This can be useful for recovering deleted information before it gets overwritten. Dumping registry data allows for deeper analysis of registry contents, especially in forensic investigations.

3. Registry Explorer

Registry Explorer is a GUI-based tool for analyzing Windows registry data. It allows users to navigate through registry hives, search for specific values, and analyze registry entries. Registry Explorer comes with various features and plugins for advanced analysis of registry data, making it a valuable tool for forensic investigations and system analysis.

3.1 Analytic Approach to Value Sizes

Analyzing value sizes in the registry can provide insights into malware behavior and suspicious activities. By monitoring changes in value sizes and setting min-max size ranges, analysts can detect anomalies and potentially malicious behavior. Registry Explorer facilitates this analysis by providing tools for searching, filtering, and visualizing registry data.

3.1.1 Adding Our Own Plugins

Registry Explorer supports plugins that extend its functionality. These plugins can be developed using the provided source code and compiled into DLLs. Adding custom plugins allows users to enhance Registry Explorer with additional features tailored to their specific needs or analysis requirements.

3.1.2 Troubleshooting

Troubleshooting Registry Explorer involves ensuring proper permissions and settings for accessing registry hives. Running Registry Explorer as an administrator and enabling options like "recover deleted values" can help in retrieving accurate registry data. Troubleshooting also includes handling issues related to file paths, directory access, and UTC time zone settings.

4. Conclusion (Registry)

In conclusion, registry analysis plays a crucial role in detecting suspicious behavior and investigating system activities. Tools like Registry Explorer, RegShot, and RECmd provide valuable capabilities for collecting, analyzing, and interpreting registry data. By combining these tools with manual analysis techniques and forensic methodologies, screen sharers can effectively identify and respond to malicious activities, ensuring system security and integrity.

Introduction to $J:

$J is a metafile on NTFS drives used to log behavioral patterns of applications and system actions.
It records actions like file deletions, attribute changes, and hex modifications, providing valuable insights for forensic analysis.
$J includes $UsnJrnl (Changes Journal) and $MAX files, which hold information about system changes and modifications to $UsnJrnl.
Using FTK Imager and $UsnJrnl:

FTK Imager can be used to analyze modification dates of $J and $UsnJrnl, providing insights into system changes.

Extracting data from $UsnJrnl is crucial for forensic analysis, especially when using tools like MFTECmd to parse the Journal.
Detection of Malicious Activities:

$UsnJrnl can be used to detect malicious activities like timestomping and hex modifications on files.
It helps in identifying suspicious user activities, detecting changes to prefetch artifacts, and identifying files with modified attributes.
Monitoring Journal and Detection of Bypasses:

Utilities like Search Everything can monitor live changes in the system and detect bypass attempts.
Tools like MFTECmd and JournalTrace are useful for extracting and analyzing data from $J.
Volume Shadow Copies (VSS) can be utilized to recover lost or overwritten Journal entries.
Understanding Journal Entries:

Journal entries without file paths are typically system-initiated actions, while those with file paths are user-initiated.
Understanding the different types of Journal entries helps in distinguishing between user and system operations.
Conclusion:

The $J metafile is an essential resource for forensic analysis, providing detailed information about system changes and user activities.
While it can be cleared, its integrity is crucial for system operation, and attempts to modify it are highly discouraged.
Combining $J analysis with other forensic techniques enhances the ability to detect and analyze bypass attempts and malicious activities effectively."

References

https://www.ired.team/offensive-security/persistence/dll-proxying-for-persistence
https://artifacts-kb.readthedocs.io/en/latest/sources/windows/ActivitiesCacheDatabase.html
https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/default.aspx
https://yara.readthedocs.io/en/latest/
https://github.com/Idov31/MrKaplan?tab=readme-ov-file
https://www.exploit-db.com/searchsploit
https://twitter.com/gazambelli/status/1005170301355864065/photo/1
https://artifacts-kb.readthedocs.io/en/latest/sources/windows/ActivitiesCacheDatabase.html
https://kacos2000.github.io/WindowsTimeline/WindowsTimeline.pdf.
https://cyber.gouv.fr/sites/default/files/2019/01/anssi-coriin_2019-analysis_amcache.pdf
https://andreafortuna.org/
Advanced “USN Journal” Forensics — Haboob
The Windows USN Journal :: Velociraptor - Digging deeper!
Windows 10 Journaling | Tom's Hardware Forum (tomshardware.com)
USN Journal (forensafe.com)
fsutil usn | Microsoft Learn
Digital Forensics – NTFS Change Journal | Count Upon Security
USN Journal (forensafe.com)
Structure of the Registry - Win32 apps | Microsoft Learn
Windows Registry (What It Is and How to Use It) (lifewire.com)
Understanding the Registry on Windows - Information Technology - UConn Knowledge Base
Understanding Windows Registry, Creating Keys and Values (maketecheasier.com)
Information Technology - UConn Knowledge Base
Windows Registry Demystified: What You Can Do With It (howtogeek.com)
Windows Registry Structure: Understanding Keys, Values, and Hives in Windows Registry | by Arunkl | TheSecMaster | Medium
What Is the Windows Registry (and What Can You Do With It)? (helpdeskgeek.com)

Index

Governing Principles.....**2**
Anti-Corruption Measures.....**3**
Introduction.....5
Instant Links.....5
Video Guides.....5
Basics of ScreenSharing.....**6**
Evidences of Execution.....**8**
Prefetch.....8
B.A.M.....8
AppCompatPCA.....8
ExecutedProgramList.....9
LastActivityViewer.....9
JumpLists.....9
Common Methods.....**10**
System Informer.....10
Explorer.....11
Csrss.....12
Renamed Extensions.....12
Search Everything.....12
JournalTool.....13
Task Scheduler.....15
USB Methods.....**16**
FAT32 Deletions.....16
USB Removals.....17
Linux SSing.....**18**
Introduction.....18
Terminal.....19
Directories.....20
Opening Minecraft/.Jar files.....**21**
Modified recycle bin.....**23**
Installed applications.....**23**
Current user.....**24**
Regedit.....**24**
Fast access.....**25**
Restarting shell before SS.....**26**
Modifying files.....**26**
Unplugging USB.....**27**
MacOS SSing.....**29**

Introduction..... 29
Chapter VI..... 30
History of Bypasses📑..... 30
First wave..... 31
Read-only rundll & regsvr..... 31
Alternate Data Streams..... 32
Unicode Characters..... 33
Camouflaged Cheats..... 34
Cloud-stored Cheats..... 35
Steganography..... 35
RunPE (VBA)..... 35
Suspending Threads & Tokens..... 36
Disabled reg inheritance & telemetry..... 36
Chapter VII..... 37
📑Anti-Forensic Labs..... 37
Anti-Forensics..... 38
OpenStego & Fake Signatures..... 38
Villain..... 39
DLL Proxying..... 39
DLL Hijacking..... 39
COM Hijacking..... 40
Extensionless Files..... 40
Shellcode Injection..... 41
Powershell Remoting..... 41
HTA exploit..... 41
Shellcode Exploit..... 41
Exploits..... 42
Red Team PL..... 42
Chapter VIII..... 43
Blue Team DFIR🧿..... 43
Digital Forensics..... 43
Activities Cache..... 44
AMCache..... 45
Windows Tasks..... 46
EVTX..... 49
VSCode..... 62
Mr.Kaplan..... 63
YARA..... 64
Customizing Collectors..... 68
ADS Detections..... 70

CHAPTER IX. Windows Registry....73
1. Introduction....73
Registry Sources and Characteristics....73
Registry Values....74
Registry Value Types....74
Registry Value manipulations....76
Registry Hivelist....78
Registry Forensics....78
Analyzing Registry Changes....80
Collecting Registry data....82
Dumping Registry data....85
Registry Explorer....85
Analytic approach to value sizes....88
Adding our own plugins (for programmers)....89
Troubleshooting....90
Conclusion (Registry)....90
CHAPTER X. USN Journal....91
Introduction to $J....91
FTK Imager and $UsnJrnl....91
Uses of $UsnJrnl....92
Resources & Reasons Table....97
Monitoring Journal....103
Human-System Operations....106
Understanding Journal....108
Conclusion (Journal)....109
Glossary....110
References....114
Index....115
Authorization....117



Authorization

Author: UnMonsieur (Konstantine-Alexander G.)

All rights reserved

Following material shall not be shared or sold to third parties without the author's permission.

04

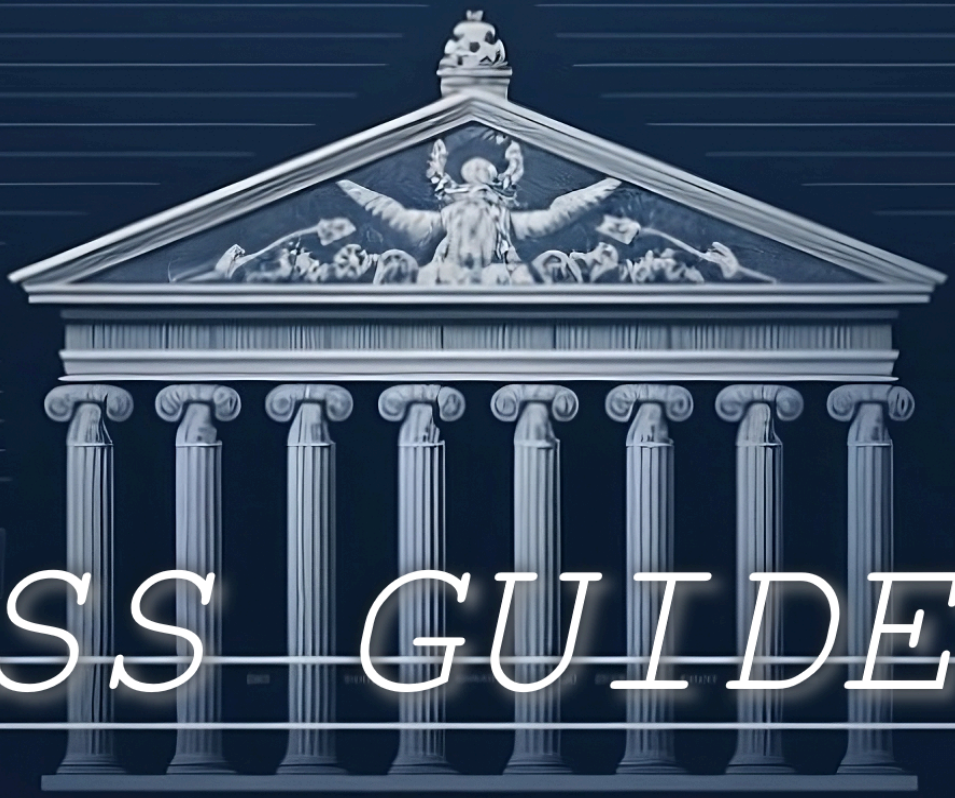
Tome II - Book II

47 original pages / reproduced in full

[RL] SS Handbook Tome II, Book II (4).pdf

Original source content begins on the next page. Author credits and illustrations are preserved.

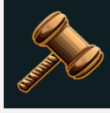
Book II



SS GUIDE

Tome II

Author: UnMonsieur | [RL HQ]





Governing Principles

This Guide is established upon "norms and regulations for the selection and proper implementation of ScreenShare teams within servers, communities, individual teams";
suggested for no imperative reason for implementation by Red Lotus Unity.
It is deemed by this Unity that the security and safety of players is more important than the accuracy and speed of ScreenShares, whether advantages are detected or not.
It is deemed by this Unity that the importance of data security is imperative over anything else that may favor Servers, Communities, Individual SS Teams.
It is condemned by this Unity to disestablish institutions regulating fair-play and privacy measures ScreenShare Teams are guided by, within communities.
It is deemed by this Unity that the experience and specialization of so-called "ScreenSharers" (other: PC Checker, Account Reviewer. . .) shall be tested and approved by positioned individuals responsible for functioning of the Team (SS Management, Administration, Leadership, e.t.c).
It is deemed by this Unity that every server implementing or copying this document, regulations, official Red Lotus regulative publications and suggestions, is responsible for building a system in complete accordance with the Unity of Red Lotus, governed by the Red Lotus Charter and Red Lotus Charter of Establishment.

~ UnMonsieur | Headquarters of Red Lotus

PREFACE

The following material is a continuation of Red Lotus Tome II, Book I - Elaboration of the techniques developed by Blue Team DFIR experts (or community members) for the development of ScreenShare practices. Furthermore, this handbook offers practical experience and understanding of this (ScreenSharing) custom from the strongest veterans of Red Lotus.

CHAPTER I. Windows Registry

1. Introduction

Most often do beginners and intermediate ScreenSharers struggle with Registry Explorer after they're done exhausting all methods for one or two checks they are aware of or have learnt from "Advanced" ScreenSharers. Veterans of the community are undoubtedly skillful and experienced, but those blinded by catching too many bypassers (most of which tend to be weak) have stopped learning as early as 2021~2022 and currently it is hard to find anybody who can recommend this utility to be used at its fullest extent.

Before I explain what Registry Explorer does or what its alternatives are, let's discuss a thing or two about Regedit itself.

Windows Registry is an apparent database of system or application settings. Upon installing Windows, for example, JumpLists and Diagtrack will have their settings preset to be disabled per installation if users refuse tracking functions of Windows, or, for instance, installing scripting environments (VSCode, Golang) as well as even text editing applications or DLL injectables (Notepad++ or HxD; RemoteDLL64) will also generate their shell application settings, boot on startup settings and so on, these will create Registry Entries.

Many online sources say that via Registry users are able to make high or low-level changes to the system, take it for granted, even malware utilizes Registry to make elevation-level (so, high) changes to the system, prime example being the UAC bypass, disabling inheritance and much more.

Contrary to that, applications, services and utilities like system optimizers (Hone for example) or any open-source github projects usually and most definitely ask users to make registry backups before initiating changes to the system settings because of irreversible damages they may hold on certain devices. *For example in the early 2018~2019s when creating custom OS themes was a great trend, many msi installers of windows themes would brick user devices because of how negatively their system settings reacted to registry modifications.*

1.1 Registry Sources and Characteristics

Registry data is stored in "configuration" files located in c:\windows\system32\config. The directory stores databases for "registry hives", *[they're called hives because the original windows NT developer... hated bees.]* Essentially, the registry is separated into multiple divisions that store branches of keys and their subkeys maintained under hive (or in other words - configuration)

files. These Hives are SOFTWARE, SYSTEM, SAM. . . Each individual user is defined under their individual Hives that store settings characteristic to that specific user. For example, if there are two separate users on the same device, each user will share the given User hives and display their configurations. Following that point, if user A was to install RemoteDLL64, user B would be able to view effective Registry configurations relevant to A's settings by viewing their user Hive. **Details like these come handy when looking for Ban Evaders and Alts as this logic doesn't require invading privacy further by checking emails, even if nobody really does or should be doing that.**

1.2 Registry Values

Registry Value is data stored under subkeys. Values exist in different types, string values, QWord & DWord values, expandable, binary and multi-string values. These are "files" displayed to the right when enrolling subkeys of registry Hives. Lifewire perfectly explains what each value stands for and what their functionalities are. In practice, rarely but realistically ScreenSharers come across Value manipulations of BAM, PCA, Activities Cache, JumpLists, Recent Docs and so on. In my own practice an expandable Variable was used, more like created by the user to launch cheats via Win+R; how did it work? **creating an expandable Registry Value means that any application or user action initiating the pre-defined setting (in this case - launching an executable) will be "expanded" into whatever the user had specified it to do.** So, if the Value is **REG_EXPAND_SZ** set "SHELL:RECENT" (just an example, modifying Windows shell values shouldn't be possible but who knows...), that is meant to launch %USERPROFILE%\DOCUMENTS\CHEAT.PDF, it'll launch that CHEAT.PDF file every time shell:recent is called, in other words, when calling SHELL:RECENT then the defined variable will be expanded, launched.

1.3 Registry Value Types

[A registry value can store data in one of several formats, such as a string value or an integer value. When you store data in a registry value—for instance by calling the RegSetValueEx function—you can indicate the type of data being stored by specifying one of the types in the table below. When you retrieve a registry value, functions such as RegQueryValueEx use these types to indicate the type of data retrieved.] (source) | Registry value types - Win32 apps | Microsoft Learn

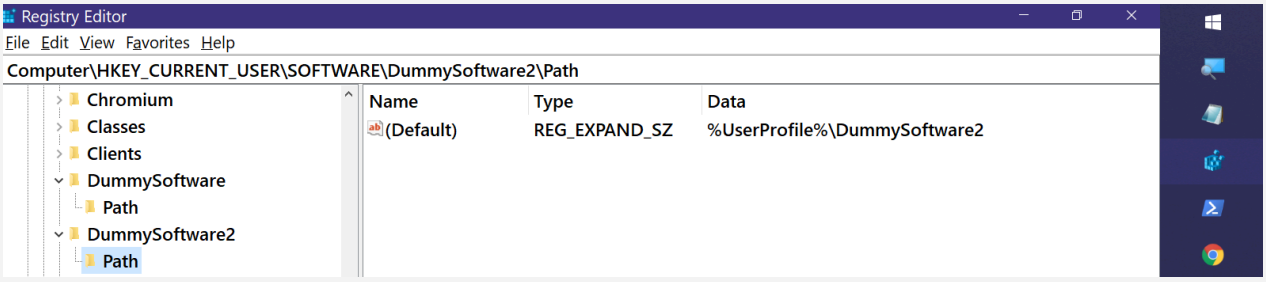
`REG_BINARY`	Binary data in any form
`REG_DWORD`	A 32-bit number.
`REG_DWORD_LITTLE_ENDIAN`	A 32-bit number in little-endian format.

	Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_DWORD in the Windows header files.
`REG_DWORD_BIG_ENDIAN`	A 32-bit number in big-endian format. Some UNIX systems support big-endian.
`REG_EXPAND_SZ`	A null-terminated string that contains unexpanded references to environment variables, for example, %PATH%. It's either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions. To expand the environment variable references, use the ExpandEnvironmentStrings function.
`REG_LINK`	A null-terminated Unicode string that contains the target path of a symbolic link that was created by calling the RegCreateKeyEx function with REG_OPTION_CREATE_LINK.
`REG_MULTI_SZ`	A sequence of null-terminated strings, terminated by an empty string (\0). The following is an example: String1\0String2\0String3\0LastString\0\0. The first \0 terminates the first string, the second-from-last \0 terminates the last string, and the final \0 terminates the sequence. Note that the final terminator must be factored into the length of the string.
`REG_NONE`	No defined value type.
`REG_QWORD`	A 64-bit number.
`REG_QWORD_LITTLE_ENDIAN`	A 64-bit number in little-endian format. Windows is designed to run on little-endian computer architectures. Therefore, this value is defined as REG_QWORD in the Windows header files.
`REG_SZ`	A null-terminated string. It's either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.

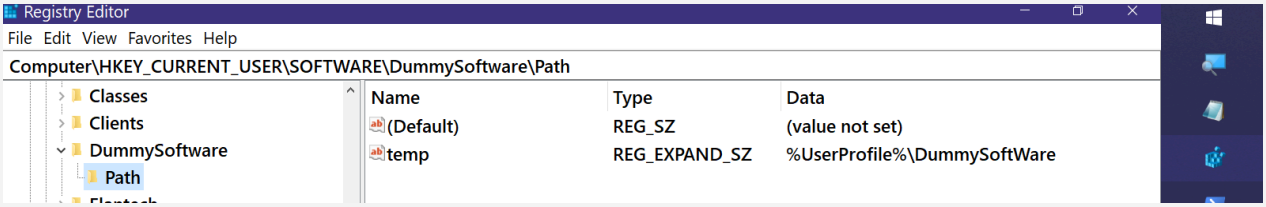
1.3.1 Registry Value manipulations

Linking Registry Value types is no accident. Let's discuss the bypass I mentioned above, the expandable Registry Value that creates a launchable application. To demonstrate it before explaining, using Powershell to create an expandable value that launches applications is quite simple. I'll cite an experiment done by Keith Miller. Here's the script Keith came up with:

```
$Splat = @{ 'Path' = 'HKCR:\SOFTWARE\DummySoftware2\Path'
'Value = '%UserProfile%\DummySoftware2'
 'ItemType' = 'ExpandString'
 'Force' = $False }
 New-Item $Splat
```



With this approach, it is interestingly possible to create launchable Executables through the temp env. for example;



Let's also review what author of the post asked prior to Keith's answer; in this blog from 2021 we encounter an expandable value manipulation using the reg-add command, syntax of the command looks like this:

reg add <keyname> [/v valuename | /ve] [/t datatype] [/s separator] [/d data] [/f] [/reg:32 | /reg:64]

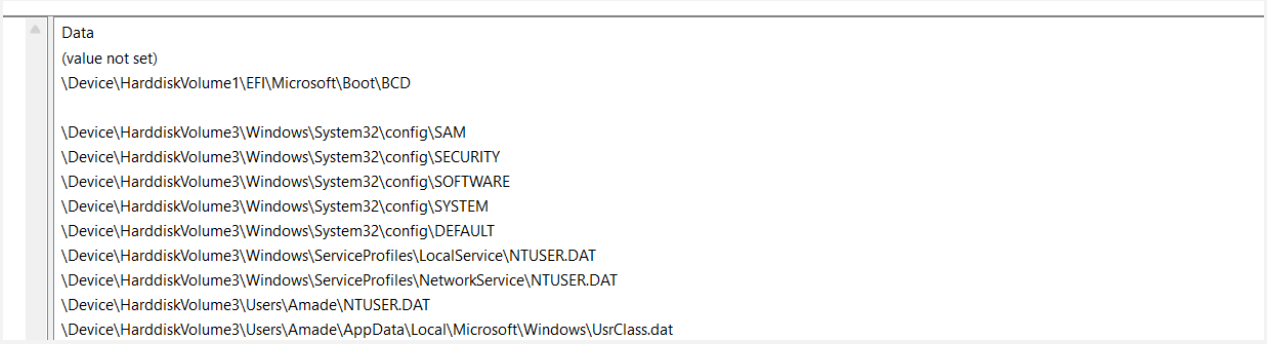
Following the Syntax, we'll first need to choose a Key (HK) like the author did as an example - HKCR\Drive\shell\Cmder\command; then, we'll set the Value name after /v (which stands for value), for example (Default) which is also shown in the previous image or whatever, let's call it (Whatever), /t stands for the data type so we'll specify it to be REG_EXPAND_SZ as shown in the previous image too with /d - data, to specify which data to expand to when a function is called. More parameters can be viewed at "reg add | Microsoft Learn";

All and all the final product looked like this: **reg add HKLM\Software\Ohmygoodness /v Path /t REG_EXPAND_SZ /d ^%systemroot^%\cheatfile.exe**

If all of that still sounds confusing, unfortunately there are practically no video demonstrations of this function online, so refer to sources that this page was based on windows - How to use REG_EXPAND_SZ from the commandline? - Stack Overflow **|** windows - Creating registry key with expandable string value named (Default)? - Super User

1.4 Registry Hivelist

Before we proceed further, it is important to know of this verifiable ability of checking what the given Regedit sources its data from, so just like with Last Activity Viewer, for example we can view which Registry keys or “dependencies” it gathers its data from to display - registry also has a Subkey that shows all of its sources and that is HKLM (Hkey Local Machine) subkey under Control **Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\hivelist**



After being told of this, it all comes down to a more understandable way of explaining what registry really is. As we can see, **registry appears to be a database of configurations applied by system or user-added files on the device that both the system and numerous other applications, as well as user-added applications use as their settings of behavior. Same goes for services as they tend to query registry entries to function, which is why it is possible to disable some of them by modifying Registry Values.**

2. Registry Forensics

Forensically analyzing any Windows artifact means **collecting evidence items** to further prove suspicious behavior, like paying attention to detail on persistent event IDs, looking at system behavior after CMD in USN Journal or $MFT and so on. It is often enough to identify these unusual chains of actions that regularly should not be occurring on the system, the challenge is to identify them correctly. To develop that “correct” understanding and logic, a ScreenSharer must first learn general ideas behind artifacts and functions of the system. Let’s review the basis, typicalities of Windows Registry.

Regedit.exe in Prefetch is not normal Windows behavior, viewing it in BAM or Prefetch usually means that the Player had accessed it some time during the given date. To speculate what could have happened, first it is worth noting that Regedit “remembers” last accessed keys and subkeys, checking user actions in regedit itself is practically impossible unless the ScreenSharer opens regedit that players forgot to close subkeys of. Therefore, paying attention to files before and after regedit.exe, as well as uptime of regedit (SRUM time or activities cache focus time) can indicate suspicious behavior. The date of regedit execution can be used for $UsnJrnl analysis or to deduct actions that followed. For example, seeing

Regedit.exe being accessed before the game (FiveM, Minecraft), uptime of which lasts until before AnyDesk, we can speculate that the user launched the game with regedit to hide evidence of its usage. Thus it is only logical to begin analyzing regedit with utilities like RECmd, Bstrings or Registry Explorer.

Sometimes it is impossible to check the uptime of applications or it takes too much time to do so. In such cases what the ScreenSharer shall orient on is the general information easily gathered via temp, recent and prefetch directories, matching times, speculating what could have happened and what to look for.

Reg.exe is a CLI based registry application preinstalled on Windows. As shown in previous examples, it can be used to modify registry values. Usually it's only the user who can launch this application, it'll log in prefetch that can be used as an indication of something being modified in Registry itself. Something removed, for example. Recent day bypasses use reg.exe to clear data from BAM because their bypasses rely on executable cheats, all of which logs in BAM. Diagtrack service can often display what command was initiated. However, when diagtrack is turned off and there is no real way of checking commands history, logically building an evidence-based conclusion is absolutely enough to issue penalties for. After all, no good ScreenSharer should let go of someone who has accessed reg.exe after CMD with CURL logs in Prefetch (or deleted which logged in Journal), before AnyDesk or near the game.

It is important to determine whether or not these applications were actually launched by the user. Reg.exe and Regedit.exe are 99% of the time **always run by the Player** but, how do we tell what is right? here is a checklist to keep in mind when analyzing uncertainties:

Pay attention to execution dates - If for example reg.exe is logged under exact hours, minutes of some other applications (for example driver booster) then it is logical to assume this was not done by the user. If for example regedit.exe was launched after accessing any desk or running scanners like Ocean or Echo then it is a fact that the user did not run them.

Link one behavior to the other - Nothing on Windows devices is a separate action. Even deleting a file has an alternative to go into the bin or not via shift deleting. Both of these actions create pop-up windows. It is logical to assume every single action leaves traces elsewhere. This is the case for injections and executions as well. Even when an application crashes it creates hang and crash logs be it in crashdumps or event logs.

Pay attention to detail - Every tiny bit of information is a detail. Distinguish between necessary and unnecessary details. For instance, when looking for renamed extensions, every extension is a necessary detail except for .dlls and .exes or .jars or whatever initial extension a ScreenSharer is looking for. It is also a necessary detail to look at execution dates and sort by date modified practically everywhere except for $MFT and its different access columns. So is knowing that Ocean generates a string scanner to avoid Ocean killers under different characters and so are paths necessary details.

Memorize or understand path permissions - Not every path can be modified by the user. Sometimes file paths can be corrupted or require special privileges to be accessed. This will be the case when trying to open Registry Explorer through Search Everything, as the file requires Admin perms and Search Everything's meant to only display applications. Paying attention to paths is exceptionally necessary when looking for things like ADS or RunDll/Regsvr (or any other prefetch) injections. For example, RUNDLL will always load Globalization or WOW64/Sytem32 applications first thus a ScreenSharer shall look for files run from paths

besides the two. It is also important to pay attention to Index in Prefetch as most cheats load their cheat file as 4th in the list of Index thus it becomes a bit easier to detect renamed extensions.

Remember registry - Whether it's a file execution, dll injection, value modification or addition, folder access, shell folders access, user logon or application hang, scheduled task creation, modification or deletion, unicode character files or folders, jumplists or alternate data streams - registry logs it all. Although sometimes temporarily, it still logs everything. When there is no other option left, registry approaches often come in handy the most.

2.1 Analyzing Registry Changes

RegShot is one of the most useful tools that allows "dynamic analysis" by comparing two Registry snapshots with each-other. It's basically old string comparator softwares used to dump strings from services for ScreenSharing, thus it is used on the ScreenSharer's device if they wish to test bypasses and find manual traces in the registry. For example, if we want to see what changes take place in Registry when installing .msi applications or hex editor applications - this way we can also extract relevant registry keys and their subkeys to then check on users and verify that a suspicious behavior occurred. [Seabreg/Regshot: Regshot is a small, free and open-source registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one - done after doing system changes or installing a new software product (github.com)]

Usage:

---

(1) Click the "1st shot" button
It pops up a menu which contains several items:
(A) "Shot", to take a snapshot only, and it will not be kept if you exit Regshot program;
(B) "Shot and save...", to take a snapshot of your registry and save the whole registry to a "hive" file and you can keep it in your hard disk for future use;
(C) "Load...", to load a "hive" file previously saved.
If you want to monitor your file system, just check the "Scan Dir [dir..]" checkbox and input the folder names below it. Note: Regshot has the ability to scan multiple folders, just separate them with ";", Regshot will also scan the subfolders of the current folders you entered.
(2) Run any program that may change your windows registry or the file system
(3) Click the "2nd shot" button
(4) Select your output LOG file type, "text" or "HTML", default is "text"
(5) Enter your comment for this action into the "comment field", e.g.: "Changes made after winzip started". The comment will only be saved into the comparison log files not into "hive" files
(6) Click the "Compare" button

Regshot will do the comparison job now (auto detect which shot is newer), when it is finished,

Regshot will automatically load the comparison log as you defined above, the log files are saved in the directory where "Output path" is defined, default is your Windows Temp Path, it was named as the "comment" you input, if the "comment field" is empty or invalid, the log will be named as "~resxxxx.txt" or "~resxxxx.htm" where "xxxx" is 0000-9999.

(7) Click the "Clear" button

The two snapshots (or separately) previously made will be cleared and you can begin a new job.

Note: "Clear" does not erase the log files!

(8) To quit Regshot, just click the "Quit" button

(9) You can change the language of the Regshot at main window, all words are saved in the file "language.ini". View it for details!

(10) New to 1.7: regshot.dat now changed to regshot.ini, skip dirs and skip registry keys are included. The executable's size is smaller!

(11) New to 1.8:

File shots are now saved in hive file.

"UseLongRegHead" option added in regshot.ini to compatible with undoReg (1.46)

"UseLongRegHead=1" means using "HKEY_LOCAL_MACHINE" instead of "HKLM", default 0

Do not compare shots saved with different "UseLongRegHead" options! (limit gone in 1.9.0)

(12) New to 1.9.0:

x64 and Unicode versions.

You can compare shots saved with different "UseLongRegHead" options.

Loading ANSI hive files in Unicode builds and vice versa is not supported.

What should you be looking for in the final results? As seen in its demo of dynamic malware analysis, it's advised to pay attention to the extraction sizes, check the second snapshot (so the "infected" one) for additional registry values or subkeys generated and keep them for future checks of scripts you may create. For example, here's a script that checks for an input registry key and its subkeys that the user provides (powershell):

```
$keyPath = Read-Host "Enter the registry key path (e.g., HKCU:\Software\Microsoft)"

if (Test-Path $keyPath) {
    Write-Host "Registry key '$keyPath' exists."

    $subKeys = Get-ChildItem -Path $keyPath -Recurse | Where-Object { $_.PsIsContainer }

    if ($subKeys.Count -gt 0) {
        Write-Host "Subkeys under '$keyPath':"
        foreach ($subKey in $subKeys) {
            Write-Host $subKey.Name
        }
    } else {
```

```
    Write-Host "No subkeys found under '$keyPath'."
  }
} else {
  Write-Host "Registry key '$keyPath' does not exist."
}
```

Thus, with this you analyze Registry behavior of cheats and implement them for personal detections. Possibilities are almost limitless.

Utilizing VirusTotal is probably the best registry analysis approach alongside with many others, it links us to other virus analysis results as well.

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	171920	172032	6.5	d77650c42fe7f26e50ffd3a1eb04824b	1013751.12
.rdata	176128	76322	76800	5.83	57d42e7dbe8bbf2a4f453a4a584667c3	2101595.5
.data	253952	13128	3584	1.83	ab6bbf08f3667724642db2d3bfd413c7	589548.12
.pdata	270336	8976	9216	5.36	189d7d6cd859f2885b41151291df8e67	291309.31
_RDATA	282624	348	512	2.82	8073ac06255148b944b04f6a459b3f91	57765
.rsrc	286720	61944	61952	7.33	ea06cf7460cdc7bec119e2d78f419f3d	308723.84
.reloc	352256	1884	2048	5.24	ec92fd910e5f98e8bd04ca09d1b19e26	17510.75

For example here, the higher file entropy the more packed (obfuscated) it is, if one of the section names say ".themida" (in any form like themida or @.themida or @themida) and so on, the fine in practice often appears to be a cheat because cheat devs use Themida to obfuscate their clients.

And in behavior, Registry actions can be also analyzed.

Registry actions

Registry Keys Set

+ HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\GoogleUpdaterInternalService126.0.6441.0\ImagePath
+ HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\GoogleUpdaterInternalService126.0.6441.0\Start
+ HKEY_USERS\S-1-5-21-4270068108-2931534202-3907561125-1001\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.exe\OpenWithProgids\exefile

2.2 Collecting Registry data

Red Lotus has already made a Collector [https://cdn.discordapp.com/attachments/1178373626156826695/1212553955515699210/RLCollector.exe?ex=663b6d1f&is=663a1b9f&hm=efd868b3af9811fb3f0de238013ab881a435f6a2d79bf9c263c9b91a6558de29&] that can be used to Collect numerous Artifact data including of the Registry itself.

For Registry collection it relies on Eric Zimmerman's RECmd [https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RECmd.zip]

The command run to collect pre-defined registry entries (that come with the utility under .reb files) was: RECmd.exe -d C:\\Windows\\system32\\config --bn .\\BatchExamples\Kroll_Batch.reb --nl false --csv .

Essentially we're providing RECmd with a directory to scan which is the config folder with all the registry files in it and parse batch examples, practically extracting all the necessary information we'd need from Registry itself. As the channel linked above already displays its collections, we won't delve deep into this utility. However, let's explain what RECmd is and how it functions.

RECmd.exe is a CLI (command line) based utility that can access and read data from Registry. Think of it as reg.exe but with more possibilities and extraction formats as results can be collected as .csv files that can then be read using Timeline Explorer. The results could also be used by bstrings to immediately hunt for necessary data in the whole registry. This approach becomes handy after analyzing cheat / malware behavior as we already know what to look for exactly.

Let's review its command options

```
    d           Directory to look for hives (recursively). -f or -d is required.
    f           Hive to search. -f or -d is required.
    q           Quiet mode. When true, hide processing details. Default is FALSE
    kn          Display details for key name. Includes subkeys and values
    vn          Value name. Only this value will be dumped
    bn          Use settings from supplied file to find keys/values. See included sample file for examples
    csv         Directory to save CSV formatted results to. Required when -bn is used.
    csvf        File name to save CSV formatted results to. When present, overrides default name
    saveTo      Saves --vn value data in binary form to file. Expects path to a FILE
    json        Export --kn to directory specified by --json. Ignored when --vn is specified
    jsonf       File name to save JSON formatted results to. When present, overrides default name
    details     Show more details when displaying results. Default is FALSE
    Base64      Find Base64 encoded values with size >= Base64 (specified in bytes)
    MinSize     Find values with data size >= MinSize (specified in bytes)
    sa          Search for <string> in keys, values, data, and slack.
    sk          Search for <string> in key names.
    sv          Search for <string> in value names
    sd          Search for <string> in value record's value data
    ss          Search for <string> in value record's value slack
    literal     If true, --sd and --ss search value will not be interpreted as ASCII or Unicode byte strings
    nd          If true, do not show data when using --sd or --ss. Default is FALSE
    regex       If present, treat <string> in --sk, --sv, --sd, and --ss as a regular expression. Default is FALSE
    dt          The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd
HH:mm:ss.fffffff
    nl          When true, ignore transaction log files for dirty hives. Default is FALSE
    recover     If true, recover deleted keys/values. Default is TRUE
    vss         Process all Volume Shadow Copies that exist on drive specified by -f or -d . Default is FALSE
    dedupe      Deduplicate -f or -d & VSCs based on SHA-1. First file found wins. Default is TRUE
    sync        If true, the latest batch files from
https://github.com/EricZimmerman/RECmd/tree/master/BatchExamples are downloaded and local files updated.
Default is FALSE
    debug       Show debug information during processing
    trace       Show trace information during processing
  Example: RECmd.exe --f "C:\Temp\UsrClass 1.dat" --sk URL --recover false --nl
      RECmd.exe --f "D:\temp\UsrClass 1.dat" --StartDate "11/13/2014 15:35:01"
```

```
RECmd.exe --f "D:\temp\UsrClass 1.dat" --RegEx --sv "(App|Display)Name"
```

--d

The --d switch refers to the “directory” of files to collect data from. So if we were to use FTK Imager to extract all config files into a separate directory (this prevents new data from being written into the registry and old data from potentially being lost or overwritten) we would specify that directory to be analyzed. However, often extractions of this kind corrupt the hives. Thus RECmd comes with RLA that can potentially correct these errors. The github page shows exactly how it works.

It is worth noting that the switch requires a target directory and not of RECmd, thus specify directories that have hives in them. Also, this utility can be used to collect data from alternative Windows user accounts that are still stored within their dedicated folders. NTUser and account hives can usually be found to be system hidden in c:\users\%username% as well. In this case we may specify separate usernames that we’re interested in.

--f

This command line orients on specific hives that one would want to scan. For example, if we determine that a certain malware’s behavior is traced in the SOFTWARE hive, there will be no need to potentially affect and also potentially overwrite all the hives using the --d switch. Or, if the --d switch prints errors it may often be because some files can’t be accessed in certain directories. --f can often get along with that.

--bn

As shown in the command above, we specified batch examples after the --bn switch that recursively took every single example command and ran it instantly. This command can be used to initiate collections that users come up with themselves. Hierarchically, RECmd shall begin with the --d switch, then --q switch to make the scan quiet (it speeds up the scan speed) and then any order given in examples.

--sa

Ignoring every other switch, --sa will look for specific strings in the entirety of Registry, this is usually an ASCII scan (excludes Unicodes) but using REGEX we can also print out Unicode characters - or simply do not specify --sa and let the --bn do its work. --regex must be used alongside with --sa, thus with --regex you must have --sa including a regex string.

Combining

In the end, RECmd command must include the following things:

1) --f or --d to scan
2) --bn or --sa with --regex (or without)
3) --q to speed up the scan (or without)
4) --sa is mandatory (it’s string to look for)
5) --csv or --csvf or --json (in the end to output a results file)
6) --dedupe before --csv to deduplicate repeating results and lines
7) --vss to also parse volume shadow copies (recovers data from previous windows snapshots) - snapshots refer to backups windows automatically makes. Not previous windows installations.

2.2.1 Dumping Registry data

If a ScreenSharer uses FTK Imager to extract a registry hive (or any other way of copying config registry hives), specifically the SYSTEM hive, bstrings can read potentially residue data still kept on it before everything gets overwritten. What does it mean? - if users clear their BAM (background activity moderator) logs and the ScreenSharer makes a copy of their SYSTEM hive (on their own system, obviously) they can then use the following command:

```
bstrings.exe -f "C:\Windows\system32\config\SYSTEM" --ls harddiskvolume
```

To read that deleted data before it gets overwritten! (as mentioned above, making a copy or extracting these hives prevents them from being updated / overwritten)

Alternatively, using the --regex switch can print only the file paths! **^([a-zA-Z]:\\.+)\\?$**

3. Registry Explorer

Registry Explorer comes with a Manual that explains everything necessary. In fact, the DFIR communities have created python scripts that allow the usage of Reg Explorer into utilities like Autopsy. source: Autopsy-Registry-Explorer/regparser.py at main · 0xHasanM/Autopsy-Registry-Explorer (github.com)

However, for ScreenSharing purposes utilities like Autopsy, Volatility and MemProcFS will not be covered in Red Lotus handbooks (mainly because they already have their own documentations and there's nothing new Lotus can offer regarding them); they may be released if Lotus+ members request to with specific approaches on how to detect certain anomalies.

[Eric Zimmerman's tools] **Registry Explorer -** is a GUI based RECmd that revolutionized working with Windows Registry. Before proceeding, let's follow the license requirement.

MIT License

Copyright (c) 2021 Eric Zimmerman

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

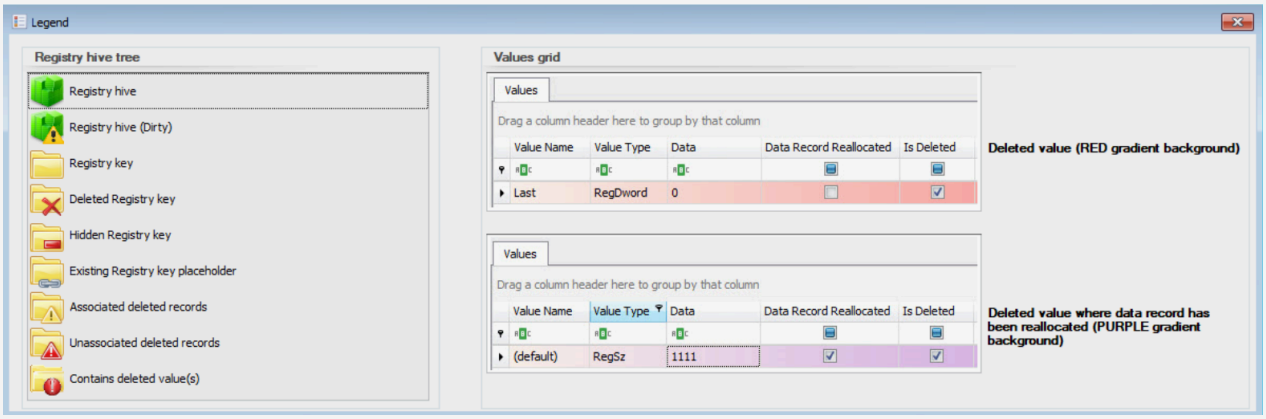
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

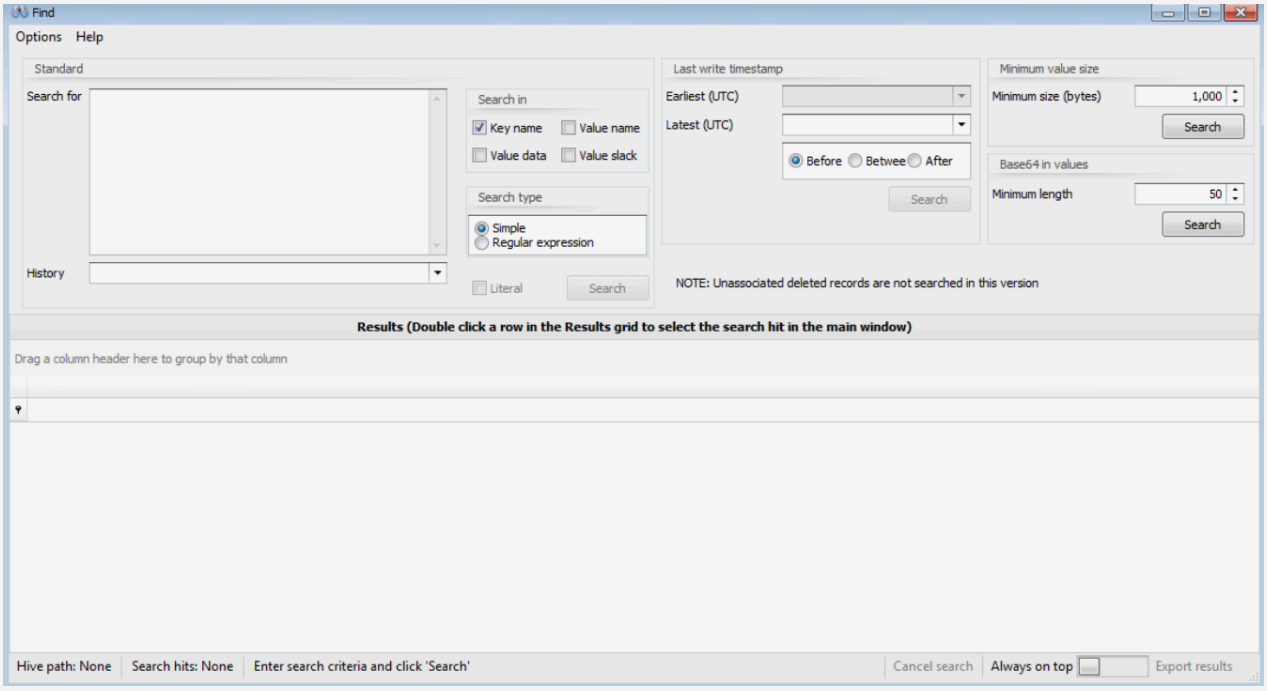
With that out of the way, let's review the most useful features (for ScreenSharing) with this utility.

Presenting icon definitions:



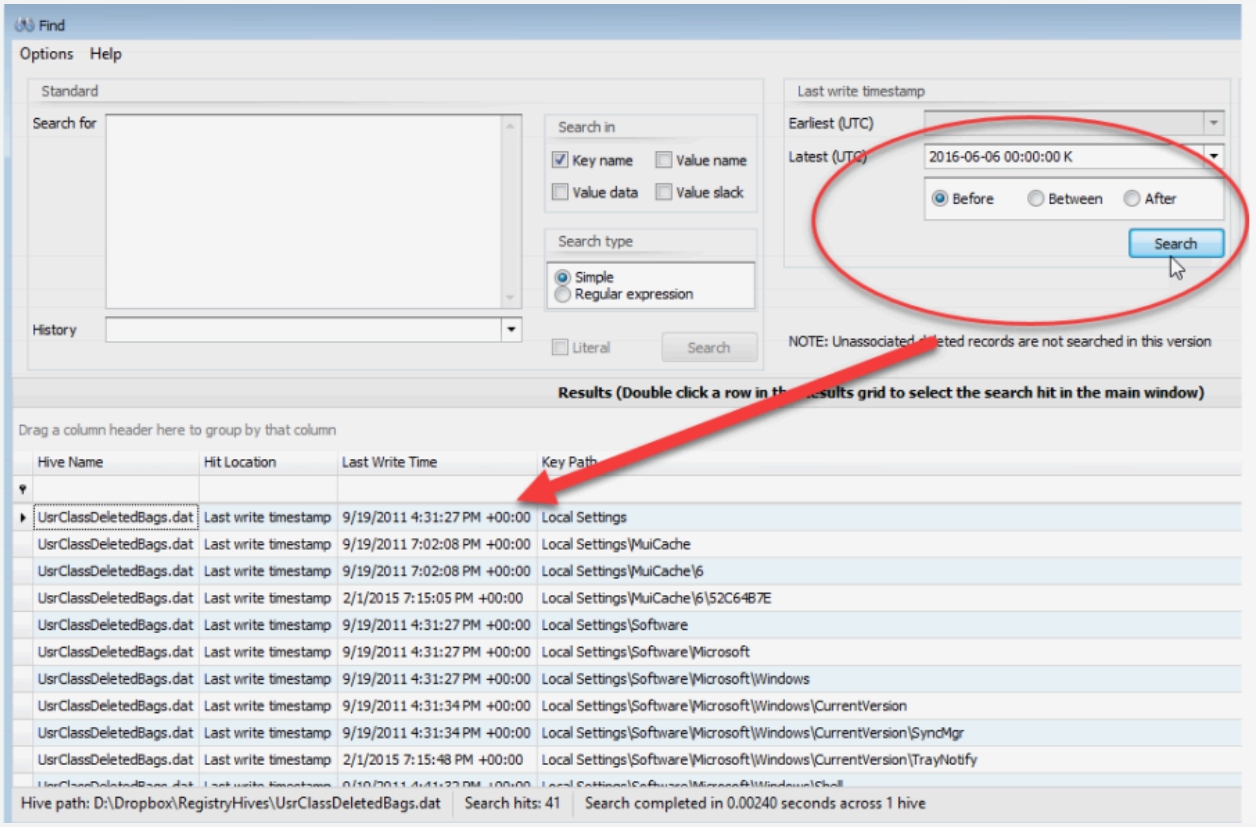
Paying attention to the last Icon in the Hive Tree section, we see a circled warning sign that we will most often see on BAM and MuiCache or ComDLG hives. What do we do and what does it mean? - it means that the given hive we are looking at includes deleted values, these values will not be displayed or highlighted in practice with BAM, however **if you see a circled warning icon on BAM or any hive - feel free to ban for it as they display only during current system boot.**

Another useful feature of Reg Explorer is the speed and ease of searching. By pressing CTRL+F we will be met with the following window:



Here, in “search for” we can type any word (string) to look for. For example, USB to look for every value containing the word “USB” or .exe or .dll or C:\ D:\ E:\ and other drive letters and much much more. Make sure to toggle value name, data and value slack in “Search in”

As explained in the original .pdf handbook, there are a lot of practical details that can be utilized while in this menu, such as searching for specific dates in UTC



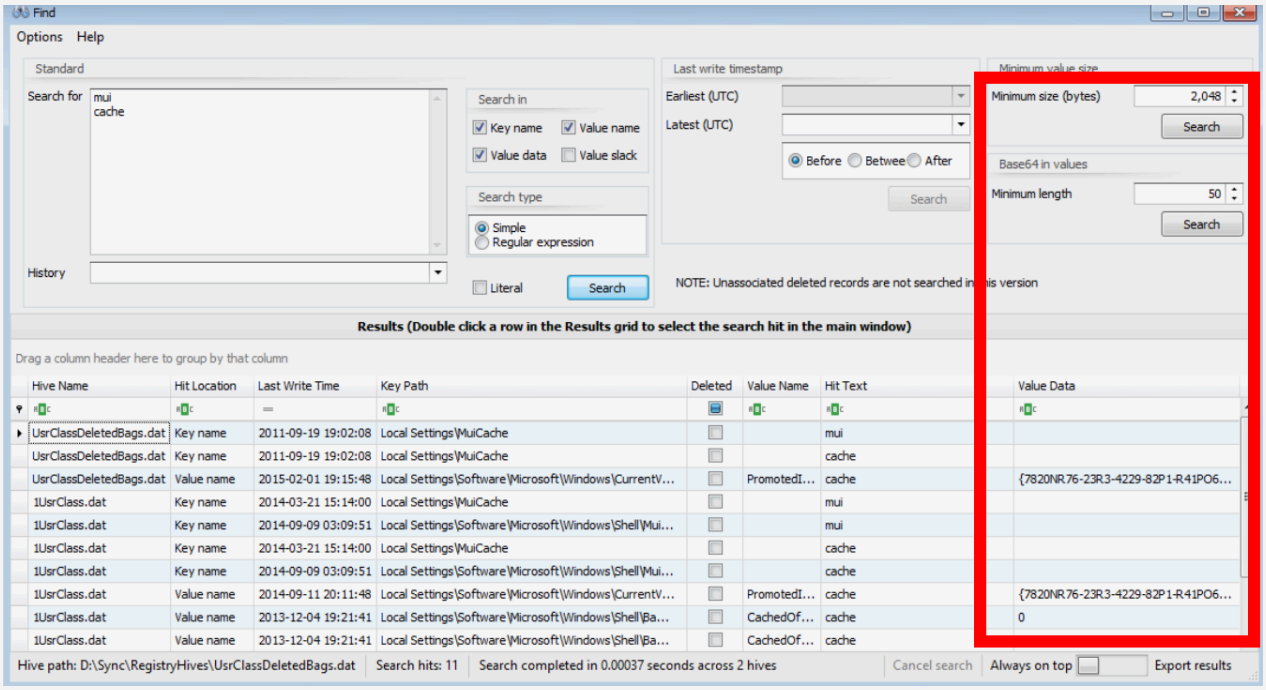
To check user's current date in UTC refer to any website(s), for example [Current UTC — Coordinated Universal Time (timeanddate.com)]();

More logical suggestions will be discussed below using source, explanations and images taken from the official handbook (comes with the download of registry explorer)

3.1 Analytic approach to value sizes

If we combine the knowledge this Chapter talked about previously regarding hunting malware behavior via Registry, human analysis can be utilized to first hunt the behavior of malwares, cheats - be it harmless autoclickers or .jar and .dll files or utilities that can be used to bypass with, we can mark their value sizes and conclude an min-max range that will ease their search. For example, let's say Karma Client's value size logged as 2,150 during first registry comparing tests and the results varied between 2,150~3,130 in a span of 10 different segments, **we can provide reg explorer with a min-max size range and analyze the results (or even export them to be used with other methods)**

More extended information about value sizes is provided in the .pdf handbook on page 48~49~50;

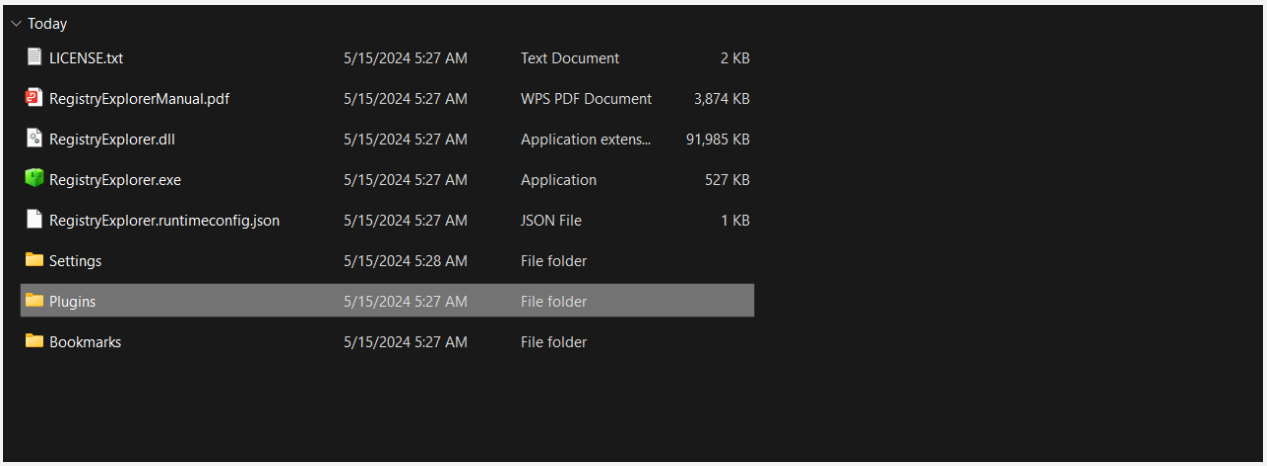


Make sure to read the official .pdf handbook as it includes more than enough information. Besides - for over 3 years the SS Communities had no idea CTRL+F was an option until I decided to read the handbook and share that piece of information, respect the authors shall we.

3.1.1 Adding our own plugins (for programmers)

https://github.com/EricZimmerman/RegistryPlugins

Registry Explorer relies on plugins that it comes with. It can be found in one of the folders when unzipping Registry Explorer.



The repository I linked above contains all source codes for its plugins, they'll need to be compiled into .dlls if we want to add any changes to it. Bookmarks and settings can be edited without compiling as they're just "targets" reg explorer will load upon launch.

3.1.2 Troubleshooting

Registry Explorer should be **run as Administrator** to attain access to configuration files. In settings, make sure to enable "recover deleted values" (or have them disabled if you want to) before loading hives, as said in the handbook of this utility - they take effect **while** loading hives, thus they don't update live and you'll have to reload all hives when making changes to them.

Make sure to run the utility from its directory, launching it via search everything might return access errors.

Information in Registry Explorer is displayed in UTC time-zone by default. For most values like ComdLG, MuiCache, ShimCache and so on - date of modification refers to the last entry in the hives. Thus if you find the date modified by ComDLG on .exes to be days before SS, it means that the last entry recorded of .exe files in ComDLG was made so long ago, thus it is not worth wasting time on going through them.

Registry Explorer's dates serve as evidence of execution. Latest entries usually start from the bottom to top.

4. Conclusion (Registry)

There are many ways Registry approaches - especially Registry Explorer can be combined with other utilities. For example, as Tome II Book I already explained - Registry Explorer can be used to detect unicode characters and scheduled tasks. With this in mind, every ScreenSharer can load Registry Explorer right as they begin ScreenSharing **to avoid overwriting registry data and losing evidence** for bans. As I already mentioned, registry explorer takes static snapshots of the registry and doesn't update live. This can come handy immediately just like Scheduler service and Bstrings on BAM analysis does.

Any information obtained from other evidence of execution or while parsing services with system informer can be used with Registry Explorer. For example, finding **volume** or **localhost** access during ScreenShare can be an indicator (if volume\ is in WinPrefetch for example or localhost already on the device with wsl) of suspicious behavior - data of which does not get lost in $MFT or Registry hives.

Access on files launched with scheduler still gets logged in muicache ad shimcache - thus with CTRL+F option in Reg Explorer we can instantly find all traces of specific files (for example, certain cheats found in DPS) all over Registry for more accuracy.

In the end, Registry Explorer and registry approaches can play a crucial and undeniably important role when detecting malicious behavior on the system or bypass attempts. If learnt

well, ScreenSharers will develop a general logic when conducting ScreenShare and can use Registry as their prime or last resort.

CHAPTER II. USN Journal

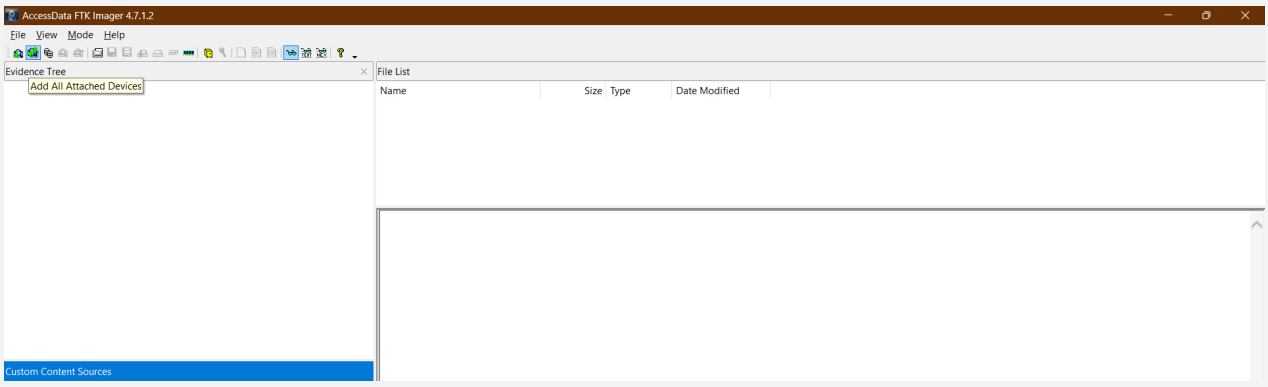
1. Introduction to $J

$J is a USN Journal metafile on NTFS drives (unlike FAT32) that keep all behavioral patterns of applications or initiated actions (by user or the system) on the device. For instance, shift deleting files, although don't update $recycle.bin date of modification will still be logged in USN Journal as deleted|close reason. $J is not only useful in ScreenShares but also the **only** resource ScreenSharers have when dealing with file modifications, hex changes, attribute changes and so on. $J includes alternate streams (metafiles), $UsnJrnl and $MAX in which $UsnJrnl is a "Changes Journal" that holds all the information about system changes done by the user or the system itself, whereas the $MAX file holds information regards changes of $UsnJrnl. By this, even if users manage to hook $UsnJrnl with via Kernel level access, $MAX file will still hold that information, thus Date Modified on $MAX equals date modified of $UsnJrnl that results in enough proof to ban for Bypass Attempt.

In this Chapter, we will cover the usage of $MFT with $J, discuss what can be done to analyze USN Journal and why it is the only resource advantageous over numerous bypasses.

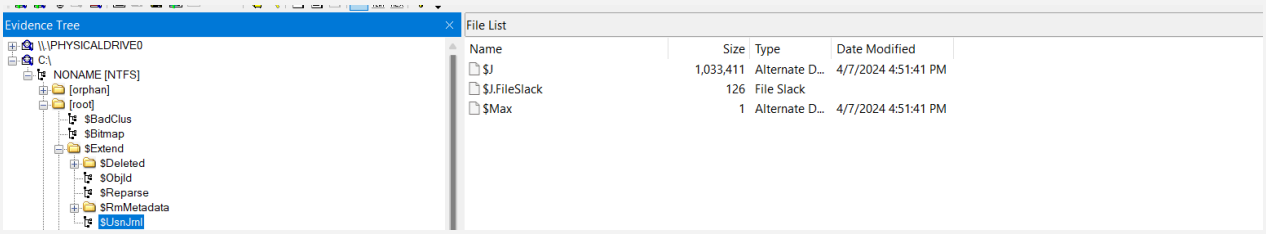
1.1.1 FTK Imager and $UsnJrnl

One way to view modification dates of $J and $UsnJrnl is by analyzing $MFT. However, since FTK Imager already does that and builds a file tree with directories with more potential uses, we will cover an approach to detect Journal modification (journal clearance or hooking) and $MAX file modification.



After attaching all devices to FTK Imager, navigating through the $Extend directory will lead us to the $J file holding $UsnJrnl and $MAX files.

It is important to note that by right clicking these files, ScreenSharers can extract all data of USN Journal to avoid further overwriting it or making confusing changes to them - especially when MFTECmd is capable of parsing the Journal further extracting static data more useful for gathering evidences.



In the following screenshot, Date Modified corresponds to when changes were made to the USN Journal (or the Max file) whereas the .fileslack **means that the $J file was overwritten - deleted while applying changes to it.** Hence why there are no dates of modifications on the fileslack.

Modification dates are usually sorted in UTC. Sometimes they glitch out and display absurd dates - in which case we can export them and parse them with MFTECmd (or use MFTECmd to parse the $MFT then check if that helped dates of modification) after which we'll analyze the latest and oldest entry. This way (unlike with JournalTrace) we can accurately determine when the Journal began logging changes on the system, whereas JournalTrace and many other Journal viewer utilities often display only recent day / month modifications, therefore losing potentially crucial data.

$J.slack file can sometimes hold the last bit of data $UsnJrnl logged before erasure, or if hooked - it'll hold the data that was "hidden" or as they call it "deleted" from Journal. But remember that deleting, overwriting or manipulating the given metafile ($J) is **not possible.** Attempting to do so usually bricks or crashes devices.

1.1.2 Uses of $UsnJrnl

Many ScreenSharers do not know it (and many have never thought of it either) but USN Journal and $MFT can be used to detect Time Stomping and Hex modifications on files. Timestomping is a technique that changes attributes (dates of modification) of files, be it their prefetch entries. This behavior logs as **Basic Info Change | Close** in USN Journal, thus any basic info change on a .pf file or .exe or any extension of a cheat shall be bannable. Hex modification logs as **Data Overwrite** or **Data Extend | Data Truncation** sometimes **Data Extend | Close** since an entire functionality of the file is modified. Let's delve deep into these anomalies and discuss with references.

Referencing the Habook forensic team:

"Since "$J" has records of all changes on files and folders in a volume including deleted files, this opens the doors for digital forensics investigators and threat hunters to empower their analysis with information such as the following:

Detection of malicious tools and bad files that were present at some point in time within the file system, which provides insight into suspicious/malicious user activity.

Detection of "Timestomping" activity, a technique used by attackers which is the alteration of timestamps of files to confuse investigators during their analysis.

Extending "Prefetch" artifacts value, which each contain the dates for the last 8 times an executable was run, which is a limitation of the artifact. This limitation can be overcome to gain the dates for more executions of an executable (Subject to the limitations of USN Journal below)."

Extending Prefetch artifacts value means that we can analyze data extending entries of Journal on prefetch traces like Conhost / Consent to determine if they were further modified - or when they were modified besides their last 8 entries. Let's review some examples, though first:

We gather Registry hives and load up Registry Explorer at the start of the ScreenShare. We have also gathered $MFT (or installed Search Everything) with SRUM and Event Logs and kept them for references for the future; just in case - we also used PECmd to export all necessary data that we can quickly skim through and used RL Collector for every other artifact that can hold dates of execution to not waste time on their collections later:

Scenario A:

In WinPrefetchView, we see two suspicious sequences: Regsvr, RunDLL, Conhost and Consent have the last 8 entries (9 including latest modification date) only a couple minutes apart. This is suspicious and leads to assumptions of **overwriting** previous data. Using $UsnJrnl we will be able to determine all sequence entries and gather dates of modification that took place before the last 8~9 entries that are clearly meant to confuse the ScreenSharer or overwrite data in existing entries **because that way** it becomes much harder to prove injections of .dll files or renamed extensions right away. On Conhost's example, we are met with the following:

Update Timestamp	Parent Path	Name	Update Reasons
=		conhost	DataExtend|DataTruncation|Close
2022-09-26 08:22:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:04:04	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:04:07	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 07:07:57	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 10:22:16	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-27 10:28:02	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:01:53	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:06:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:10:06	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:55:45	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 10:59:12	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-28 11:01:37	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:36:03	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:47:18	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-09-29 07:49:30	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:23:32	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:25:11	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 09:45:35	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close
2022-10-23 11:47:40	.\Windows\Prefetch	CONHOST.EXE-1F3E9D7E.pf	DataExtend|DataTruncation|Close

Thus, by selecting beyond 8 dates of modifications on Conhost - we can copy the Update Timestamp of that and filter using that timestamp **in the** Journal itself (to see what files were modified as they will be logged) or use that specific timestamp in other evidences of execution to see which file initiated and logged in Conhost.

Scenario B:

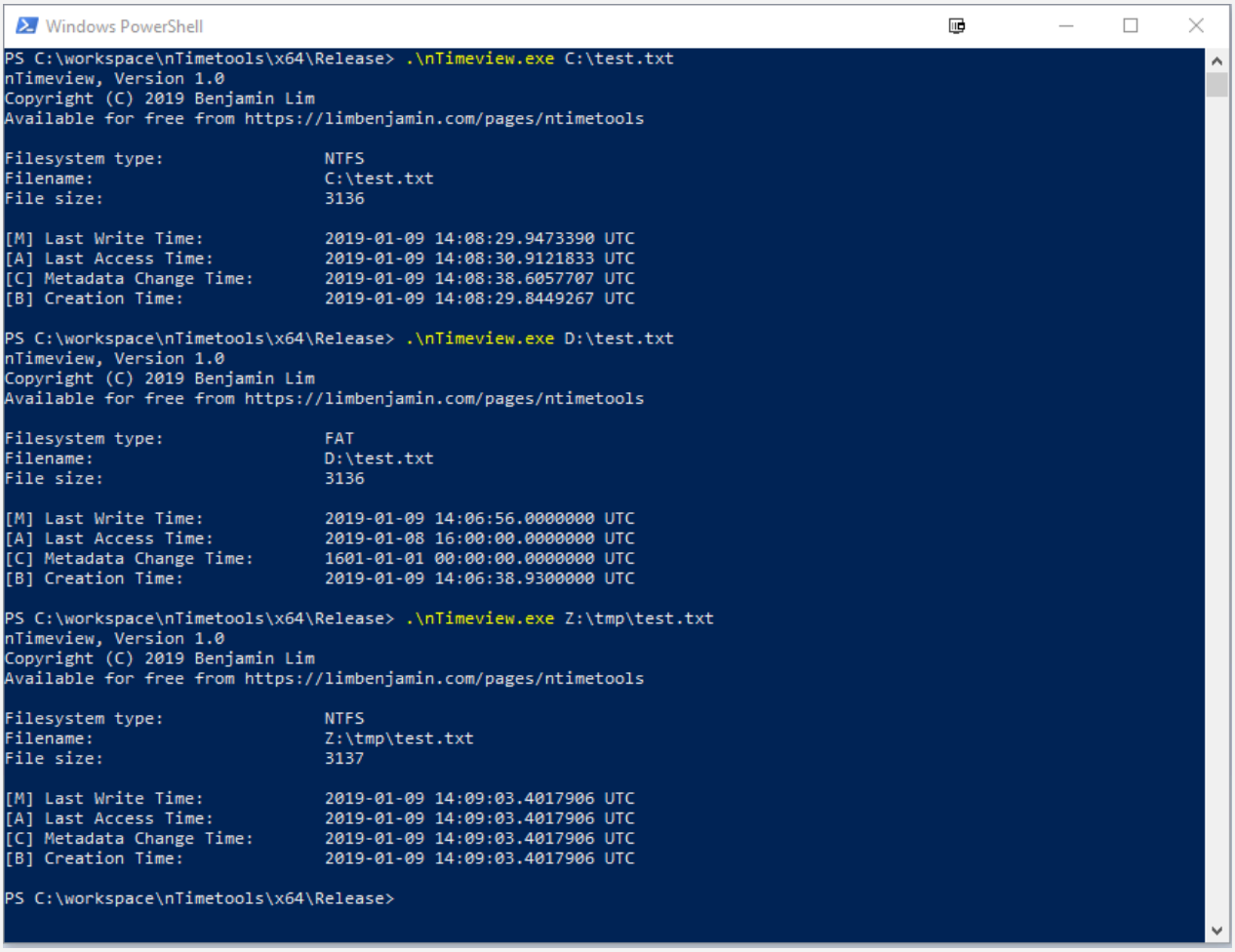
In WinPrefetchview we see a .pf trace of slinky that was executed months ago. However, the file itself was logged both in csrss and DPS. Using USN Journal with Basic Info Change | Close - we will see that .pf trace modification and conclude that the file had Read-Only attribute applied to avoid further update sequences per launch. Enough to ban for.

Scenario C:

We generally want to detect files that have their attributes modified. For example, calc process hollowing tools. We will utilize Journal the same way as in Example B and read the results.

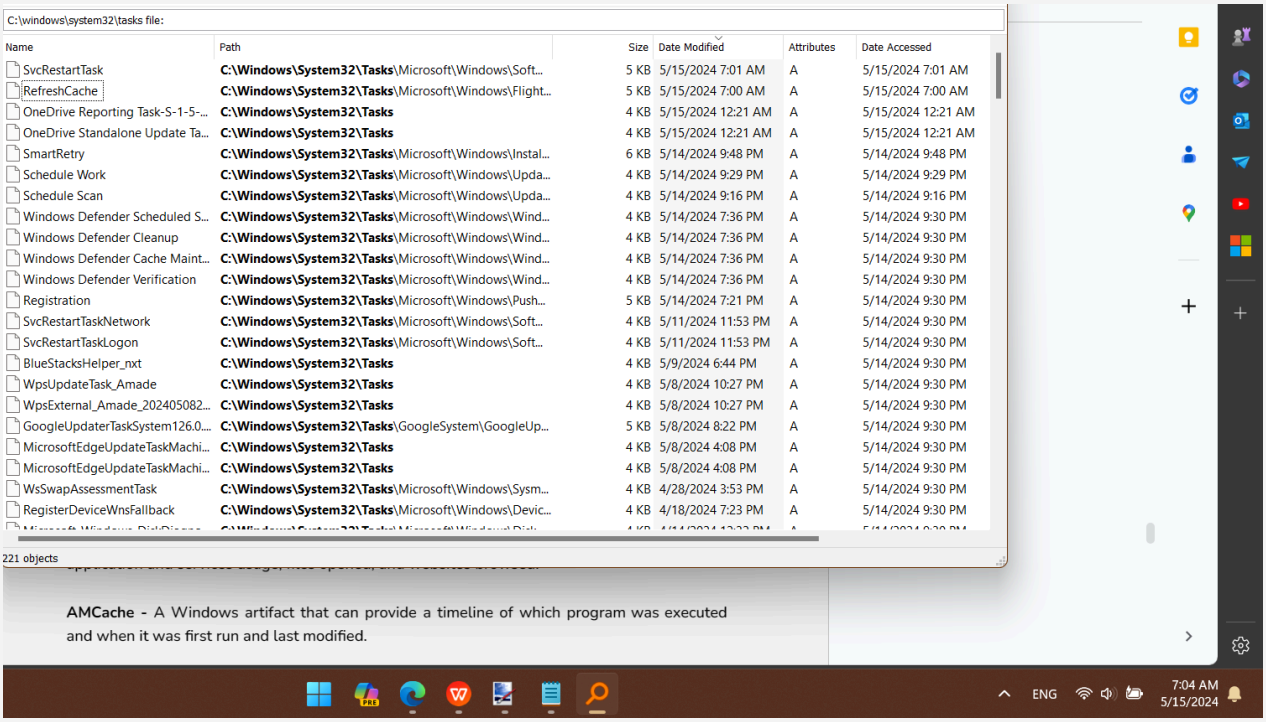
Update Timestamp	Parent Path	Name	Update Reasons
=	.\users\user\desktop\article	calcx.exe	
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	FileCreate
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate|BasicInfoChange
2022-10-24 16:33:25	.\Users\User\Desktop\article	calcx.exe	DataExtend|FileCreate|BasicInfoChange|Close
2022-10-24 16:34:55	.\Users\User\Desktop\article	calcx.exe	BasicInfoChange
2022-10-24 16:34:55	.\Users\User\Desktop\article	calcx.exe	BasicInfoChange|Close

Changing timestamps of files can be done using any date modification application such as nTimeStamp.exe for example. Here's how it'd look:



Scenario D:

We open Search Everything and analyze scheduled tasks. We are met with the following image:



Certain tasks have dates modified 3 minutes before ScreenShare. Such behavior is usually normal with SvcRestartTask and RefreshCache though we will use them as examples for now. How do we tell if the file was modified before ScreenShare?

Solution (A)

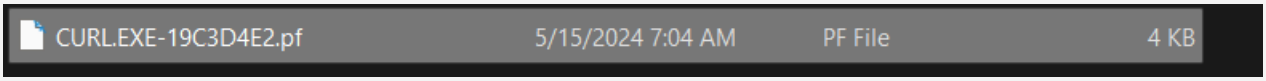
If the Bypasser used right click > open as, it logs in WinPrefetchView under **Open As**; any .exe or task logged there is enough to ban for.

Solution (B)

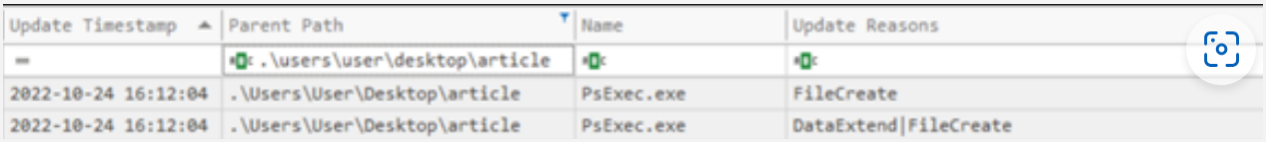
Data Extend | Close or Data Extend | Data Truncation | Close will be enough to ban for as these sequences do not update on scheduled task files by default.

Scenario E:

In Prefetch we see a Curl trace but we are met with the same problem as in Example A. Is it over? can we no longer find any evidence? are we doomed? do we ban right away? the answer is no to each of these questions. Here's why:



By analyzing Journal and finding its all previous entries (if nothing found in Curl using WinPrefetchView) we can copy the date and **use that date on File Create | Close entries!**



Update Timestamp	Parent Path	Name	Update Reasons
	.\users\user\desktop\article		
2022-10-24 16:12:04	.\Users\User\Desktop\article	PsExec.exe	FileCreate
2022-10-24 16:12:04	.\Users\User\Desktop\article	PsExec.exe	DataExtend|FileCreate

This means a file was created. In this case - downloaded.

Scenario F:

There is a bypass spread in the community that relies on Hex modifying cheat files; this bypassed every SS Tool and every ScreenSharer on the platform. Nobody can prove anything

because it's impossible to recover the old file so they issue false bans on people who they assume use the bypass. What to do?

Refer to Journal. Data Extend | Data Truncation | Close or Data Extend | Close **especially** Stream Change | Close or Data Overwrite | Close on .exe files (or any file that displays hashmismatch and invalid signature in signature checks) **is enough to ban for as that's what hex overwriting displays.**

WhiteLotus.exe	15/05/2024 07:13:02	Data extend | Data truncation
WhiteLotus.exe	15/05/2024 07:13:02	Data extend | Data truncation | Close

1.2.1 Resources & Reasons Table

[Introduction to MFTECmd - NTFS MFT and Journal Forensics]

Value	Meaning
USN_REASON_BASIC_INFO_CHANGE 0x00008000	A user has either changed one or more file or directory attributes (for example, the read-only, hidden, system, archive, or sparse attribute), or one or more time stamps.
USN_REASON_CLOSE 0x80000000	The file or directory is closed.
USN_REASON_COMPRESSION_CHANGE 0x00020000	The compression state of the file or directory is changed from or to compressed.
USN_REASON_DATA_EXTEND 0x00000002	The file or directory is extended (added to).

USN_REASON_DATA_OVERWRITE 0x00000001	**The data in the file or directory is overwritten.**
USN_REASON_DATA_TRUNCATION 0x00000004	**The file or directory is truncated.**
USN_REASON_EA_CHANGE 0x00000400	**The user made a change to the extended attributes of a file or directory. These NTFS file system attributes are not accessible to Windows-based applications.**
USN_REASON_ENCRYPTION_CHANGE 0x00040000	**The file or directory is encrypted or decrypted.**
USN_REASON_FILE_CREATE 0x00000100	**The file or directory is created for the first time.**
USN_REASON_FILE_DELETE 0x00000200	**The file or directory is deleted.**
USN_REASON_HARD_LINK_CHANGE 0x00010000	**An NTFS file system hard link is added to or removed from the file or directory.**

	An NTFS file system hard link, similar to a POSIX hard link, is one of several directory entries that see the same file or directory.
USN_REASON_INDEXABLE_CHANGE **0x00004000**	**A user changes the FILE_ATTRIBUTE_NOT_CONTENT_INDEXED attribute.** **That is, the user changes the file or directory from one where content can be indexed to one where content cannot be indexed, or vice versa. Content indexing permits rapid searching of data by building a database of selected content.**
USN_REASON_INTEGRITY_CHANGE **0x00800000**	**A user changed the state of the FILE_ATTRIBUTE_INTEGRITY_STREAM attribute for the given stream.** **On the ReFS file system, integrity streams maintain a checksum of all data for that stream, so that the contents of the file can be validated during read or write operations.**
USN_REASON_NAMED_DATA_EXTEND **0x00000020**	**The one or more named data streams for a file are extended (added to).**

USN_REASON_NAMED_DATA_OVERWRITE 0x00000010	**The data in one or more named data streams for a file is overwritten.**
USN_REASON_NAMED_DATA_TRUNCATION 0x00000040	**The one or more named data streams for a file is truncated.**
USN_REASON_OBJECT_ID_CHANGE 0x00080000	**The object identifier of a file or directory is changed.**
USN_REASON_RENAME_NEW_NAME 0x00002000	**A file or directory is renamed, and the file name in the USN_RECORD_V2 structure is the new name.**
USN_REASON_RENAME_OLD_NAME 0x00001000	**The file or directory is renamed, and the file name in the USN_RECORD_V2 structure is the previous name.**
USN_REASON_REPARSE_POINT_CHANGE 0x00100000	**The reparse point that is contained in a file or directory is changed, or a reparse point is added to or deleted from a file or directory.**
USN_REASON_SECURITY_CHANGE 0x00000800	**A change is made in the access rights to a file or directory.**

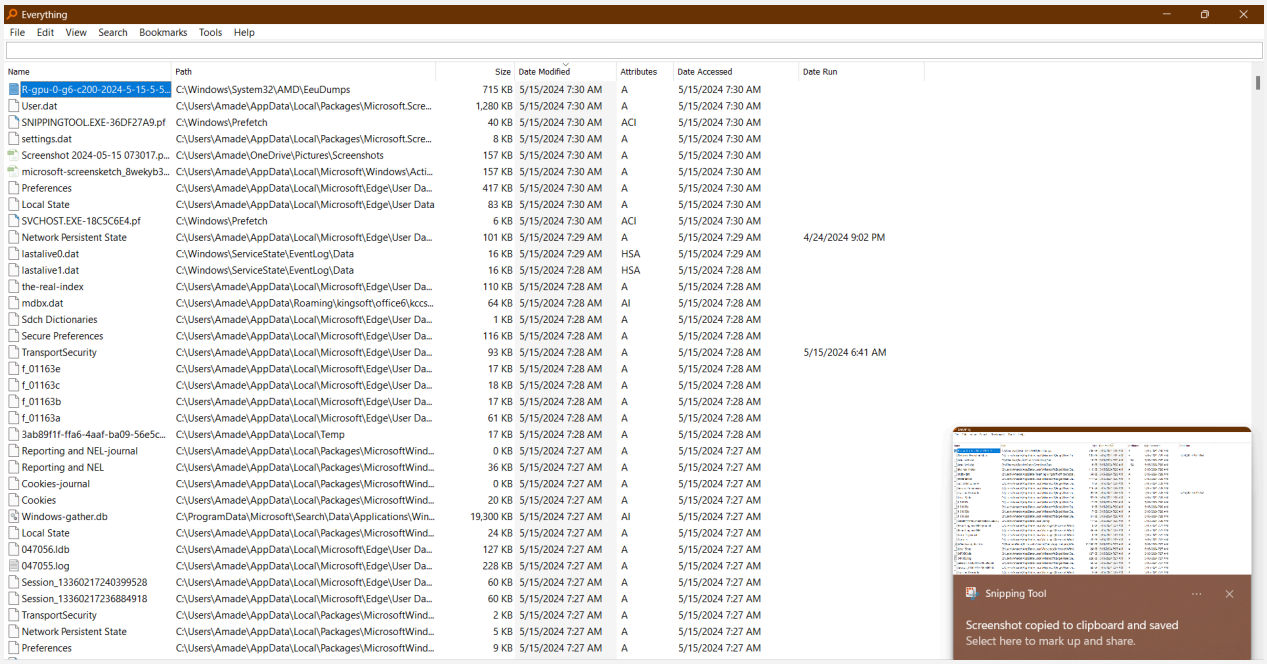
USN_REASON_STREAM_CHANGE **0x00200000**	**A named stream is added to or removed from a file, or a named stream is renamed.**
USN_REASON_TRANSACTED_CHANGE **0x00400000**	**The given stream is modified through a TxF transaction.**

Value	Meaning
USN_SOURCE_AUXILIARY_DATA **0x00000002**	**The operation adds a private data stream to a file or directory.** **An example might be a virus detector adding checksum information. As the virus detector modifies the item, the system generates USN records. USN_SOURCE_AUXILIARY_DATA indicates that the modifications did not change the application data.**
USN_SOURCE_DATA_MANAGEMENT **0x00000001**	**The operation provides information about a change to the file or directory made by the operating system.**

	A typical use is when the Remote Storage system moves data from external to local storage. Remote Storage is the hierarchical storage management software. Such a move usually at a minimum adds the USN_REASON_DATA_OVERWRITE flag to a USN record. However, the data has not changed from the user's point of view. By noting USN_SOURCE_DATA_MANAGEMENT in the SourceInfo member, you can determine that although a write operation is performed on the item, data has not changed.
USN_SOURCE_REPLICATION_MANAGEMENT 0x00000004	**The operation is modifying a file to match the contents of the same file which exists in another member of the replica set.**
USN_SOURCE_CLIENT_REPLICATION_MANAGEMENT 0x00000008	**The operation is modifying a file on client systems to match the contents of the same file that exists in the cloud.**

1.2.2 Monitoring Journal

If we want to detect a bypass or what it does to the system, we can obviously use snapshots of USN Journal (clear it, do the bypass, extract and save) over and over again. There are also utilities that monitor changes on the system and display them live. One of such utilities is.. Search Everything. Search Everything tracks the NTFS and displays live changes or updates. Thus, if we want to create a detection to the bypass - we can start by maximizing search everything in the background and **closely monitoring changes** while doing bypasses. For example, as shown in the screenshot by just screenshotting multiple changes were applied to the system - a prefetch was made and caches dates were modified, thus data was updated.



For more “closed” checks, for example to determine if replacing a file logs any special changes in USN Journal, using tools like MFTECmd or JournalTrace is sufficient. With MFTECmd, extracting USN Journal is pretty simple. Let’s review all of its switches first.

```
Administrator: C:\WINDOWS\system32\cmd.exe
MFTECmd version 0.5.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

        f               File to process ($MFT | $J | $LogFile | $Boot | $SDS). Required

        json            Directory to save JSON formatted results to. This or --csv required unless --de or --body is spe
cified
        jsonf           File name to save JSON formatted results to. When present, overrides default name
        csv             Directory to save CSV formatted results to. This or --json required unless --de or --body is spe
cified
        csvf            File name to save CSV formatted results to. When present, overrides default name

        body            Directory to save bodyfile formatted results to. --bdl is also required when using this option
        bodyf           File name to save body formatted results to. When present, overrides default name
        bdl             Drive letter (C, D, etc.) to use with bodyfile. Only the drive letter itself should be provided
        blf             When true, use LF vs CRLF for newlines. Default is FALSE

        dd              Directory to save exported FILE record. --do is also required when using this option
        do              Offset of the FILE record to dump as decimal or hex. Ex: 5120 or 0x1400 Use --de or --vl 1 to se
e offsets

        de              Dump full details for entry/sequence #. Format is 'Entry' or 'Entry-Seq' as decimal or hex. Exam
ple: 5, 624-5 or 0x270-0x5.
        fls             When true, displays contents of directory specified by --de. Ignored when --de points to a file.

        ds              Dump full details for Security Id as decimal or hex. Example: 624 or 0x270

        dt              The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd HH:mm:ss.
fffffff
        sn              Include DOS file name types. Default is FALSE
        at              When true, include all timestamps from 0x30 attribute vs only when they differ from 0x10. Defaul
t is FALSE

        vss             Process all Volume Shadow Copies that exist on drive specified by -f . Default is FALSE
        dedupe          Deduplicate -f & VSCs based on SHA-1. First file found wins. Default is FALSE

        debug           Show debug information during processing
        trace           Show trace information during processing


Examples: MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out" --csvf MyOutputFile.csv
          MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --json "c:\temp\jsonout"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --body "c:\temp\bout" --bdl c
          MFTECmd.exe -f "C:\Temp\SomeMFT" --de 5-5

          Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes

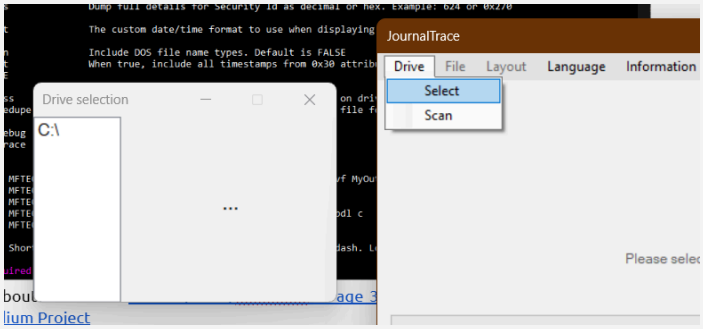
-f is required. Exiting
```

More about MFTECmd: [MFT Explorer/MFTECmd - Page 3 of 4 - AboutDFIR - The Definitive Compendium Project]

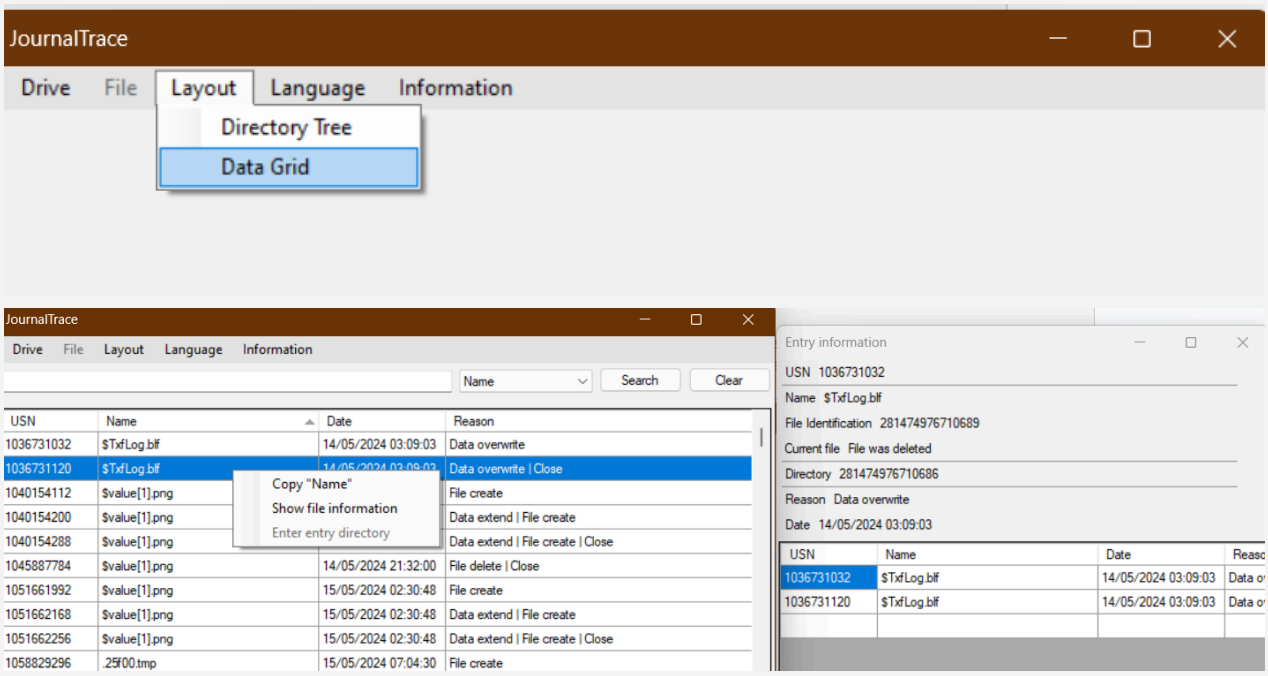
Thus, if we want to extract Journal only, we'll have to use the following command and read results in TimeLine Explorer (as shown in the 2. Video demonstration):

MFTECmd.exe -f C:\$Extend\$UsnJrnl:$J --csv .

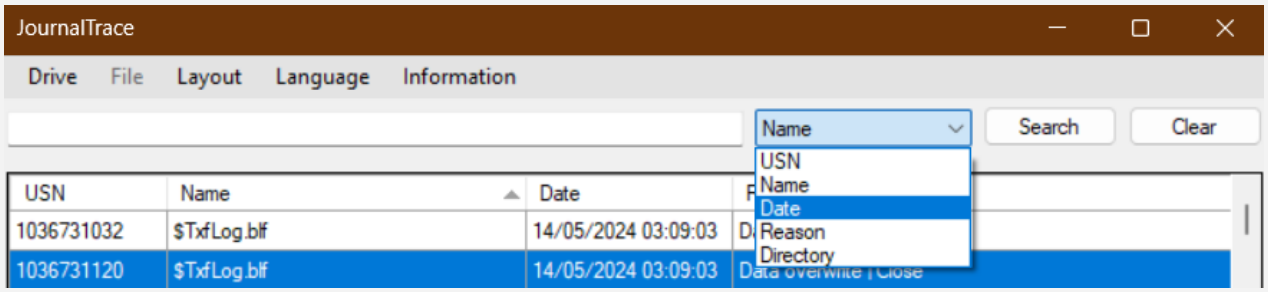
Though here's JournalTrace: [Releases · ponei/JournalTrace (github.com)]



Select a drive and scan it (you'll have to go back and forth to parse) and view the results via "Layout" > "Data Grid" (directory Tree returns a list of directories like via /dir) if you want to make changes to directories or get used to dir sequences.



For extended information you can right click on the file name and "Show file information", by right clicking you can also copy the dates and follow examples listed in above situational scenarios.



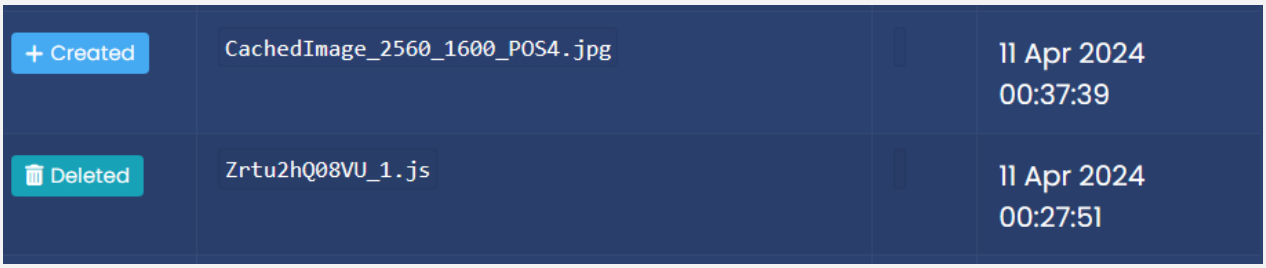
Name is for file names, Reason is for update sequences, directory is for directories and USN is for USN entry numbers (though that usually crashes JournalTrace)

You can learn more about USN Journal here:

[USN_JOURNAL_DATA_V2 - Win32 apps | Microsoft Learn]
[USN_RECORD_V2 - Win32 apps | Microsoft Learn]
[USN_RECORD_V3 - Win32 apps | Microsoft Learn]
[USN_RECORD_V4 - Win32 apps | Microsoft Learn]

2.1.1 Human-System Operations

Although it is clear how to use Journal, how to read it, what its sequence numbers and reasons represent, still how do we tell if an action was initiated by the user and when was it initiated by the system itself? We will be using Echo scan results for these examples.



Point A:

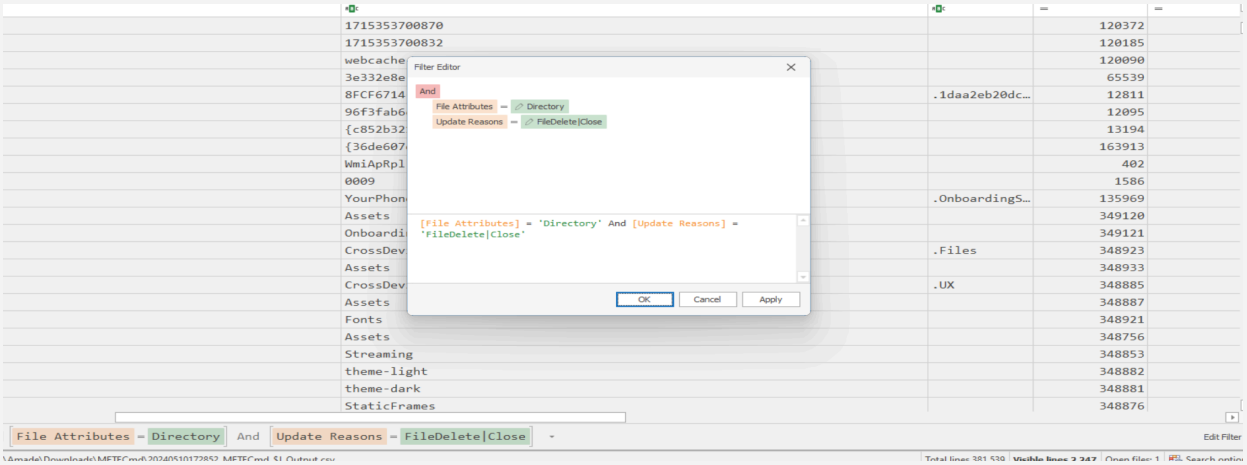
Usn Journal entries that do not include file paths are system initiated actions. For example, being me with .js or image.png[1] results in Echo (or JournalTrace) will always have either no file paths or sequence numbers for file paths (Directory);

	Directory
e: new name | Basic info change | ...	C:\Users\Amade\AppData\Local\Microsoft\Edge\User Data\Default\Network
	C:\Users\Amade\AppData\Local\Microsoft\Edge\User Data\Default\Network
	C:\Users\Amade
end	C:\Users\Amade
	283445301547631552
	283445301547631552

This can also mean that the root directory of a file was deleted. Although, practice has shown that it is very complicated to prove user initiated actions or bannable offenses without relying on other tools and evidence of execution when dealing with such scenarios. Let's review hypothetical cases.

Scenario A - Folder deletion:

When ScreenSharing, we notice a folder in the shell:recent being accessed that is no longer on the system. In Journal, we see a bunch of **non-executable** files deleted shortly before ScreenShare. Using a parsed $J .csv with MFTECmd will help us determine if the folder was deleted:



For this, we can use the following filter: **[File Attributes] = 'Directory' And [Update Reasons] = 'FileDelete|Close'**

	Update Reasons	File Attributes
	= FileDelete|Close	= Directory
8	FileDelete|Close	Directory
8	FileDelete|Close	Directory

And after that we can use System Informer (csrss, dps) or any other evidence of execution (especially Registry) to determine if the file was launched.

How do we determine if a deleted folder / file was a system operation? it will be useful to know what are considered “bad filename characters”;

Illegal Filename Characters

Do not use any of these common illegal characters or symbols in your filenames or folders:

- # pound
- % percent
- & ampersand
- { left curly bracket
- } right curly bracket
- \ back slash
- < left angle bracket
- > right angle bracket
- * asterisk
- ? question mark
- / forward slash
- blank spaces
- $ dollar sign
- ! exclamation point
- ' single quotes
- " double quotes
- : colon
- @ at sign
- + plus sign
- ` backtick
- | pipe
- = equal sign
- emojis
- alt codes

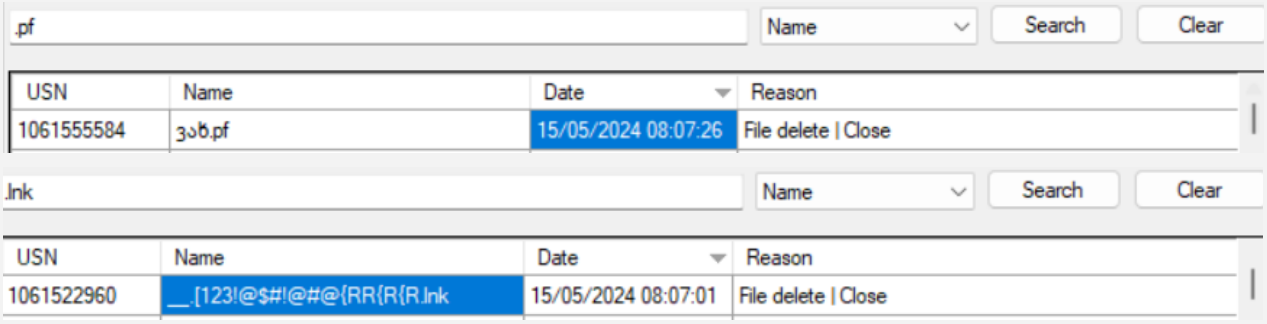
Most of these marks are **never seen in human made / named files and folders** unless of course they are aware that the ScreenSharer will rely on this logic. In such cases, to differentiate a human-system operation, we can rely on the **consistency of operations and file modifications in Journal** to conclude what really could’ve happened. For this, a combined use of Registry (or other evidence of execution) or JumpLists and Journal can be useful.

For instance, if we see a folder deletion in Journal **but the folder still exists** and while accessing the folder **we are met with warning popups** it is more likely a system operation than of the user. In which case we can try to copy files from that directory to other directories and launch them.

In short, **to distinguish user-system operations and actions, rely on evidence of execution, directory access and consistency of Journal and System Informer logs, keeping “bad characters” into account.**

2.1.2 Understanding Journal

USN Journal is best reliable and used when ScreenSharers already have something to cling on while analyzing system behavior. For example - no recent prefetch entries can already hint on **deletion of .pf** files and no .lnk files can hint on clearance of Recent docs or the very same .lnk files.

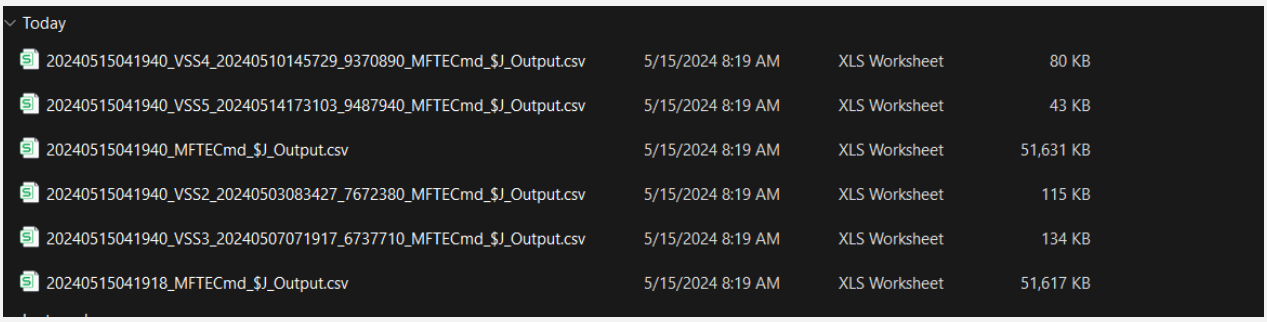


.lnk files are “link” files (thus in shell:recent) and .pf files are prefetch files - thus in the Prefetch folder. Most of this is already explained in Tome I of the Red Lotus guide (which is free) so we won’t be repeating the same information.

Unlike an unpopular opinion that unicode characters can be detected in Journal by filtering question marks, as you can see for my device, unicodes are still normally visible. Thus a conclusion: **not all unicode characters log as question marks on all systems.**

How do we get along with it? quite simple! use Timeline Explorer and exclude the entirety of the English Alphabet! well... that won’t really work, they can have extensions in English, after all... so, how can the Journal help with that and why did I bring it up? The point is, sometimes even Journal can be used to determine the full file name or file execution and modification, even if they are deleted by Alternate Data Streams.

Just search Stream Change in the MFT $UsnJrnl output and job’s done, it can even detect $DATA streams. **Oh, and don’t forget to parse VSS too MFTECmd.exe -f C:\$Extend\$UsnJrnl:$J --vss --csv .**



That way you can extract multiple $J outputs from multiple volume shadow copies of the system. Which means, if the Bypasser doesn’t know how to clear volume shadow copies then ScreenSharers will be able to (one in a thousand chances but still) recover Journal entries that are lost, overwritten or **even cleaned.** Although chances are slim and VSS usually can also auto clear itself or not create snapshots at all, it is still worth giving a try.

And yes, I didn't forget about unicode ADS, here's an example screenshot:

arent Sequence Number	Update Sequence Number	Update Reasons	File Attri
=			
393	1062144296	NamedDataExtend|NamedDataTruncation|Close	Archive
393	1062139848	NamedDataExtend|NamedDataTruncation	Archive
393	1062139672	NamedDataTruncation	Archive
393	1062055240	StreamChange|Close	Archive
393	1062054680	StreamChange	Archive

In this case, ADS modifications log as NamedData modifications (be it truncation or extend) alongside with StreamChange (while on .exe files for example it can mean file replacement - as Zone.Identifier of the file changes as well)

All and all if you cut in half entries in System Informer services, you can refer to USN Journal and filter by Name or Path - maybe you're on a system that can still read Unicode Characters.

3. Conclusion (Journal)

Journal is a non **bypassable** NTFS characteristic that logs all changes on the device. Changes can be traced using Search Everything (as well as other utilities found on Github) and monitored using Journal extracting utilities (Like Velociraptor, JournalTrace, MFTECmd) to develop bypass detections and understand the system better. It can often read Unicode characters and determine alternate data stream modifications, file-folder modifications and much more. Combined use of Registry and Journal can often be the only resources capable of detecting bypass attempts. Even though Journal can be cleared, it is simple to detect by $J and $MAX files via Event Logs or FTK Imager or MFTECmd. It is physically impossible to modify Journal, Microsoft itself and many. **If they do,** I quote DFIR Experts and Professors:

[You can delete or disable journaling and change journals with fsutil but that can have a big negative impact on any application using them to operate. If you disable or delete an active change journal all entries in the master file table (MFT) are changed to zero. I don't think you can actually get rid of MFT entirely, if that's what you're asking.]

Thus Journal is highly linked with $MFT and cannot be distinguished without. To fully grasp the abilities of Journaling, one would have to delve deeper into Windows systems - that is not the topic of this Tome nor within the range of our experience.

It is important to have something to cling on before analyzing Journal; if we still begin by analyzing USN Journal - we better have a snapshot taken before beginning the ScreenShare to avoid extending $UsnJrnl itself. It must be used with other evidence of execution or $MFT itself to determine things like Rundll, conhost / consent, regsvr (and so on) changes.

With that being said, it is important that ScreenSharers experiment this on their devices more and more - almost every day to fully grasp the possibilities. Get a file name, date - analyze.

Glossary

“1. Registry

The registry is a database of configurations stored in "configuration" files located in c:\windows\system32\config. It comprises multiple divisions called "hives" (named as such due to historical reasons), which store branches of keys and their subkeys. Important hives include SOFTWARE, SYSTEM, and SAM. Each user has their individual hive, storing settings characteristic to that specific user.

1.1 Registry Hives

Registry hives are the divisions within the registry where data is stored. They are stored in configuration files and include hives like SOFTWARE, SYSTEM, and SAM, among others. Each user has their individual hive containing settings specific to them.

1.2 Registry Values

Registry values are data stored under subkeys in the registry. They exist in different types, including string values, QWord & DWord values, expandable, binary, and multi-string values. Registry values are displayed to the right when exploring subkeys of registry hives.

1.3 Registry Value Types

Registry value types denote the format in which data is stored in the registry. Types include REG_BINARY, REG_DWORD, REG_EXPAND_SZ, REG_MULTI_SZ, REG_QWORD, and REG_SZ, among others. Each type serves a specific purpose in storing data.

1.3.1 Registry Value Manipulations

Registry value manipulations involve linking different registry value types to achieve specific functionalities. Examples include creating expandable registry values to launch applications, which can be done using PowerShell or command-line tools like reg-add.

1.4 Registry Hivelist

Registry hivelist is a subkey within the registry that shows the sources from which Regedit retrieves its data. It provides information on the dependencies of the registry, allowing users to understand the sources of registry data.

2. Registry Forensics

Registry forensics involves analyzing the Windows registry to collect evidence and identify suspicious behavior. This process helps forensic analysts understand the system's activities and identify unusual actions that may indicate malicious activity.
2.1 Analyzing Registry Changes

Analyzing registry changes involves comparing two registry snapshots using tools like RegShot. This dynamic analysis helps detect changes made to the registry by comparing snapshots before and after system

modifications or software installations. RegShot allows for easy comparison and extraction of relevant registry keys and subkeys for further analysis.

2.2 Collecting Registry Data

Collecting registry data involves using tools like RECmd to extract information from registry hives. RECmd is a command-line utility that can parse registry hives and collect data in various formats such as CSV or JSON. It can be used to search for specific strings or values within the registry and can also recover deleted keys/values.

2.2.1 Dumping Registry Data

Dumping registry data involves extracting residue data from registry hives using tools like bstrings. This can be useful for recovering deleted information before it gets overwritten. Dumping registry data allows for deeper analysis of registry contents, especially in forensic investigations.

3. Registry Explorer

Registry Explorer is a GUI-based tool for analyzing Windows registry data. It allows users to navigate through registry hives, search for specific values, and analyze registry entries. Registry Explorer comes with various features and plugins for advanced analysis of registry data, making it a valuable tool for forensic investigations and system analysis.

3.1 Analytic Approach to Value Sizes

Analyzing value sizes in the registry can provide insights into malware behavior and suspicious activities. By monitoring changes in value sizes and setting min-max size ranges, analysts can detect anomalies and potentially malicious behavior. Registry Explorer facilitates this analysis by providing tools for searching, filtering, and visualizing registry data.

3.1.1 Adding Our Own Plugins

Registry Explorer supports plugins that extend its functionality. These plugins can be developed using the provided source code and compiled into DLLs. Adding custom plugins allows users to enhance Registry Explorer with additional features tailored to their specific needs or analysis requirements.

3.1.2 Troubleshooting

Troubleshooting Registry Explorer involves ensuring proper permissions and settings for accessing registry hives. Running Registry Explorer as an administrator and enabling options like "recover deleted values" can help in retrieving accurate registry data. Troubleshooting also includes handling issues related to file paths, directory access, and UTC time zone settings.

4. Conclusion (Registry)

In conclusion, registry analysis plays a crucial role in detecting suspicious behavior and investigating system activities. Tools like Registry Explorer, RegShot, and RECmd provide valuable capabilities for collecting, analyzing, and interpreting registry data. By combining these tools with manual analysis techniques and

forensic methodologies, screen sharers can effectively identify and respond to malicious activities, ensuring system security and integrity.

Introduction to $J:

$J is a metafile on NTFS drives used to log behavioral patterns of applications and system actions.
It records actions like file deletions, attribute changes, and hex modifications, providing valuable insights for forensic analysis.
$J includes $UsnJrnl (Changes Journal) and $MAX files, which hold information about system changes and modifications to $UsnJrnl.
Using FTK Imager and $UsnJrnl:

FTK Imager can be used to analyze modification dates of $J and $UsnJrnl, providing insights into system changes.
Extracting data from $UsnJrnl is crucial for forensic analysis, especially when using tools like MFTECmd to parse the Journal.
Detection of Malicious Activities:

$UsnJrnl can be used to detect malicious activities like timestomping and hex modifications on files.
It helps in identifying suspicious user activities, detecting changes to prefetch artifacts, and identifying files with modified attributes.
Monitoring Journal and Detection of Bypasses:

Utilities like Search Everything can monitor live changes in the system and detect bypass attempts.
Tools like MFTECmd and JournalTrace are useful for extracting and analyzing data from $J.
Volume Shadow Copies (VSS) can be utilized to recover lost or overwritten Journal entries.
Understanding Journal Entries:

Journal entries without file paths are typically system-initiated actions, while those with file paths are user-initiated.
Understanding the different types of Journal entries helps in distinguishing between user and system operations.
Conclusion:

The $J metafile is an essential resource for forensic analysis, providing detailed information about system changes and user activities.
While it can be cleared, its integrity is crucial for system operation, and attempts to modify it are highly discouraged.
Combining $J analysis with other forensic techniques enhances the ability to detect and analyze bypass attempts and malicious activities effectively."

References

Advanced “USN Journal” Forensics — Haboob
The Windows USN Journal :: Velociraptor - Digging deeper!
Windows 10 Journaling | Tom's Hardware Forum (tomshardware.com)
USN Journal (forensafe.com)
fsutil usn | Microsoft Learn
Digital Forensics – NTFS Change Journal | Count Upon Security
USN Journal (forensafe.com)
Structure of the Registry - Win32 apps | Microsoft Learn
Windows Registry (What It Is and How to Use It) (lifewire.com)
Understanding the Registry on Windows - Information Technology - UConn Knowledge Base
Understanding Windows Registry, Creating Keys and Values (maketecheasier.com)
Information Technology - UConn Knowledge Base
Windows Registry Demystified: What You Can Do With It (howtogeek.com)
Windows Registry Structure: Understanding Keys, Values, and Hives in Windows Registry | by Arunkl | TheSecMaster | Medium
What Is the Windows Registry (and What Can You Do With It)? (helpdeskgeek.com)

Index

Governing Principles.....**2**
PREFACE.....**3**
CHAPTER I. Windows Registry.....**4**
1. Introduction.....4
1.1 Registry Sources and Characteristics.....4
1.2 Registry Values.....5
1.3 Registry Value Types.....5
1.3.1 Registry Value manipulations.....7
1.4 Registry Hivelist.....9
2. Registry Forensics.....9
2.1 Analyzing Registry Changes.....11
2.2 Collecting Registry data.....13
2.2.1 Dumping Registry data.....16
3. Registry Explorer.....16
3.1 Analytic approach to value sizes.....19
3.1.1 Adding our own plugins (for programmers).....20
3.1.2 Troubleshooting.....21
4. Conclusion (Registry).....21
CHAPTER I. USN Journal.....**22**
1. Introduction to $J.....22
1.1.1 FTK Imager and $UsnJrnl.....22
1.1.2 Uses of $UsnJrnl.....23
1.2.1 Resources & Reasons Table.....28
1.2.2 Monitoring Journal.....34
2.1.1 Human-System Operations.....37
2.1.2 Understanding Journal.....39
3. Conclusion (Journal).....40
Glossary.....**41**
References.....**45**
Index.....**46**
Authorization.....**47**



Authorization

Author: UnMonsieur
All rights reserved
Following material shall not be shared or sold to third parties without the author's permission.