



Purple Team

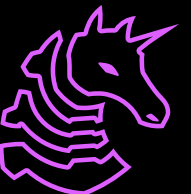
FA2025 • 2025-10-16

Windows Forensics

Bryce Kurfman

Announcements

- CCDC registration is about to be sent
Have you indicated your interest in CCDC Invitationals?
 - Be on the lookout for a GOOGLE FORM link in [#purple-announcements](#)
 - Questions? DM your Purple leads!
 - I'll collect responses thru TUESDAY 10/21
- CyberForce
 - Expect communication about this tomorrow, we'll schedule 1 meeting in advance to discuss expectations as we still await further instructions for pre-competition deliverables



ctf.sigpwny.com

sigpwny{wtf_is_a_registry}



Overview

- Windows Event Viewer
- Windows Registry:
 - Keys, Hives, & Transaction Logs
- NTFS Artifacts:
 - \$MFT, \$UsnJrnl, & \$I30, Windows Search Database
- Processes and Child Processes
- Execution Activities
 - Logs, Prefetch, SRUM, BAM, AmCache.hve, etc.
- Linking User Actions
 - Object Access Logging, LNK Files, & Shellbags
- Evidence Collection
 - Memory Acquisition w/ WinPmem
 - Triaging w/ KAPE
- **Example:** Hunting for Scheduled Task Persistence
- Active Directory GPOs

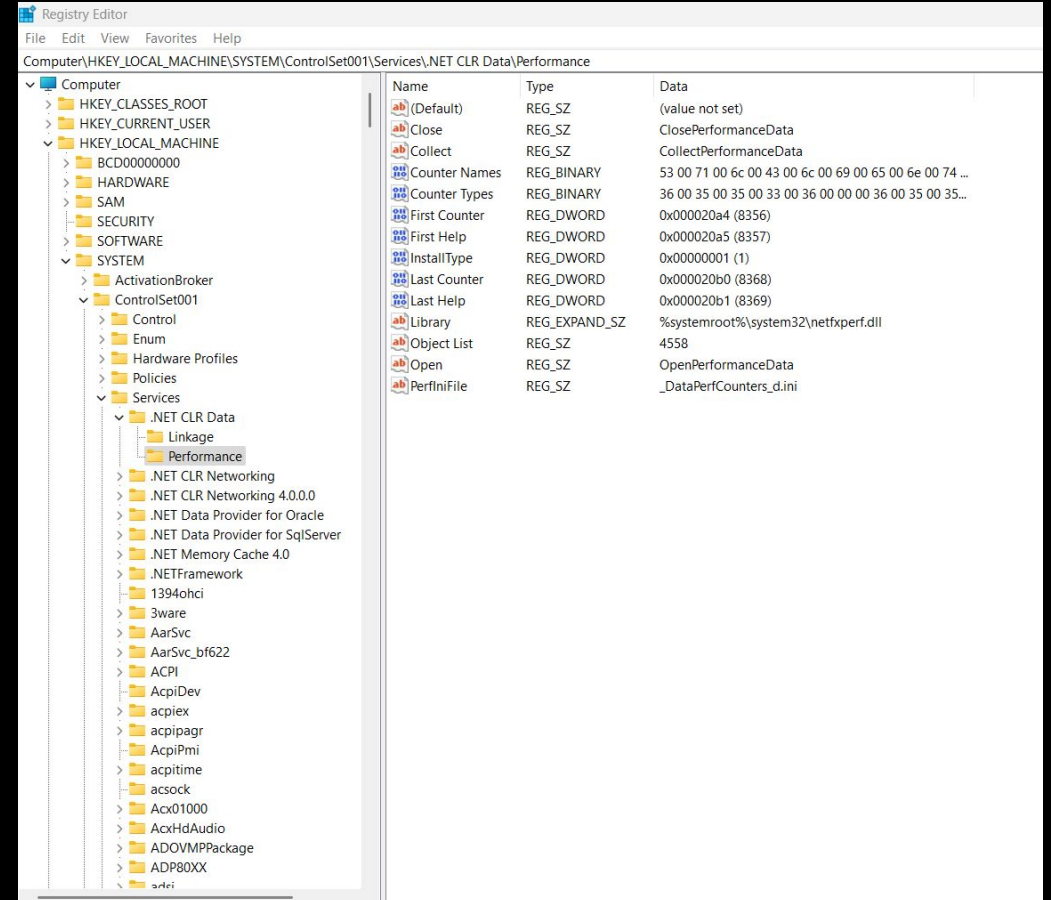


Windows Registry



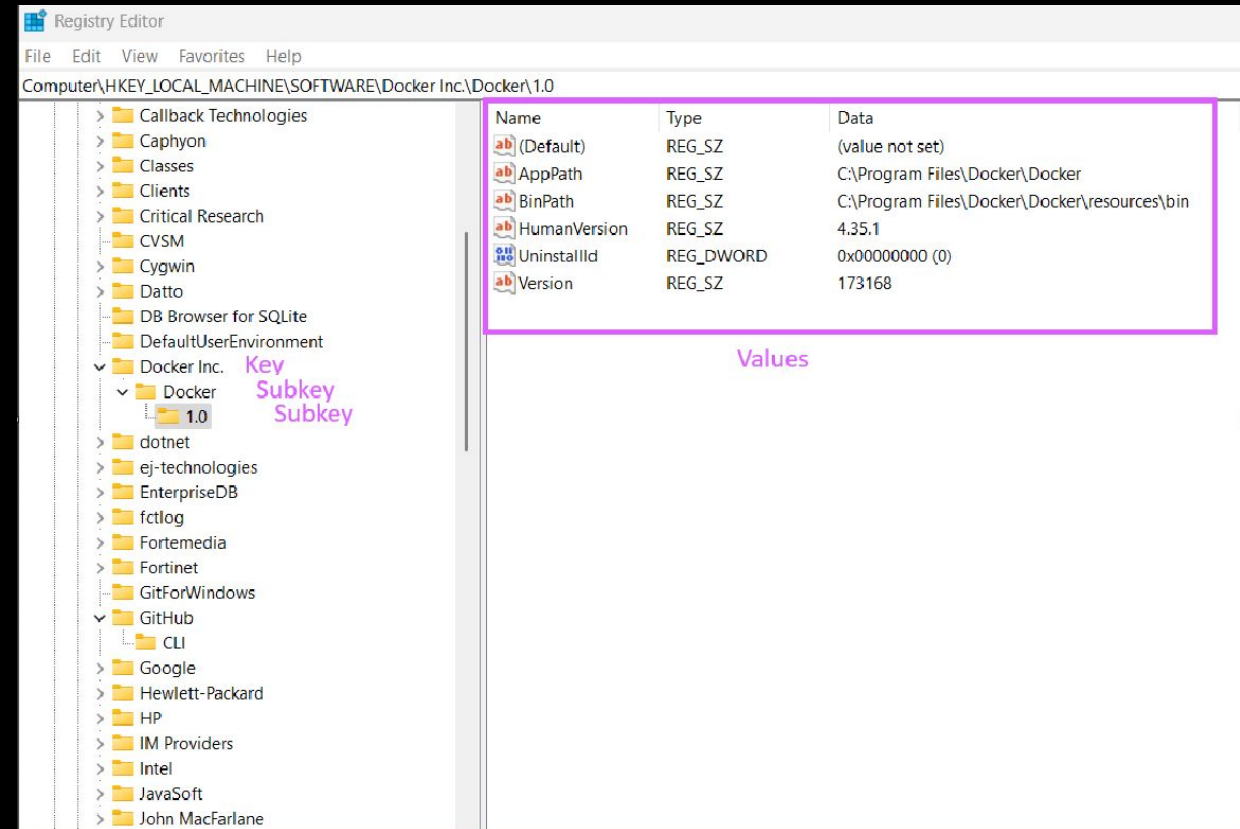
Windows Registry

- The Windows Registry is a central, **hierarchical** database that stores low-level settings and configurations
- The information stored here persists across reboots and shutdowns
- To access the registry, hit **Win + R** and type “**Regedit**”
- More info [here](#)



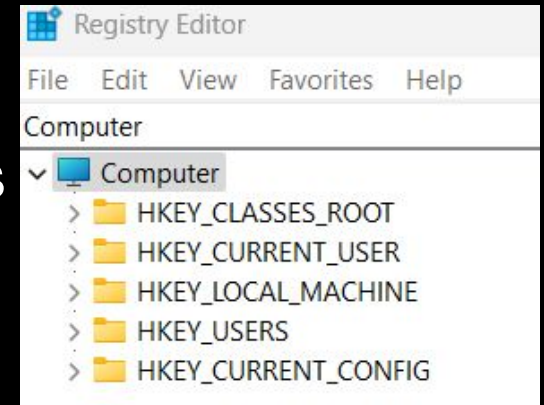
Registry Keys and Values

- The registry is structured on keys and values:
 - A **key** can contain **subkeys** and/or **values**
 - **Values** are the smallest representation and store data and configurations
 - ie. the path of a program, that program's version, other relevant information
 - Can be any number of data types



The Five Registry Subtrees

- Subtrees are the **root** of the registry, and do not contain data but instead keys, subkeys, and entries where the data is stored
- The five registry subtrees are:
 - **HKEY_USERS** (HKU): contains all the loaded user profiles
 - **HKEY_CURRENT_USER** (HKCU): profile of the currently logged-on user
 - **HKEY_CLASSES_ROOT** (HKCR): configuration information on the application used to open files
 - **HKEY_CURRENT_CONFIG** (HKCC): hardware profile of the system at startup
 - **HKEY_LOCAL_MACHINE** (HKLM): configuration information including hardware and software settings



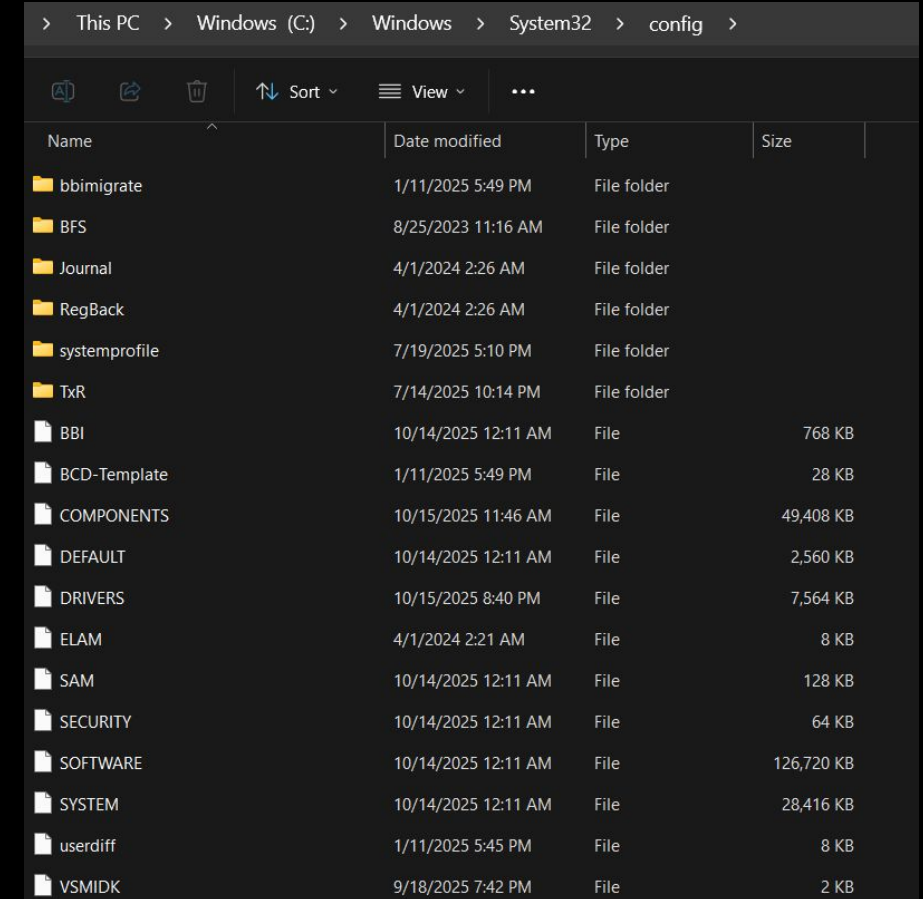
Registry Hives

- Hives are logical groups of permanent registry keys stored as disk files at `C:\Windows\System32\Config` and updated with each login
 - **SAM**: Contains information stored about the Security Accounts Manager (SAM) service, like local user accounts, passwords, creation and login dates, etc..
 - **SECURITY**: Contains the security information stored in the key `HKLM\SECURITY`.
 - **SOFTWARE**: Contains information stored in the key `HKLM\SOFTWARE` about the computer's software configuration.
 - **SYSTEM**: Contains information stored in the `HKLM\SYSTEM` about the computer's system configuration, like event log policies.
 - **DEFAULT**: Contains the default system information that is stored in the key `HKEY_USERS\DEFAULT`.



Additional Note about Hives

- There are a bunch of other hives on a Windows system, such as these:
 - **Amcache.hve**: used to track executed binaries
 - **Ntuser.dat**: used to track user-specific configs



The screenshot shows a Windows File Explorer window with the address bar set to 'This PC > Windows (C:) > Windows > System32 > config'. The window displays a list of files and folders in a table format. The columns are 'Name', 'Date modified', 'Type', and 'Size'. The files listed include folders like 'bbimigrate', 'BFS', 'Journal', 'RegBack', 'systemprofile', and 'TxR', and files like 'BBI', 'BCD-Template', 'COMPONENTS', 'DEFAULT', 'DRIVERS', 'ELAM', 'SAM', 'SECURITY', 'SOFTWARE', 'SYSTEM', 'userdiff', and 'VSMIDK'.

Name	Date modified	Type	Size
bbimigrate	1/11/2025 5:49 PM	File folder	
BFS	8/25/2023 11:16 AM	File folder	
Journal	4/1/2024 2:26 AM	File folder	
RegBack	4/1/2024 2:26 AM	File folder	
systemprofile	7/19/2025 5:10 PM	File folder	
TxR	7/14/2025 10:14 PM	File folder	
BBI	10/14/2025 12:11 AM	File	768 KB
BCD-Template	1/11/2025 5:49 PM	File	28 KB
COMPONENTS	10/15/2025 11:46 AM	File	49,408 KB
DEFAULT	10/14/2025 12:11 AM	File	2,560 KB
DRIVERS	10/15/2025 8:40 PM	File	7,564 KB
ELAM	4/1/2024 2:21 AM	File	8 KB
SAM	10/14/2025 12:11 AM	File	128 KB
SECURITY	10/14/2025 12:11 AM	File	64 KB
SOFTWARE	10/14/2025 12:11 AM	File	126,720 KB
SYSTEM	10/14/2025 12:11 AM	File	28,416 KB
userdiff	1/11/2025 5:45 PM	File	8 KB
VSMIDK	9/18/2025 7:42 PM	File	2 KB



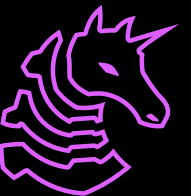
Transaction Logs History

- Prior to Windows 8.1, Transaction Logs were like a cache for the registry:
 - If we change the computer hostname, Windows would write the new hostname to the registry
 - However, Windows used caching to group series of registry updates together and writes them all in one shot (a given change is not immediate)
- These cached changes were stored in “transaction logs” on the disk and are permanently written to the registry when:
 - System idle
 - Prior to a shutdown
- If a system failure occurs before new registry changes are written from the log to the hive, they will be applied to the registry on the next boot



Transaction Logs History Cont.

- Transaction logs were written in the same directory as their corresponding registry hives and use the same file name as the hive but with a .LOG1 and .LOG2 extension
- There could be pending updates at any time in transaction logs that hadn't been written to the registry
 - It was important to inspect both transaction logs and actual registry hives to spot recent unwritten changes
- Before registry hives get updated, they are called dirty hives
 - Tools like [Registry Explorer](#) will detect dirty hives and allow you to write pending changes to registry hives



NTFS Artifacts



NTFS

- New Technology File System (NTFS)
 - Starting with Windows NT 3.1, NTFS was set as the default file system, superseding the File Allocation Table (FAT) file system
 - Is a **journaling** file system:
 - It maintains a transactional log of all changes made on any NTFS volume (e.g. file and folder creation, modification, deletion, etc.) which enables recovery from various failures like power loss, system crash, and maintain integrity
 - This means that we can find most file and folder activity information in the NTFS file system journaling files such as **\$MFT**, **\$UsnJrnl**, **\$LogFile**, and **\$I30**



\$MFT (Master File Table)

- The Master File Table (\$MFT) is a database that tracks all objects (files and folders) current changes on a NTFS filesystem.
 - Each object has its own record in the \$MFT file, containing metadata about that file, but **entries can be reused when files are deleted**
 - \$MFT is stored in the root of the NTFS partition (i.e., C:\), can acquire it with a triager like KAPE or a full disk image
- Notable \$MFT Fields
 - **In use**: if unchecked, it's a deleted object
 - **Has Ads**: indicates if this object contains Alternate Data Streams (NTFS feature that allows a file to store multiple different data types). This was created to allow Windows to read the macOS HFS file system. Attackers can use ADS to create additional stream to hide some data



Exploring the \$MFT

- To explore the contents of \$MFT, use [r-studio](#) (not the other one) or [MFTECmd](#) which produces a CSV files that can be viewed using [Timeline Explorer](#)

```
Command line: -f C:\Cases\F\%MFT --de 0
File type: Mft
Processed C:\Cases\F\%MFT in 4.9380 seconds
C:\Cases\F\%MFT: FILE records found: 108,447 (Free records: 106) File size: 106.2MB

Dumping details for file record with key 00000000-00000001
Entry-seq #: 0x0-0x1, Offset: 0x0, Flags: InUse, Log seq #: 0xAA0687A, Base Record entry-seq: 0x0-0x0
Reference count: 0x1, FixUp Data Expected: 62-00, FixUp Data Actual: 00-00 | 00-00 (FixUp OK: True)
```

[\[Link\]](#)



\$UsnJrnl

- \$UsnJrnl (Update Sequence Number Journal) provides monitoring of file and folder changes, but doesn't hold data, it contains two streams using ADS:
 - It is located at `$Extend\${USNJrnl}` under the drive root
 - Typically stores data for a few days at a time
 - `$J`
 - Holds records of changes (creations, deletions, renamings, modifications, etc.) over a file's lifecycle (whereas the \$MFT is better for current files)
 - Check [here](#) for a list of what the different attributes mean
 - Can be parsed with MFTECmd as well:
 - `MFTECmd.exe -f '<USNJRN_J$>' --csv <OUTPUTDIR_PATH>`



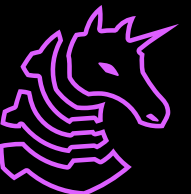
\$LogFile

- **\$LogFile** stores details of low-level changes to provide more resilience to the file system
 - Located in the volume root
 - MFTEcmd cannot parse **\$LogFile** but [NTFS Log Tracker](#) can (it can also do the other two in a single interface)
 - One of its features is “Sus behaviours detection” which can identify attempts to wipe away attack traces



\$I30

- The NTFS Index Attributes (**\$I30**) is used to track which files are in which directories
 - Each time folder content is updated, the index will be updated to reflect new changes
 - This may keep a track record of deleted files, even if securely wiped which makes it great for **proving existence** of a particular file on the system, even if it is now gone
 - Use MFTEcmd or [INDXRipper](#) to parse \$I30 file and produce a CSV file ready for Timeline Explorer



Windows Search Database

- The **Windows Search Database** stores the index to speed up searches and also helps recover deleted files
 - Even when a file is deleted, it may persist in the “windows search database” until it is updated
 - The Search Index also records various user interactions with files and browsers
 - Logs all URLs accessed via IE and MS Edge
 - Tracks file openings on a per-user basis
 - Is located at:
`C:\ProgramData\Microsoft\Search\Data\Applications\Windows\Windows.edb`
 - In windows 11, it's now `Windows.db`
- To parse and explore the contents of the database, the Search Index Database Reporter (SIDR) is great



Processes & Child Processes



Windows Processes

- When a Windows application starts, several things occur in memory:
 - Windows loads the application executable **into memory**, which creates a new process. This process acts as a container for the running memory and holds information about the application state, as well as a process ID
 - This process can also create threads that share the process's memory and other resources
- The **security privilege** that the process runs under is determined by the context of the user who ran the application
 - This can be used to enforce security restrictions and prevent the process from doing certain things like establishing network connections



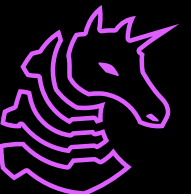
Windows Processes

- The **parent** process is the process that launched the current process
 - By seeing the relationship between parent-child, we can understand who is spawning suspicious processes
- The process **command line** can also provide important information about the application's behavior and configuration
 - ie. running **svchost.exe** without **-k** which allows the Service Control Manager to track running services



Process Analysis

- Parent processes ID (PPID)
- Process ID (PID)
- Processes creation time and exit time
- Process file path
- Process command line arguments
 - This is especially important since most adversaries will not try to hide their commands. If a process is spawned with `powershell -ep bypass -e "..."` it is most likely an attacker
- Process handle (e.g., associated files and registry keys)
- The user account the process runs under
- The security privilege the process runs under (e.g., high, medium, low)

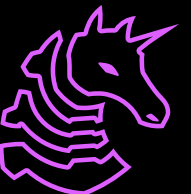


Execution Activities



Windows Services

- Configuration of Windows services is stored in the SYSTEM registry hive at: `C:\Windows\System32\config\SYSTEM` under the `CurrentControlSet\Services` (normally `ControlSet001`) key
 - Creating malicious services is a common persistence mechanism
 - Other variants of `ControlSet` keys in the system are used as backups. To determine the active `ControlSet`, check to `HKLM\SYSTEM\SELECT` key and see the loaded one under key value `current`



Important Service Event Log IDs

- System.evtx
 - **EID 7034**
 - Service crash event, possible due to process injection
 - **EID 7035**
 - OS sends a start/stop control signal to the service
 - **EID 7036**
 - Service actually starts/stops
 - **EID 7040**
 - Start type of the service is changed (e.g., auto, manual, automatic-delayed, disabled), which may indicate persistence
 - **EID 7045**
 - Similar to 4697 but does not include info about the account that installed the service



Important Service Event Log IDs Cont.

- Security.evtx
 - **EID 4697**
 - service installation, contains executable path, service name, and account that installed the service.
 - Must enable “**Security System Extension**” audit policy using the command:
“AuditPol.exe /set /subcategory:”Security System Extension” /failure:enable /success:enable”



PowerShell Script Block Logging

- PowerShell Script Block logging captures the **full content** of PS scripts, giving a lot of insight into what attackers are doing
 - This is stored in the Microsoft-Windows-PowerShell/Operational log file
 - When Script Block Logging is enabled, Event ID 4104 captures:
 - Complete script block text (the actual PowerShell code)
 - Script block ID (for multi-part scripts)
 - Path to script file (if executed from file)
- To activate, set
HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging EnableScriptBlockLogging = 1



Autorun Applications

- An attacker can persist on a system by adding malware to a list of programs that are automatically executed at system startup or when a user logs in
 - C:\Windows\system32\config\SOFTWARE:Microsoft\Windows\CurrentVersion\Run
 - C:\Windows\system32\config\SOFTWARE:Microsoft\Windows\CurrentVersion\RunOnce
 - C:\Windows\system32\config\SOFTWARE:WOW6432Node\Microsoft\Windows\CurrentVersion\Run
 - C:\Windows\system32\config\SOFTWARE:WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce
 - C:\Users<User>\ntuser.dat:Software\Microsoft\Windows\CurrentVersion\Run
 - C:\Users<User>\ntuser.dat:Software\Microsoft\Windows\CurrentVersion\RunOnce



ShimCache (AppCompatCache)

- **ShimCache**, is a feature in designed to maintain compatibility for applications running on newer operating systems
 - When an executable file is run from any source, from local drive, removable media, or network shares, an entry is made
 - Executables that are merely viewed in the Windows Explorer GUI are also recorded, even if they are not executed
 - As such, ENTRIES DO **NOT** CONFIRM EXECUTION.
 - ShimCache entries are stored in the registry at:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache
 - Use the [AppCompatCache](#) tool by Zimmerman to parse this data



AmCache.hve

- **AmCache.hve** is also known as the Windows Files Protection Cache
 - Contains a record of all the files that have been installed on a system, including name, version, and location
 - Located in C:\Windows\AppCompat\Programs, can be used to track the history of installed programs and updates. Note that it is a binary file and will need to use a tool like [AmcacheParser](#)
 - One of the most useful fields is the UnassociatedFileEntries which contains a list of all installed applications, identify any sus entries and look at their SHA1 hashes to see if they're known malicious
 - Can also use this info to create a unique blacklist/whitelist



BAM Registry Key

- Background Activity Monitor is a services that controls the activity of background applications
 - Background applications can be any type of app that can continue to run and perform tasks even when the user is to using/interacting with it, includes non-GUIs
 - Stored in SYSTEM hive under the key:
ControlSet001\Services\bam\State\UserSettings
 - Provides info about the executable files that were run, including the full file path, last execution date and time, other details



Prefetch

- Prefetch is a feature of the Memory Manager that can improve the performance of the Windows boot process and reduce the time it takes for programs to start up
 - It stores the files required by an application in the RAM as soon as the application is launched.
 - Stored in: [C:\Windows\Prefetch](#) directory with a .pf extension
 - Use the Nirsoft [WinPrefetchView](#) tool to analyze those files
 - Can extract details like the number of times a program was executed and the files associated with the executables



SRUM

- The System Resource Usage Monitor (SRUM) tracks system resource usage, like application resource usage, energy usage, Windows push notifications, network connectivity, data usage, for a period of 30 to 60 days.
 - Enabled by default, can be viewed by the normal used in Taskmanager
 - Can use SrumECmd to analyze the SRUM database located at C:\Windows\System32\sru
 - make sure to check first if SRUDB.dat needs to be repaired
 - Face Time field in SRUM database refers to the amount of time the application was actively being used by the user



Windows Event Viewer



Event Viewer

- Natively, Windows contains a built in Event Viewer application.
- Event Viewer works by logging significant events on a Windows system
 - System errors
 - Kernel power events
 - Application events
 - Security events
- EV tags these events with a unique Event ID classifier
- You filter for events where you wish to eventually establish a timeline of events. This can be useful in determining evidence for IOCs (indicators of compromise)
- Popular event IDs of interest: Logon events (4624), scheduled tasks, admin logon (4648), policy change, account management



Event Viewer Interface

Query events on right pane

Event Viewer (Local)

Custom Views

Windows Logs

- Application
- Security
- Setup
- System
- Forwarded Events

Applications and Services Logs

Subscriptions

Application

Number of events: 4,023

Level	Date and Time	Source	Event ID	Task
Information	10/7/2025 5:58:24 PM	Security-SPP	900	Noni
Information	10/7/2025 5:58:23 PM	edgeupdate	0	Noni
Information	10/7/2025 5:56:50 PM	Windows Error Report...	1001	Noni
Error	10/7/2025 5:56:49 PM	Application Error	1000	Appl
Information	10/7/2025 5:56:31 PM	RestartManager	10001	Noni
Warning	10/7/2025 5:56:31 PM	RestartManager		

Event 1000, Application Error

General

Details

Faulting application name: NVDisplay.Container.exe, version: 1.39.3323.1171, time stamp: 0x64e85748
Faulting module name: NVDisplay.Container.exe, version: 1.39.3323.1171, time stamp: 0x64e85748
Exception code: 0xc0000409
Fault offset: 0x00000000000932e5
Faulting process id: 0x1024
Faulting application start time: 0x1DC37DD999508CD
Faulting application path: C:\WINDOWS\System32\DriverStore\FileRepository\nvami.inf_amd64_b24c\Display.NvContainer\NVDisplay.Container.exe
Faulting module path: C:\WINDOWS\System32\DriverStore\FileRepository\nvami.inf_amd64_b2408e\Display.NvContainer\NVDisplay.Container.exe
Report Id: 7946adcd-294e-427c-a4bf-fdacd7283b31
Faulting package full name:
Faulting package-relative application ID:

Log Name: Application
Source: Application Error
Event ID: 1000
Level: Error
User: SYSTEM
OpCode: Info
Logged: 10/7/2025 5:56:49 PM
Task Category: Application Crashing Events
Keywords:
Computer: pkay

Security

Number of events: 25,977 (!) New events available

Keywords	Date and Time	Source	Event ID	Task Category
Audit Success	10/16/2025 3:38:40 PM	Microsoft Windows...	4798	User Account Mana...
Audit Success	10/16/2025 3:38:40 PM	Microsoft Windows...	4798	User Account Mana...
Audit Success	10/16/2025 3:38:40 PM	Microsoft Windows...	4798	User Account Mana...
Audit Success	10/16/2025 3:38:40 PM	Microsoft Windows...	4798	User Account Mana...
Audit Success	10/16/2025 3:38:40 PM	Microsoft Windows...	4798	User Account Mana...
Audit Success	10/16/2025 3:38:36 PM	Microsoft Windows...	5382	User Account Mana...
Audit Success	10/16/2025 3:38:36 PM	Microsoft Windows...	5382	User Account Mana...
Audit Success	10/16/2025 3:38:35 PM	Microsoft Windows...	4672	Special Logon
Audit Success	10/16/2025 3:38:35 PM	Microsoft Windows...	4624	Logon
Audit Success	10/16/2025 3:38:34 PM	Microsoft Windows...	5058	Other System Events
Audit Success	10/16/2025 3:38:33 PM	Microsoft Windows...	5059	Other System Events
Audit Success	10/16/2025 3:38:33 PM	Microsoft Windows...	5061	System Integrity

Event 4624, Microsoft Windows security auditing.

General

Details

Subject:
Security ID: SYSTEM
Account Name: PKAYS
Account Domain: WORKGROUP
Logon ID: 0x3E7

Log Name: Security
Source: Microsoft Windows security i
Event ID: 4624
Level: Information
User: N/A
OpCode: Info
Logged: 10/16/2025 3:38:35 PM
Task Category: Logon
Keywords: Audit Success
Computer: pkay

Actions

Application

Open Saved Log...
Create Custom View...
Import Custom View...
Clear Log...

Security

Open Saved Log...
Create Custom View...
Import Custom View...
Clear Log...
Filter Current Log...
Properties
Find...
Save All Events As...
Attach a Task To this Log...
View
Refresh
Help

Event 4624, Microsoft Windows secur...

Event Properties
Attach Task To This Event...
Copy
Save Selected Events...
Refresh
Help

Examples: App Error & Logon Events (by Michael)



Linking User Actions



Object Access Logging

- Windows does not log file/folder (object) access by default
 - We will need to enable object auditing policy on the system using the security policy through the local security policy or a Group Policy Object
 - `auditpol /set /category:"Object Access" /success:enable /failure:enable`
 - You can run this on your own machines but make sure to configure auditing on the specific objects you want, because logging everything is quite intensive

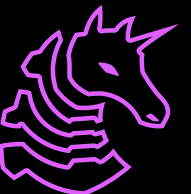
```
C:\Windows\System32>auditpol /set /category:"Object Access" /success:enable /failure:enable
The command was successfully executed.
```

- **Handle ID**
 - UID assigned to each object
 - Can use it to correlate all actions on the object.



Key Object Access Event IDs

- Security.evtx
 - **EID 4656**
 - access attempt to an object is made (success or fail)
 - **EID 4660**
 - object is deleted
 - **EID 4663**
 - access attempt (EID 4656) is successful
 - **EID 4658**
 - accessing of the object ended



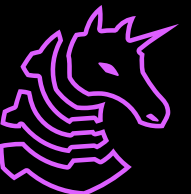
Shellbags

- Shellbags represent a log of all folders a user viewed on the system, including external drives.
 - Using shellbags, can determine what folders any user viewed and what the contents were, even if it doesn't exist anymore
- Shellbags registry keys are stored in:
 - NTUSER.DAT\Software\Microsoft\Windows\Shell\
 - NTUSER.DAT\Software\Microsoft\Windows\ShellNoRoam\
 - USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\Shell\
 - USRCLASS.DAT\Local Settings\Software\Microsoft\Windows\ShellNoRoam\
- Use [ShellbagsExplorer](#) to parse the bags



LNK Files

- LNK files are Windows shortcuts to open a file or folder and hold:
 - File attributes, the volume serial number, type of drive where the file resides, volume label, file path
 - Hostname, system MAC address
- LNK Files can be created automatically by the OS when a user opens a file or manually, using the create shortcut option.
- The best way to capture LNK files is to capture a triage image for ***.LNK** which will grab every LNK files in the selected path. Most LNK files exist in the Desktop, Downloads, or Recent folders
 - Once KAPE has been used to create a triage image, parse them use [LECmd](#)

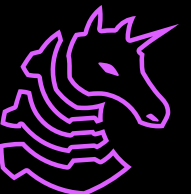


Evidence Collection



Memory Acquisition

- WinPmem is a physical memory acquisition tool that supports Windows 7 through 10, both x86 and x64
 - It needs to be run from a shell with administrator privileges
- The "mini" imagers can only produce images in RAW format, whereas the "go" version has additional features like extracting specific files from an already acquired memory image
 - For the purposes of memory acquisition, running the mini versions is better as they have a smaller footprint.
- To acquire a memory image, port the WinPmem binary to the target machine and run the following command (make sure to switch up the executable based on whether the system is 32 or 64 bit).
 - F:\WinPmem> .\winpmem_mini_x64_rc2.exe mem_output.raw
 - Make sure to save the output to external storage



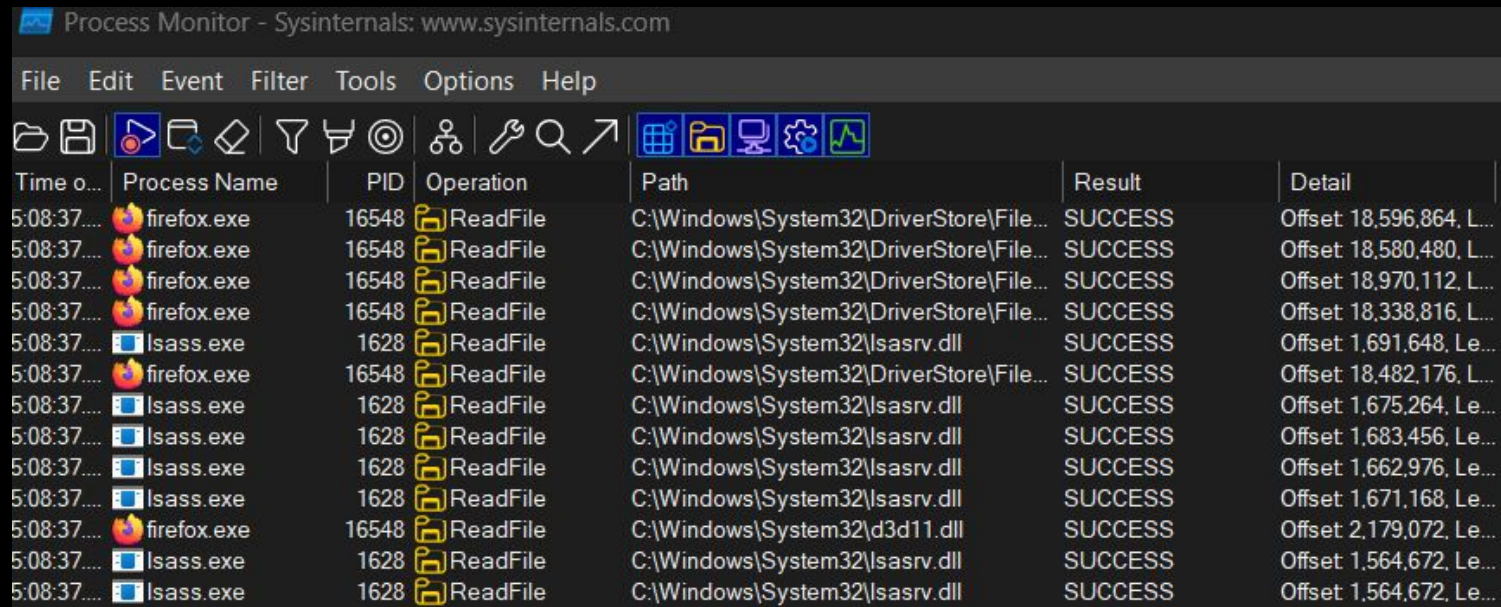
Triage Imaging

- KAPE is an efficient and highly configurable triage program that will target any device or storage location
 - From the GUI version of KAPE, you can choose what targets (what artifacts to collect) and the modules (additional parsing)
 - Can also use the command line version, with combined target and modules at once:
 - `kape.exe --tsource C: --tdest D:\Collection --target !SANS_Triage --module !EZParser --mdest D:\Results`



Procmon

- Procmon (Sysinternals) is a real-time monitoring tool that captures file system, registry, network, and process/thread activity at the kernel level.
 - Very useful for incident response and dynamic malware analysis



Time o...	Process Name	PID	Operation	Path	Result	Detail
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\DriverStore\File...	SUCCESS	Offset 18,596,864, L...
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\DriverStore\File...	SUCCESS	Offset 18,580,480, L...
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\DriverStore\File...	SUCCESS	Offset 18,970,112, L...
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\DriverStore\File...	SUCCESS	Offset 18,338,816, L...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,691,648, Le...
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\DriverStore\File...	SUCCESS	Offset 18,482,176, L...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,675,264, Le...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,683,456, Le...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,662,976, Le...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,671,168, Le...
5:08:37...	firefox.exe	16548	ReadFile	C:\Windows\System32\d3d11.dll	SUCCESS	Offset 2,179,072, Le...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,564,672, Le...
5:08:37...	lsass.exe	1628	ReadFile	C:\Windows\System32\lsasrv.dll	SUCCESS	Offset 1,564,672, Le...



Hunting for Scheduled Task Persistence



Scheduled Tasks

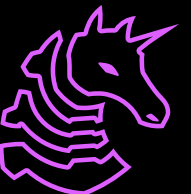
- Scheduled tasks provide a granular way to establish persistence, allow tasks to be run under certain conditions (e.g. when a particular user logs in, system reboots, other system events, etc.) and attackers can blend in with normal system behavior
- **schtasks.exe**
 - command line tool with several flags that manipulate scheduled tasks

/create	used to set up new tasks
/delete	removes tasks, can erase evidence
/run	manually runs a scheduled task immediately
/query	enumerates existing scheduled tasks on a system
/change	modifies an existing task
/end	stops a currently running task



Malicious Examples

- `schtasks.exe /create /sc daily /tn "sus_task" /tr powershell.exe -NoProfile -ExecutionPolicy Bypass -File C:\Windows\Downloads\malware.ps1 /st 04:20`
 - sus_task is created to run a PowerShell script daily with execution policy bypass to avoid script block logging
- `schtasks.exe /create /sc onstart /tn "sus_task2" /tr "dundll32.exe C:\Windows\Temp\beans.dll, evilbeans" /ru SYSTEM`
 - sus_task2 loads beans.dll and executes the evilbeans function with SYSTEM privileges



EIDs for Scheduled Tasks

- Task Scheduler Operation Log
(Microsoft-Windows-TaskScheduler/Operational)
 - **106**: Scheduled Task Created
 - **140**: Scheduled Task Updated
 - **141**: Scheduled Task Deleted
 - **200**: Scheduled Task Executed
 - **201**: Scheduled Task Completed
- **Windows Security Log**
 - **4698**: Scheduled Task Created
 - **4699**: Scheduled Task Deleted
 - **4700**: Scheduled Task Enabled
 - **4701**: Scheduled Task Disabled
 - **4702**: Scheduled Task Updated
- Sysmon (Event ID 1) and Windows Event Logs (Event ID 4688)
 - Tracks process creation, including when schtasks.exe is used



Hunting Process

- Run Queries to Find Scheduled Tasks
 - Sysmon Query (EID 1)
 - event.code: 1 AND process.name: "schtasks.exe"
 - Windows Event Logs Query (EID 4688)
 - event.code: 4688 AND process.name "schtasks.exe"
 - TaskScheduler Operational Log (EID 106)
 - event.code: 106
 - Security Log (EID 4698)
 - event.code: 4698
 - (These queries are from Elastic SIEM, practice converting these to work in Event Viewer)
- Investigate anomalous tasks, looking for:
 - Unusual task names
 - Non-standard directories
 - High-right binaries
 - scheduled task running commands like powershell.exe, rundll32.exe, or regsvr32.exe are red flags especially if they're executing scripts or DLLs directly



Hunting Process Continued

- Comprehensive Query:
 - event.code: 1 AND process.name: schtasks.exe AND process.command_line: (*/*create* OR /*delete* OR rundll32 OR regsvr32 OR powershell OR cmd)
 - this command focuses on high-risk patterns
- Remember that we can also cross-reference other logs and artifacts to build a more complete picture of what went down



Active Directory GPOs (investigative)



What is a Group Policy Object (GPO)

- On Active Directory networks where network resources and user access is defined, one can set up rules in place enforceable by this environment
- A GPO can be as simple as: “enforce password changes every X days” or “Assign X [user] to Y [group] with Z [privileges]”
- As always, *gpupdate /force*
 - Your changes go live when you do this

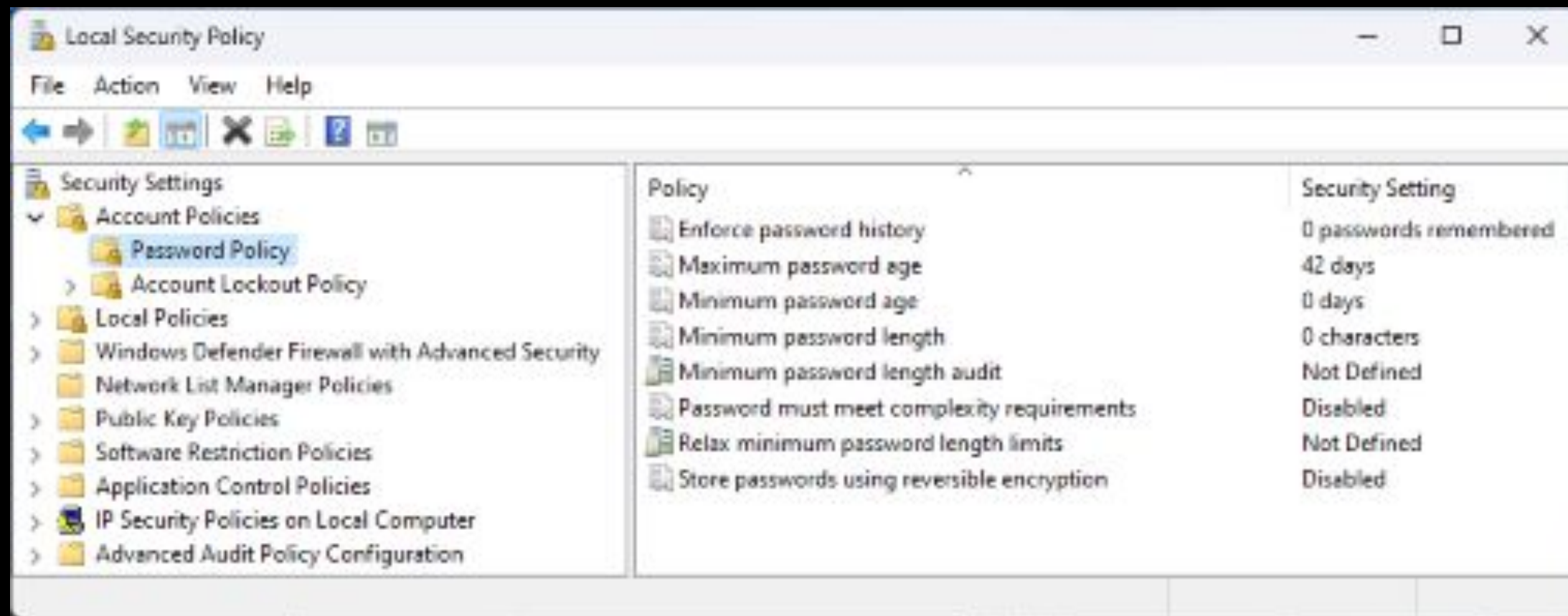


Group Policy Objects

- We'll discuss this in greater detail for system administration and system hardening
- This is relevant for the following reason: Asking ourselves what lack of a policy enabled compromise or movement within an active directory network
- Also ask yourself if certain users or groups have more power than they should within the system and if the system is properly segmented
- Remember Event IDs



Example GPO: Passwords



Next Meetings

2025-10-21 • This Tuesday

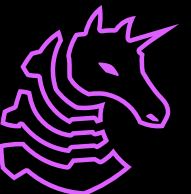
- Active Directory II
- Delegation, LAPS, DACLS, S4U, and more!

2025-10-23 • Next Thursday

- To be announced

2025-10-28 • Next Tuesday

- Active Directory III
- Asymmetric Cryptography, MSSQL, Smart Cards, cross-protocol
- attacks, and SCCM



ctf.sigpwny.com

sigpwny{wtf_is_a_registry}

Meeting content can be found at
sigpwny.com/meetings.

